**Developing Information Systems - Chapter 11 – Study Guide**

**Key Terms**

The following alphabetical list identifies the key terms discussed in this chapter. The page number for each key term is provided.

|  |  |
| --- | --- |
| Acceptance testing, 384 | PERT charts, 405 |
|  | Phased approach, 386 |
| Conversion, 386 | Process specifications, 394 |
| Customization, 390 | Production, 386 |
| Data flow diagram (DFD), 393 | Project, 397 |
| Direct cutover strategy, 386 | Project management, 397 |
| Documentation, 384 | Prototyping, 388 |
| End-user development, 389 | Rapid application development (RAD), 392 |
|  | Request for Proposal (RFP), 389 |
| Feasibility study, 383 | Scope, 397 |
| Gantt charts, 405 |
| Implementation, 404 |
| Information requirements, 382 | System testing, 384 |
| Information systems plan, 400 | Systems analysis, 382 |
| Intangible benefits, 400 | Systems design, 383 |
| Joint application design (JAD), 393 | Systems development life cycle (SDLC), 387 |
| Maintenance, 386 | Tangible benefits, 400 |
|  | Test plan, 384 |
| Organizational impact analysis, 405 | Testing, 384 |
| Parallel strategy, 386 | Unit testing, 384 |
| Structured Methodology (DFDs) | User-designer communication gap, 404 |

## Defining and Understanding the Problem

Figuring out who needs what information, where, when, and how will challenge the political dynamics of any organization. No system can answer every need, so you’re going to have to decide who gets what.

You must *think* and then *rethink* the proposed solution. Make sure you thoroughly investigate the **information requirements**—you’re going to live or die by the outcome. Whatever happens at this stage will carry through to all the other stages.

The final dilemma is whether a **new information system is really the answer**. Would it be better to address the problem through management changes, more training, or changing existing organizational processes?

## Developing Alternative Solutions

The danger in solving problems and developing systems is that only a single solution will be considered when it’s more likely that alternatives exist that could help. Using effective systems analysis provides the organization with viable choices.

## Evaluating and Choosing Solutions

Is the recommended solution even feasible? You might be surprised how often organizations fail to ask this question. A **feasibility study** helps you determine if your proposed solution is achievable *before* you spend thousands of dollars. The study will help you review the **technical feasibility** of hardware and software when you’re deciding whether your proposed answer is the right one. Too often organizations underestimate the cost of a new system, especially when it comes to people: training, downtime, lost productivity, and employee disruption. The feasibility study will help determine the **economic feasibility** of the idea.

Something’s got to give with the new system. What are the changes, who will manage them, and how will they be incorporated into the existing organizational structure? How much change can the organization handle, monetarily and emotionally? Those are the questions the **operational feasibility** study will help you answer.

## Implementing the Solution

If you get to the **systems design** stage, it means you managed to live through the analysis phase. Now you can get down to figuring out how the system will actually solve the problem or help you take advantage of new opportunities. Remember, your goal is to fit the system into the organization and not make the organization fit the new system. Or at least you want to keep them in tandem; that is, the organization should decide what technology is necessary, while the system capabilities can help reshape the organization.

When we discussed database management systems, we distinguished between two methods of viewing data: the physical design (how the data would actually be stored in the system) and the logical design (how the data would look to the user). Use the same definitions when you are designing your system, and concentrate on the logical design. The physical design should determine how the new system will support the current organizational structure, or spell out the changes in that structure that will successfully integrate the new system.

### Completing Implementation

Now that you’re through the analysis and design phases, you can move on to the remaining steps in the process. Remember, you can always go back to those two steps and probably should at some point.

*Hardware selection and acquisition:* Choosing the appropriate hardware that supports the problem solution is more than simply picking out the newest, fanciest desktop computers. PDAs, cell phone-based computing devices, wireless networks, and servers may all be part of the new configuration.

*Software development and programming:* The actual programming phase will, in all likelihood, be carried out by the IT department or outsourced to a third party. If you’re using a fourth-generation language, the programming could very well be done by the end user. Either way, make sure that the programming supports the analysis and design phases. If not, go back and work through them again. It could very well be that what was designed simply can’t be *programmed.* *The usual impulse is to program around the design flaws. Don’t! Redesign instead.*

***Testing:*** “Hey, it works!” But does it really work as it was designed in a real-world situation? Was every aspect thoroughly tested by independent testers in an actual workplace setting? Several things that go wrong in this phase of the development process can severely hamper the project’s success.

For one thing, this step is glossed over by both techies and non-techies. People assume that because something was designed and programmed according to the specifications in the analysis stage, it is right. So they just fly right through the testing process. Or they run one or two tests, usually by the very people who designed and programmed the system. “Hey, I know it works ‘cause I programmed it.” Wrong! You should never have the people who were involved with the design and programming stages do all the testing. Get a fresh pair of eyes to look at the system according to the **test plan** developed by the programmers and the users.

Most of all, if you do find a flaw in the testing, do not give into the temptation to ignore it or explain it away. Go back to the analysis, design, and programming stages. Get rid of the flaw the right way.

Of the three types of testing explained in the book, **unit,** **system,** and **acceptance (sometimes called user acceptance testing),** the last is the one that is most important and yet the most underrated. **Managers and users must be adamant about testing the system, measuring it against the analysis and design requirements, and then accepting the system only when it does in fact measure up.**

*Training and documentation:* However you convert to the new system, make sure everyone knows what’s going on. Tell them through **documentation** of a formal conversion plan and not the grapevine. Use the information you gathered in the earlier stages of the development process to help guide the implementation plan. Make sure you figure out how to convert the data and train the users. User resistance through fear of the unknown can destroy all your hard work and planning.

***Conversion****:* You’re getting close to the end. You’ve been through the agony of analyzing, designing, programming, and testing. The system meets the requirements, works right, the end users love it, and now the bosses are clamoring to see some results.

There is no right way or wrong way to implement the system; you have to look at it in the context of your particular organization.

* You can use the **parallel strategy,** but it’s expensive to run two separate systems at one time. If you don’t have a lot of confidence in your new system, you might want to go with this one.
* If you’re really confident in your development process or if the old system simply doesn’t work anymore, you can use the **direct or plunge cutover** strategy. For instance, Friday you’re using the old system; on Monday you’re using the new one. This is a high risk approach, but often the least expensive.
* If neither of the above describes your organization or your new information system, you might want to consider the **phased approach** strategy. You can introduce the new system into a single area of the organization. If all goes well there, you can install the new system in other areas. You’re still going to have to figure out how to run two systems at once and also figure out how to integrate the new system with the old system.

*Production and maintenance:* You buy a new car and think your problems with the old junker are over. Only for a while. Eventually, you’re going to have to change the oil, buy new tires, get a new air filter. Sooner or later, the new car will become an old car. The same is true with an information system.

After you install the new system and it’s in **production,** you want to go back one more time and make sure it’s meeting your needs. Eventually you’re going to have to perform **maintenance** on the system no matter how well you designed and built it. And someday you’ll have to make major changes or replace it altogether.

# Alternative Systems-Building Approaches

Just as there are multiple solutions to a problem, there are multiple methods you can use to build a system.

## Traditional Systems Development Lifecycle

The **systems development lifecycle (SDLC)** describes large- and medium-size systems projects. It has existed for years and uses tried-and-true methods that help ensure the success of the system from its humble beginnings as an idea to an old relic that eventually needs replacing.

The traditional lifecycle may seem rigid today. There are very definite roles for techies and non-techies. In fact, the user is left out of the loop on a lot of the development and implementation. Even though end users or managers have to sign off and accept the system at the end of each stage, they are not as involved in this method as some of the other methodologies.

The lifecycle approach **works well for major systems but doesn’t fit the bill for smaller ones** **or Web development**. It’s expensive, time-consuming, and sometimes doesn’t allow techies and non-techies to work together as they should.

## Prototyping

*Fast, cheap, user-centered.* **Prototyping** can be the best way to develop a new system if the end users don’t have a clue about what they really want the end product to look like. Even if they have a few clues, this approach works well because the user can guide the process based on what they see as the system is built.

Have you ever watched a television show where the police artist draws a picture of the crook as the victim looks on? The artist draws the eyes and gets approval from the onlooker. Then the mouth is sketched in and approved. Pretty soon a composite drawing is completed, and the cops are off and running. That pretty much describes the process used to build a prototype system.

Generally, you use prototyping for very small systems or small parts of a larger system. You wouldn’t want to use this method to build a company-wide information system. It can be too unstructured, making it harder to manage large projects. Prototyping **works well when you’re developing user interfaces and output reports—areas the users will see the most**.

***Review in the text the 4 steps in prototyping (there are some questions on the exam relating to this)***

Prototyping can be less costly than the traditional systems approach, but if you fail to follow some of the basic principles of systems development, it can be more costly. For instance, if you ignore the basic principle of how the prototype fits into the other information systems in the organization, or how it supports the business plan in general, you may be costing the organization more money than you realize. Did you just create an island of information that is incompatible with other systems, or is it fully supportive and easy utilized in other areas?

**The greatest advantage of the prototyping method of developing end-user interfaces is that users see the product, or at least a pretty good replica of it, right away. If they like it, you press on. If they don’t like it, changes can be made immediately. There’s less red tape and bureaucracy (perceived or otherwise) to work through with this method.**

A disadvantage of prototyping is that it does not capture performance requirements like throughput or response time.

## End-User Development (I tend to be opposed to end user development except in very limited situations)

*Taking matters into their own hands.* This method of system development is a bit like prototyping, but the end user designs and develops the new system using fourth-generation language tools. It’s convenient for small applications, and the user can have complete ownership of the system.

The tools available to the end user are getting easier to use all the time and increase the likelihood that the system will meet the user’s specifications, since the user is building it. There’s no one else to blame if the system doesn’t do what the user wants it to do. **But don’t attempt to build larger and more complex systems using this method: The capabilities of the tools are limited**.

***Managers should be aware of some inherent dangers when allowing users to develop their own systems. Standardization can be a tough issue when you use this method of system development. You’re almost begging for conflicts in data processing and storage, since each user will have her own method of creating, defining, and developing data.***

***The biggest danger is creating those islands of information. The chance of redundant end products just went up, since each user might create a system with slight differences that prohibits cross-utilization of the information.***

## Purchasing Solutions: Application Software Packages and Outsourcing

I mentioned earlier that software programs are becoming extremely complex to design, develop, and build. Many companies either don’t have the in-house staff to accomplish the task or decide to focus on their core competencies and have someone else develop the software they need.

Just as you would for any piece of equipment, you first seek **Requests for Proposal (RFP)** from several vendors to fully evaluate the software package according to your needs. Remember, you give up a lot of control when you choose to go with a prewritten software package. Well-written RFPs help minimize the impact.

### Application Software Packages

*Fast, easy, convenient, user-driven.* Many software packages are extremely convenient for non-techies to use to develop their information products. Commonly called “off-the-shelf” software, these packages can be the best method of creating an information system if that system is fairly standard across different types of businesses.

You don’t have to worry about system documentation, since that usually comes with the software. You still have to write local procedures for using the program, but you don’t have to start from scratch. Training is easier because once you learn how to use the menus and toolbars in one program, the same skills can be carried over to other programs. Training manuals often come with the program or are available through online help functions.

**Application software packages** also provide an easier method of obtaining code corrections, updates, and enhancements: simply go to the Web site of the company that wrote the software and download the latest version. Need technical support for the program? Log on to the Web and you’re there. No telephone calls, no waiting on hold for hours, no begging the IT staff to fix your problem. In fact, you’ll probably find answers to questions you didn’t even know you had!

Most of the common programs still require usage standards within the organization. For instance, if you use an accounts receivable application software program, you should still set standards for how you will adapt the program to the unique requirements of your company.

*Most off-the-shelf software can’t be changed, so you have to take what comes. Chances are some of your organization’s requirements may not be met. You’ll end up having to change your methods to match the software, instead of the other way around. Some software packages do allow some* ***customization,*** *but not as much as a program developed solely for your organization.*

### **Outsourcing**

What happens if an organization decides it doesn’t have the in-house expertise to support the system development process or any of the system maintenance required? No problem: **outsource.** There are hundreds of outside companies that will do the job. These companies offer expertise and experience, often at a lower cost than in-house staff. They can also offer smaller organizations economies of scale that make overall information processing cheaper.

The total cost of ownership of a system can be cheaper because of outsourcing. Perhaps the outsourcing company can keep up with changing technologies better than the organization. It may simply be that the organization decides to spend in-house information resource dollars in other ways.

Should you decide to use an outsourcing company to develop an information system, you must be more careful than ever to ensure that everything, right down to the smallest detail, is in writing and agreed upon by both sides. You are signing a contract with the outsourcer that carries the full force of law. You must agree on how changes will be made to the current system. How responsive will the outsourcer be to changing requirements? You still have some responsibilities for the system; what will they be? Get it in writing!

You must continually analyze the outsourcing company’s performance and cost and make sure it remains the cheaper, better way to handle the organization’s needs. At some point in time, you may find a different method is, in fact, cheaper. The table below provides information on the total cost of offshore outsourcing.

****

## Rapid Application Development for E-Business

The supply of technical specialists is not enough to support the demand for new systems, or maintenance of the old ones. Something has to fill the gap—that’s why you see so many new methods already on the market and more advanced, easier-to-use tools coming down the road. The shortage of skilled technicians is also why you see more and more companies moving away from the structured methods we’ve reviewed. There just isn’t enough time.

**Rapid Application Development (RAD)** reduces the time it takes to build systems by using many of the tools that we’ve discussed. You can choose from prototyping or fourth-generation tools to develop systems much more quickly. End users and techies can work hand-in-hand with **joint application design** (JAD) tools to reduce the development time for new applications.

# Modeling and Designing Systems

Just as there are alternative methods for system building, there are alternatives you can choose from for modeling and designing systems.

## Structured Methodologies

Traditionally, systems have been structured in a very orderly manner. The methods used to build the systems begin at the top and progress to the lowest detail always with an eye towards keeping the processes separate from the data. The designers sketch out how the data moves through the system by using **data flow diagrams**. The designers easily track all the processes and their interrelationships by using the data flow diagrams. The DFDs can be used to help spot trouble areas before the system is actually built. Below is a typical data flow diagram.

****

**Data Flow Diagram.**

The advantage of using data flow diagrams is that they can be used to show a very general, high-level process or very minute detail using the same tools. Anyone can view the overall system and then easily drill down through the diagrams to lower levels of the system. Couple the DFDs with a data dictionary and you can develop **process specifications** that describe how the data is transformed into useable information.

# Project Management

 “Gee, we thought we did everything by the book. Why doesn’t the system work the way we envisioned?” Perhaps it’s not the system itself but the way changes in the organization were managed.

## Project Management Objectives

Information system **projects** range from very small, end-user development projects, to major implementations of enterprise systems. Regardless of the size they all have some common characteristics.

First, they require the effective use of **project management** tools and technologies that help keep the project on time, within budget, and meet objectives. Every project includes the same five variables:

* Scope: the work that is or is not included in a project.
* Time: the established timeframes for each component of a project
* Cost: the amount of time multiplied by the cost of **human resources** required of a project
* Quality: the degree to which a project will improve organizational performance and decision making
* Risk: the potential problems that may threaten the project’s success

***Important note:* *Adding more people to a delayed systems development project will further delay the project because training new project members and getting them up to speed slows down team members. In addition, the communication among members becomes more complex and difficult.***

## Selecting Projects: Making the Business Case for a New System

It’s not healthy for an organization’s survival to spend money on a new system if there isn’t a strong possibility the financial gain will outweigh the cost.

### **Determining Project Costs and Benefits**

One of the more difficult choices to make when evaluating new systems is to determine the **tangible benefits** versus **intangible benefits.** For example, when a financial institution tries to decide whether to offer online banking, it may determine that it will cost half a million dollars to implement. The immediate cost savings of not having employees interface directly with customers may be only $250,000. On the surface you could say that the new system isn’t worth the cost—the bank will lose $250,000. But the intangible benefits the bank’s customers may enjoy could potentially be worth a million dollars. In that case, the new system’s intangible benefits will far exceed the tangible benefits. Below are some of the tangible and intangible benefits of information systems.



Costs and Benefits of Information Systems

### **The Information Systems Plan**

Too many companies buy the hardware they think is necessary for a new or improved information system. Then they purchase some software to go along with the new hardware. Now they realize their hardware is inadequate for the new software, so they buy more powerful hardware. And the vicious circle continues. Pretty soon they have a whole bunch of hardware and a lot of expensive software, but do they have an information system? Only if they have made sure all the hardware and software purchases fit in with their organizational **information systems plan** and their people know how to use them.

A *good* information plan will help companies systematically figure out what they need to get the job done and whether all the hardware and software is necessary and if they really do meet the requirements of the organization. A good information plan will also take personnel needs into account and help determine how all three elements of the triangle will work together for success.

The problem is that too many companies don’t have a plan for integrating new hardware and software purchases into their overall business plan, let alone meshing them with user needs and desires.

Of course, the information plan should support the overall business plan and not conflict with it. The plan must include all levels of the organization, including the strategic and executive levels.

## Managing Project Risk and System-Related Change

You’ve probably bought a can opener at some point. It looked good on the store shelf, all shiny and glittery with fancy options. Before you bought it, did you grab a can from the shelf and try it out? Probably not. You simply took it home believing it would work as you expected, and for the first few times it did. But then you tried a large size can and started experiencing glitches. Maybe the can opener broke or the can ended up crooked, or the can top was splintered and shredded. You’re cut up and bleeding. And what about all those fancy options you don’t really use but paid extra for? What you really need is a simple tool that works right every time for every job. But the designer thought it was pretty cool.

The majority of the time, information systems design problems are a direct result of the techies not understanding the non-techies and vice versa. Hardware issues win out over persware (people)issues. Perhaps there are too many bells and whistles in the system instead of solid user interface functionality.

### Implementation and Change Management

**Implementation** of a new system is *not* just about how to put the hardware and software into place. You have to address and manage the people element of the triangle and make sure that it is in sync with the hardware and software. You become a change agent. You have to convince users that the system is going to improve their work world and that the new will be better than the old. If people are going to lose their jobs because of the new system or if they are going to experience a significant difference in responsibilities, you must be clear in communicating with them.

The key elements in the success or failure of a new system are:

* User roles
* Management support
* Risk levels and implementation complexity
* Management of the implementation process

Make users feel they own the new system instead of it being an enemy or something they should fear. That’s why I stress user involvement through the entire development process. The new system shouldn’t be a surprise on Monday morning! Familiarity doesn’t always breed contempt; it should breed acceptance when it comes to new information systems.



The User-Designer Communications Gap

The table above gives a good insight into the **user-designer communications gap.** As a manager, your job is to bridge that gap to help ensure success of the new system. Too little discussion and communication between the techies and the non-techies will be apparent through design flaws and a poorly implemented project. Understand where both sides are coming from, and you’ll do a better job of getting them to work together. You can never have too *much* communication.

### **Controlling Risk Factors**

The more complex the project, the more risk. That’s easy to understand, but harder to manage. Risks associated with the project should be clearly outlined and discussed. **The three major risks are project size, project structure, and experience with technology.** The bigger the project size, the bigger the headaches. It’s pretty evident that managing two or three people working on a small system development project is easier than managing 20 people working on a huge project that will envelop the entire organization. The more structured the project, the more people will understand what is expected of them. The more people know, understand, and are familiar with the hardware and software, the easier implementation will be.

Selecting the proper people to manage the project whether they are internal or external to the organization can go a long way towards ensuring success with new projects. Automated management tools such as **PERT** or **Gantt charts** (**formal planning and control tools)** can help you manage complex projects. They are extremely beneficial for scheduling events and tracking the hundreds of details involved. You can purchase **project management software** that helps you quickly and easily create PERT and Gantt charts and keep them updated as you work through the project.

### Overcoming User Resistance

“Just what will this new system do for us?” That’s a very appropriate question but unfortunately, it’s often ignored. Everyone seems to get caught up in the hustle and bustle of the implementation process and they forget to address the **organizational impact analysis.**

I urge you to write down what you want the end result to be in terms of the organization. If you do so, you can use these notes as the basis for your impact analysis. The analysis can also be a great communication tool to explain to people how their jobs will be affected, to explain the changes required, and to help them plan the individual efforts required for a successful new system.

The human element should be at the top of your list throughout the entire implementation process so that it is incorporated into every facet of your new system.

**Questions**

**Adding more people to a delayed systems development project will further delay the project because training new project members and getting them up to speed slows down team members. In addition, the communication lines between member becomes more complex and difficult.**

**1. What are the core problem-solving steps for developing new information systems?**

* Define and understand the problem entails defining the problem and identifying its causes, solution objectives, and information requirements.
* Develop alternative solutions entails defining alternative solutions and most likely paths to follow given the nature of the problem.
* Choose the best solution entails an assessment of the technical, financial, and organizational feasibility of each alternative and selection of the best solution.
* Implement the solution entails finalizing design specifications, acquiring hardware and software, testing, providing training and documentation, conversion, and evaluating the system once it is in production.

Define information requirements and explain why they are important for developing a system solution.

Information requirements involve identifying who needs what information, where, when, and how. They define the objectives of a new or modified system and contain a detailed description of the functions the new system must perform. Gathering information requirements is perhaps the most difficult task of the systems analyst, and faulty requirements analysis is a leading cause of systems failure and high systems development costs.

List the various types of design specifications required for a new information system.

Systems design shows how the chosen solution should be realized. A system design is the model or blueprint for an information system solution and consists of all the specifications that will deliver the functions identified during systems analysis. These specifications should address all of the technical, organizational, and people components of the system solution. Table 11.1 lists the types of specifications that would be produced during system design. They include: Output, input, user interface, database, processing, manual procedures, security and controls, conversion, training and documentation, and organizational changes.

Explain why the testing stage of systems development is so important. Name and describe the three stages of testing for an information system.

Testing is critical to the success of a system because it is the only way to ascertain whether the system will produce the right results. Three stages of information system testing are:

* Unit testing — separately testing or checking the individual programs.
* System testing — the entire system as a whole is tested to determine whether program modules are interacting as planned.
* Acceptance testing — the system undergoes final certification by end users to ensure that it is ready for installation.

Describe the roles of documentation, conversion, production, and maintenance play in systems development.

* **Documentation** shows how the system works from both a technical and end-user standpoint.
* **Conversion** is the process of changing from the old system to the new system.
* **Production** is the operation of the system once it has been installed and conversion is complete. The system will be reviewed during production by both users and technical specialists to determine how well it has met its original objectives and to decide whether any revisions or modifications are needed.
* **Maintenance** involves modifications to hardware, software, documentation, or procedures to a production system to correct errors, meet new requirements, and

2. What are the alternative methods for building information systems?

Define the traditional systems lifecycle and describe its advantages and disadvantages for systems building.

The traditional systems lifecycle is a formal methodology for managing the development of systems and is still the principal methodology for medium and large projects. The overall development process is partitioned into distinct stages, each of which consists of activities that must be performed to fashion and implement an information system. The stages are usually gone through sequentially with formal “sign-off” agreements among end users and data processing specialists to validate that each stage has been completed. Users, managers, and data processing staff have specific responsibilities in each stage. The approach is slow, expensive, inflexible, and is not appropriate for many small desktop systems.

The systems lifecycle consists of systems analysis, systems design, programming, testing, conversion, and production and maintenance. Systems analysis is the phase where the problem that the organization is trying to solve is analyzed. Technical specialists identify the problem, gather information requirements, develop alternative solutions, and establish a project management plan. Business users provide information requirements, establish financial or operational constraints, and select the solution. During systems design, technical specialists’ model and document design specifications and select the hardware and software technologies for the solution. Business users approve the specifications.

During the programming phase, technical specialists translate the design specifications into software for the computer. During the testing phase, technical specialists develop test plans and conduct unit, system, and acceptance tests. Business users provide test data and scenarios and validate test results.

During the conversion phase, technical specialists prepare a conversion plan and supervise conversion. Business users evaluate the new system and decide when the new system can be put into production. During the production and maintenance phase, technical specialists evaluate the technical performance and perform maintenance. Business users use the system and evaluate its functional performance.

The advantages of using this method for building information systems include that it is highly structured; it has a rigorous and formal approach to requirements and specifications and tight controls over the system building process; it is appropriate for building large transaction processing and management information systems and for building complex technical systems. The disadvantages include: it is very costly and time-consuming; it is inflexible and discourages change even though requirements will change during the project due to the long time this method requires; it is ill-suited to decision-oriented applications that can be rather unstructured and for which requirements are difficult to define.

Define information system prototyping and describe its benefits and limitations. List and describe the steps in the prototyping process.

Prototyping builds an experimental system quickly and inexpensively for demonstration and evaluation so that users can better determine information requirements. The prototype is modified and refined until it conforms precisely to what users want. Information requirements and design are determined dynamically as users interact with and evaluate the prototype.

Prototyping is most valuable when requirements are uncertain and cannot be entirely pre-specified or when an appropriate design solution is unclear. Prototyping is especially helpful for designing end-user interfaces (screens and reports) and for determining elusive requirements of decision-support type applications. Prototyping can help reduce implementation costs by capturing requirements more accurately at an earlier point in the implementation process. It is not so useful for a very structured, well-understood, or routine problem.

It is best suited for smaller applications oriented toward simple data manipulation. Large systems with complex processing may only be able to have limited features prototyped. The prototype may be built so rapidly that design is not well thought out or must be reworked for a production environment. The problem arises when the prototype is adopted as the production version of the system without careful analysis and validation. Prototypes are built so rapidly that documentation and testing are glossed over. The system is so easily changed that documentation may not be kept up-to-date.

The steps in prototyping include identifying the users’ basic requirements; developing a working prototype of the system outlined in the basic requirements, using the prototype, and revising and enhancing the prototype based on the users’ reactions. The third and fourth steps are repeated until users are satisfied with the prototype.

Define end-user development and explain its advantages and disadvantages.

End-user development refers to the development of information systems by end users with minimal or no assistance from professional systems analysts or programmers. This is accomplished through sophisticated user-friendly software tools and gives end users direct control over their own computing.

Advantages include:

* + improved requirements determination
	+ realizing large productivity gains when developing certain types of applications
	+ enabling end users to take a more active role in the systems development process
	+ can be used for prototyping
	+ works well for adding new functions such as graphics, modeling, and ad-hoc information retrieval

Disadvantages include

* + not being suited for large transaction-oriented applications or applications with complex updating requirements
	+ standards for testing and quality assurance may not be applied
	+ proliferation of uncontrolled data and private information systems

End-user development is suited to solving backlog problems because the end users can develop their needed applications themselves. It is suited to developing low-transaction systems. End-user development is valuable for creating systems that access data for such purposes as analysis (including the use of graphics in that analysis) and reporting. It can also be used for developing simple data-entry applications.

Describe the advantages and disadvantages of developing information systems based on application software packages.

Software packages provide several **advantages**:

* + The vendor has already established most of the design that may easily consume up to 50 percent of development time.
	+ Programs are pretested, reducing testing time and technical problems.
	+ The vendor often installs or assists in the installation of the package.
	+ Periodic enhancement or updates are supplied by the vendor.
	+ Vendors also maintain a permanent support staff well versed in the package, reducing the need for individual organizations to maintain such expertise in-house.
	+ The vendor supplies documentation.

The usage of software packages has several **disadvantages**:

* + There are high conversion costs for systems that are sophisticated and already automated.
	+ Packages may require extensive customization or reprogramming if they cannot easily meet unique requirements.
	+ A system may not be able to perform many functions well in one package alone.

Define outsourcing. Describe the circumstances in which it should be used for building information systems. List and describe the hidden costs of offshore software outsourcing?

Outsourcing is the process of turning over an organization’s computer center operations, telecommunications networks, or applications development to external vendors who provide these services. Outsourcing is an option often considered when the cost of in-house information systems technology has risen too high. Outsourcing is seen as a way to control costs or to develop applications when the firm lacks its own technology resources to do this on its own.

Many firms underestimate costs for identifying and evaluating vendors of information technology services, for transitioning to a new vendor, for improving internal software development methods to match those of outsourcing vendors, and for monitoring vendors to make sure they are fulfilling their contractual obligations. Outsourcing offshore incurs additional costs for coping with cultural differences that drain productivity and dealing with human resources issues, such as terminating or relocating domestic employees. These hidden costs undercut some of the anticipated benefits from outsourcing. Firms should be especially cautious when using an outsourcer to develop or to operate applications that give it some type of competitive advantage. Figure 11-5 shows best- and worst-case scenarios for the total cost of an offshore outsourcing project. Hidden costs increase the total cost of an offshore outsourcing project by an extra 15 to 57 percent.

Explain how businesses can rapidly develop e-business applications?

RAD is a process for developing systems in a very short time period by using prototyping, fourth-generation tools, and close teamwork among users and systems specialists. RAD allows the creation of working software in a very short time through objects and automation of much of the code generation. Usually they depend on interfaces to databases.

3. How should information systems projects be selected and evaluated?

Explain the difference between tangible and intangible benefits?

Tangible benefits can be quantified and assigned a monetary value. Intangible benefits are classified as nonquantifiable and cannot be assigned a monetary value.

List six tangible benefits and six intangible benefits.

Tangible benefits include: increased productivity, lower operational costs, reduced workforce, lower computer expenses, lower outside vendor costs, lower clerical and professional costs, reduced rate of growth in expenses, reduced facility costs, and increased sales.

Intangible benefits include: improved asset utilization, improved resource control, improved organizational planning, increased organizational planning, increased organizational flexibility, more timely information, more information, increased organizational learning, legal requirements attained, enhanced employee goodwill, increased job satisfaction, improved decision making, improved operations, higher client satisfaction, and better corporate image.

4. How should information systems projects be managed?

Explain the importance of implementation for managing the organizational change surrounding a new information system.

The term implementation refers to the entire process of organizational change surrounding the introduction of a new information system. Information systems design and the entire implementation process should be managed as a planned organizational change using an organizational impact analysis. A very large percentage of information systems fail to deliver benefits or solve the problems for which they were intended because the process or organizational change surrounding system building was not properly addressed. The principal causes of information system failure are (1) insufficient or improper user participation in the systems development process, (2) lack of management support, (3) high levels of complexity and risk, and (4) poor project management.

Define the user-designer communication gap and explain the kinds of implementation problems it creates.

The user-designer communication gap deals with the relationship that exists between end users and information systems specialists. These two groups have different backgrounds, interests, and priorities and has traditionally been a problem for information systems implementation efforts. These differences create user-designer communications gaps. Information systems specialists often have a highly technical orientation to problem solving, focusing on technical solutions in which hardware and software efficiency is optimized at the expense of ease of use or organizational effectiveness. End users prefer systems that are oriented toward solving business problems or facilitating organizational tasks.

List and describe the factors that influence project risk and describe strategies for minimizing project risks.

Strategies you can follow to increase the chances of a successful system include:

* New systems that involve challenging and complex technology can be helped by recruiting project leaders with strong technical and administrative experience.
* If the firm does not have staff with the required technical skills or expertise, outsourcing or using external consultants are options that may be pursued.
* Using formal planning and control tools, such as Program Evaluation and Review Technique (PERT) or Gantt charts improve project management by listing the specific activities that make up a project, their duration, and the sequence and timing of task.
* Promote user participation by making user education and training easily available, and by providing better incentives for users who cooperate.
* Exercise sensitivity to ergonomic issues.
* Solve organizational problems prior to introducing new systems.
1. Discuss the role of business end users and information system professionals in developing a system solution. How do both roles differ when the solution is developed using prototyping or end-user development?

Business end users are the people who actually use the system. It is critical that these individuals play a role in any systems development efforts. Their input greatly enhances the whole systems development process. Business end users who are involved in systems development projects feel more “ownership” towards it, and will strive harder to ensure that the system is successful. No matter how well a system is designed, without user acceptance it will suffer major consequences, and many actually fail.

The role of the information system professional is to design a system that meets both the needs of the organization and its end users. The role of an information system professional is to clearly understand the needs of the organization and those of the people who will ultimately be the users of the system. Both groups achieve common organizational goals by keeping communication lines open and involving different levels of personnel.

When a solution is developed using prototyping, the system is rapidly and inexpensively developed for end users to interact with and evaluate. The prototype is refined and enhanced until users are satisfied that it includes all of their requirements and can be used as a template to create the final system. Prototyping encourages end-user involvement in systems development and iteration of design until specifications are captured accurately. The rapid creation of prototypes can result in systems that have not been completely tested or documented or that are technically inadequate for a production environment.

1. Systems fail when systems builders (IT people) ignore “people” problems. Why is this so?

System building efforts often fail because there is too much emphasis on the technology and not enough attention to changes in organizational structure, job design, workflows, and reporting relationships. Inattention to these issues often breeds resistance to a new system and may also produce a system that is incompatible with the organization. Conflicts between the technical orientation of system designers and the business orientation of end users must also be resolved for successful implementation of systems. The success or failure of organizational change can be determined by how well information systems specialists, end users, and decision makers deal with key issues at various stages of implementation.