Q.2 a. Who is the system analyst and what are the roles played by system analyst?

Answer:

Ans: The Role of the Systems Analyst Systems analysts understand both business and computing. They study business problems and opportunities and then transform business and information requirements into specifications for information systems that will be implemented by various technical specialists including computer programmers. Computers and information systems are of value to a business only if they help solve problems or effect improvements. Systems analysts initiate change within an organization. Every new system changes the business. Increat4ngly, the very best systems analysts literally change their organizations—providing information that can be used for competitive advantage, finding new markets and services, and even dramatically changing and improving the way the organization does business. The systems analyst is basically a problem softer throughout this book; the term problem will be used to describe many situations. Including:

- Problems, either real or anticipated, that require corrective action.
- Opportunities to improve a situation despite the absence of complaints.
- Directives to change a situation regardless of whether anyone has complained about the current situation.

4+4=8

The systems analyst's job presents a fascinating and exciting challenge to many individuals. It offers high management visibility and opportunities for important decision making and creativity that may affect an entire organization. Furthermore, this job can offer these benefits relatively early in your career (compared to other entry-level jobs and careers).

b. With the help of a diagram explain the role of the network in Information Systems.

Answer: Page Number 58 of Text Book

Q.3 a. What are the resemblance in system life cycle and system development methodology?

Answer:

Ans: The terms system life cycle and system development methodology are frequently and incorrectly interchanged. Most system development processes are derived from a natural system life cycle. The system life cycle just happens. Notice that there are two key events that trigger a change from one stage to the other: When a system cycles from development to operation and maintenance. At some point in time, obsolescence occurs (or Is imminent) and a system cycles from operation and maintenance to redevelopment. Actually, a system may be in more than one stage at the same time. For example, one version may be in operation and support while the next version Is in development So how does this contrast with a systems development methodology? A systems development methodology "executes" the systems development stage of the system life cycle. Each individual information system has its own system life cycle. The methodology is the standard process to build and maintain that system and all other Information systems through their life cycles. Consistent with the goals of the CMM, methodologies ensure that:

- A consistent, reproducible approach is applied to all projects.
- There is reduced risk associated with shortcuts and mistakes.
- Complete and consistent documentation is produced from one project to the next.

4

Systems analysts, designers, and builders can be quickly reassigned between projects because all use the same process. As development teams and staff constantly change, the results of prior work can be easily retrieved and understood by those who follow. Methodologies can be purchased or home-grown. Why purchase a methodology? Many information system organizations can't afford to dedicate staff to the development and continuous improvement of a home-grown methodology. Methodology vendors have a vested interest in keeping their methodologies current with the latest business and technology trends. Home-grown methodologies are usually based on generic methodologies and techniques that are well documented in books and on Web sites. Examples of system development methodologies are listed in the margin on the following page. You should be able to research most of them on the Web. Many of their underlying methods will be taught in this textbook. Throughout this book, we will use a methodology called FAST; which stands for Framework for the Application of : Systems Thinking. FAST is not a real-world commercial methodology We developed it as a composite of the best practices we've encountered in many commercial and reference methodologies. Unlike many commercial methodologies FAST is not prescriptive. That is to say. FAST is an agile frame-work that is flexible enough to provide for different types of projects and strategies. Most important FAST shares much in common with both the book-based and the commercial methodologies that you will encounter in practice.

b. Differentiate Sequential v/s Iterative development process.

Answer:

Ans: The above discussion of phases might lead you to assume that systems development Is a naturally sequential process. Moving in a one-way direction from phase to phase Such sequential development is in fact, one alternative.
We have used the four classic phases rather than the eight FAST phases in the interest of simplicity. This strategy requires that each phase be "completed" one after the other until the information system is finished. In reality, the phases may somewhat overlap one another in time. For example, some system design can be started prior to the completion of system analysis.

Given its waterfall-like visual appearance, this approach is often called the waterfall development approach. The waterfall approach has lost favor with most modem system developers. A more popular strategy is commonly referred to as the Iterative development approach, or incremental development process. This approach requires completing enough analysis. Design and implementation to be able to fully develop apart of the new system and place it into operation as quickly as possible.
Once that version of the system is implemented the strategy is to then perform some additional analysis, design, and implementation to release the next version of the system. These iterations continue until all parts of the entire information system have been implemented the popularity of this iterative and incremental process can be explained simply: System owners and users have long complained about the excessive time required to develop and implement information systems using the waterfall approach. The iterative approach allows versions of useable information to be delivered in regular and shorter time frames. This results in improved customer (system owner and user) satisfaction.

Q.4 a. Describe the Accelerated Systems Analysis Approaches briefly.

Answer:

Ans: Discovery prototyping and rapid architected development are examples of accelerated systems analysis approaches that emphasize the construction of prototypes to more rapidly identify business and user requirements for a new system. Most such approaches derive from some variation on the construction of prototypes, working but incomplete samples of a desired system. Prototypes cater to the 'Ill know what I want when I see it" way of thinking that is characteristic of many users and managers. By 'incomplete," we mean that a prototype will not include the error checking, input data validation security and processing completeness of a finished application. Nor will it be as polished or offer the user help as in a final system. But it can be developed quickly. It can quickly identify the most crucial of business-level requirements. Sometimes, prototypes can evolve into the actual, completed information systems and applications. In a sense, accelerated analysis approaches place much emphasis on the continuation building blocks in your information system framework by constructing sample forms and reports. At the same time, the software tools used to build prototypes also address the DATA and PROCESS building blocks. These accelerated approaches are common in the rapid application development (RAP) methodologies and routes that were introduced in Chapter 3. RAP approaches require automated tools. While some repository-based CASE tools include very simple RAP facilities, most analysts use true RAP programming environments such as Sybase Power builder Microsoft Access. Microsoft Visual Basic NET or IBM Websphor Studio for Application Development (Java-based). Let's briefly examine two popular accelerated analysis approaches.
Discovery Prototyping Discovery prototyping uses rapid development technology to help users discover their business requirements. For example, it is very common for systems analysts to use a simple development tool like Microsoft Access to rapidly create a simple database, user input forms, and sample reports to solicit user responses as to whether the database, forms, and reports truly represent business requirements. The intent is usually to develop the final new system in a more sophisticated application development tool and language. but the simpler tool allows the analyst to more quickly prototype the user's requirements. In discovery prototyping, we try to discourage users from becoming preoccupied with the final look and feel' of the system prototypes—that can be changed during system design! Therein lies the primary criticism of

prototyping—software templates exist in prototyping tools to quickly generate some very elegant and visually appealing prototypes. Unfortunately this can encourage a premature focus on. and commitment to design represented in the prototype. Users can also be misled to believe (I) that the completed system can be built Just as rapidly or (2) that tools like Access can be used to build the final system. While tools like Access can indeed accelerate systems development, their use in discovery prototyping Is fast only because we omit most of the de-tailed database and application programming required for a complete and secure application. Also, tools like Access typically cannot support the database sizes, number of users, and network traffic that are required of most enterprise applications. Regardless, discovery prototyping is a preferred and recommended approach. Unfortunately, some systems analysts and developers are using discovery prototyping to completely replace model-driven design, only to learn what true engineers has e known for years: you cannot prototype without some amount of more formal design enter rapid architecture analysis.

Rapid Architected Analysis Rapid architected analysis is an accelerated analysis approach that also builds system models. Rapid architecture analysis is made possible by reverse-engineering technology that is included in many automated tools such as CASE and programming languages. Reverse-engineering tools generate system models from existing software applications or system prototypes. The resulting system models can then be edited and improved by systems analysts and users to provide a blueprint for a new and improved system. It should be apparent that rapid architected analysis is a blending of model-driven and accelerated analysis approaches. There are two different techniques for applying rapid architected analysis: Most systems have already. been automated to some degree and exist as legacy information systems. Many CASE tools can read the underlying data-base structures and/or application programs and reverse engineer them into various system models. Those models serve as a point of departure for model-driven user requirements analysis. If prototypes have been built into tools like Microsoft Access or Visual Basic, those prototypes can sometimes be reverse engineered into their equivalent system models. The system models usually better lend themselves to analyzing the users' requirements for consistency, completeness, stability scalability and flexibility to future change. Also, the system models can frequently be forward engineered by the same CASE tools and ADB (application development environments) into databases and application templates or skeletons that will use more robust enterprise-level database and programming technology Both techniques address the previous issue that engineers rarely prototype in the total absence of a more formal design, and, at the same time, they preserve the advantages of accelerating the systems analysis phases.

b. Give an analysis on logic design phase.

Answer:

Not all projects embrace model-driven development, but most include some amount of system modeling. A logical design further documents business requirements using system models that illustrate data structures, business processes, data flows, and user Interfaces (increasingly using object models, as introduced earlier in the chapter). In a sense, they validate the requirements established in the previous phase. Once again, your information systems building blocks (Figure 5-16) can serve as a useful framework for documenting the information systems requirements. Notice that we are still concerned with the sys-rEm USERS' perspectives. In this phase, we draw vari-ous system models to document the requirements for a new and improved system. The

models depict various aspects of our building blocks. Alternatively, prototypes could be built to - discover requirements." Discovery prototypes were introduced earlier in the chapter. Recall that some prototypes can be reverse engineered into system models. Figure 5-17 illustrates the typical tasks of the logical design phase. The final phase deliverable and milestone is producing a BUSINESS REQUIREMENTS sratemasrr that will ful-fill the system improvement objectives identified in the previous phase. One of the first things you may notice in this task diagram is that most of the tasks are not as se-quential as in previous task diagrams. Instead. many of these tasks occur in parallel as the team works toward the goal of completing the requirements statement. The logical design phase typically includes the following tasks: 4.1a Structure functional requirements. 4. Ib Prototype functional requirements. 4.2 Validate functional requirements. 4.3 Define acceptance test cases. Let's now examine each of these tasks in greater detail.

c. Describe Decision Analysis Phase in brief.

Answer:

Ans: Given the business requirements for an improved information system, we can finally address how the new system—Including computer-based alternatives might be implemented with technology. The purpose of the decision analysis phase is to identify candidate solutions, analyze those candidate solutions, and recommend a target sys-tem that will be designed, constructed, and implemented. Chances are that someone has already championed a vision for a technical solution. But alternative solutions, per-haps better ones, nearly always exist. During the decision analysis phase, it is imperative that you Identify options, analyze those options, and then sell the best solution based on the analysis. Once again, your information systems building blocks (Figure 5-18) can serve as a useful framework for the decision analysis phase. One of the first things you should notice is that information technology and architecture begin to influence the decisions we must make. In some cases, we must work within standards. In other cases we can look to apply different or emerging technology. You should also notice that the perspectives are in transition—from those of the SYSTEM VMS to those of the system DESIGNERS. Again, this reflects our transition from pure business concerns or technology. But we are not yet designing. The building blocks indicate our goal as developing a proposal that will fulfil requirements .The typical tasks of the decision analysis phase .The final phase deliverable and milestone is producing a PROPOSAL that will fulfil the business requirements identified in the previous phase. The decision analysis phase typically includes the following tasks:

1.Identify candidate solutions.
2. Analyze candidate solutions.
3. Compare candidate solutions.
4. Update the project plan.
5. Recommend a system solution.

(4)

Q.5 a. Describe the four types of actors in use case modelling.What is the purpose of the use-case ranking, priority matrix and use case dependency diagram?

Answer: Page Number 247 of Text Book

b. Write a note on Key-based data model.

Answer: Page Number 292 of Text Book

c. Define the following by giving suitable examples:-
(i) Cardinality of a relationship (ii) Degree of a relationship

Answer: Page Number 275 of Text Book

Q.6 a. Describe the history of object modelling.

Answer:

The object-oriented approach is cantered around a technique referred to as object modelling. The object modelling technique prescribes the use of methodologies and diagramming notations that are completely different from the ones used for data modelling and process modelling. In the late 80s and early 90s many different object-oriented methods were being used throughout

industry The most notable of these were Grady Booth's Booth Method. James Rumbaugh's Object Modelling Technique (0311), and Kai.Jacobson's Object-Oriented Sofia one Engineering (00SE). The existence of so many methods and associated modelling techniques was a major problem for the object-oriented system development industry. It was not uncommon for a developer to have to learn several object modelling techniques depending on what was being used on the project at the time. Because so many were being used, this was limiting the ability to share models across projects (reduced reusability) and develop-ment teams. Consequently, it hampered communication between team members and users, which led to many errors being introduced into the project. These problems and others led to the effort to design a standard modelling language.

(8)

b. What is polymorphism? When is it applied?

Answer: Page Number 380 of Text Book

c. What do you understand by -The "Buy" Solution from Systems Design for Integrating Commercial Software?

Answer:

ANS:

Let's now confine systems design for solutions that involve acquiring a commercial off-the-shelf (COTS) software product. The life cycle for projects that involve purchase or -buy," solutions. Notice that the business requirements statement (for software) and its integration as a business solution trigger a series of phases absent from the in-house development process we just learned about. The most notable differences between the buy and the in-house development projects is the inclusion of a new procurement phase and a special decision analysis phase (process labelled -5A") to address software and services. When new software is needed, the selection of appropriate products is often difficult. Decisions are complicated by technical, economic, and political considerations. A poor decision can ruin an otherwise successful analysis and design. The systems analyst is becoming increasingly involved in the procurement of software packages (as well as peripherals and computers to support specific applications being developed by that analyst). The purpose of the procurement and decision analysis phases is to do the following: I. Identify and research specific products that could support our recommended solution for the target information system. 2. Solicit. Evaluate, and rank vendor proposals. 3. Select and recommend the best vendor proposal. 4. Contract with the awarded vendor to obtain the product. In this section we will examine the tasks involved in completing the procurement and decision analysis phases for a buy solution. As a buy solution affects how other phases in the life cycle

are also completed (phases that are impacted are shaded in light blue). Alter examining the procurement and decision analysis phases, we will explore the impacts that a buy solution would have on how those phases would he completed. Is a task diagram depicting the work (= tasks) that should be per-formed to complete the procurement and decision analysis phases for a buy project solution. This task diagram does not mandate any specific methodology but we will describe in the accompanying paragraphs the approaches, tools, and techniques you might want to consider for each design task. This task diagram Is only a template. The project team and project manager may expand on or alter the template to reflect the unique needs of any given project. The first two tasks (4.1 and 4.2) are procurement phase tasks, and the remaining tasks are decision analysis-related tasks. Let's now examine each task in detail.

Q.7 a. What do you mean by the term human factors?

Answer:

Before designing user interfaces, you may find it useful to understand the elements that frequently cause people to have difficulty with computer systems. Our favourite user interface design expert, Wilbert Ga.litz (see the Suggested Readings) offers the fol-lowing interface problems: • Excessive use of computer jargon and acronyms. • No obvious or less-than-intuitive design. • Inability to distinguish between alternative actions (-What do I do near). • Inconsistent problem-solving approaches. • Design Inconsistency According to Galitz, these problems result in confusion, panic. frustration, boredom, misuse. abandonment, and other undesirable consequences. To solve these problems Galitz offers the following overriding-commandments* of user interface design: • Undo stand your users and their tasks. This becomes increasingly difficult as we extend our information systems to implement business-to-consumer (B2C) and business-to-business (B2B) functionality using the Internet. allows you to call attention to something important—for example. the next field to be entered, a message, or an instruction. Default values for fields and –teem to be entered by the user should be specified. In windowing environments, valid values are frequently presented in a separate window or dialogue box as a scrollable region. The default value, if applicable, should usually be first and clearly highlighted. Anticipate the errors users might make. System users will make errors, even when given the most obvious instructions. If it is possible for the user to execute a dangerous action, let it be known (for example, a message or dialogue box could read ARE NOV SURE YOU TO DELETE THIS FILE?). RR ounce of prevention goes a long way! With respect to a user should not be allowed to proceed without correcting an error Instructions (and examples) on how to correct the error should be displayed. The error can be highlighted with sound or colour and then explained in a pop-up window or dialogue box .A option can be defined to trigger display of additional instructions .If the user does something that could be catastrophic. The keyboard should be locked to prevent any further input and an instruction to call the analyst or technical support should be displayed.

b. Discuss the special considerations for user interface design?

Answer:

In addition to establishing a user interface style, analysts must address certain special considerations for user interface design. How will users be recognized and authenticated to use the system? Are there any security or privacy considerations to be accommodated in the user interface? Finally, how will users get help via the user interface?

Internal Controls—Authentication and Authorization In most environments, system users must be authenticated and authorized by the system before they are permitted to perform certain actions. In other words, system users must log into" the system. Most logins require both a USER to and a System users should not be required to learn and memorize multiple USER IDS and PASSWORDS. Ideally they should be required to use the same log-in as is used for their local area network account. (Windows and 2000 allow for this authentication to occur without the need to retype either field.) Figure 17-10(a) demonstrates the user interface for the SoundStage log-in. The USER ID and PASSWORD will be authenticated against the network accounts file. Notice that the password is printed as asterisks as the user types it in, a common security and privacy measure. Should the user ID or password fail to be authenticated, the security authorization dialogue in Figure 17-10(b) will be displayed. Authentication is only half of the solution. Once authenticated, the user's access and service privileges for his information system must be established. There are many models for establishing and managing privileges. An important guideline is to assign privileges to roles, not to individuals. In most businesses people change jobs routinely—they are reassigned and promoted to new job responsibilities and roles. Also job descriptions and roles change from time to time. Finally, people leave the business and some are terminated. For all of these reasons, privileges should be assigned to roles. Then it is a simple matter of identifying the roles that any user ID can assume. For each role, the specific privileges that should be assigned to the role need to be defined. Privileges may include permission to read specific tables or views: permission to create, update, or delete records (rows) in specific tilts or views; permission to generate and view specific reports: permission to execute specific transactions; and the like. Although not technically part of the interface, defining these roles and permissions is needed both to design an appropriate log-in interface and to functionally specify the complete authentication and authorization security model for the system. Different user views could actually be applied to customize the user interface for different categories of users. For example, it is fairly easy to -ghost" (change the font from black to gray) and disable those menu options and dialogue boxes that are to be restricted from certain classes of system users. With the emergence of e-commerce. Consumers and other business must have confidence that we are who we claim to be. Consumers may be providing credit card numbers and other private information for transmission over the Internet. For this reason Sound Stage purchased a Web certification to authenticate Itself to its club members and prospective members .At any time, using the browser interface. Sound Stage members can view the authentication certificate. With this certification, the Sound Stage Web site will display a "Secure Server Certification-icon (see margin—the padlock) that will tell consumers their data will be encrypted (securely scrambled) to ensure that their credit card and personal data is not being intercepted or accessed by others when passed along the network.

Q.8 a. Explain the terms Control Classes, Persistence Classes, System Classes in brief.

Answer:

Ans :

Control Classes

Control classes implement the business logic or business rules of the system. Generally each use case is implemented with one or more control classes. Control classes process messages from an interface class and respond to them by sending and receiving messages from the entity classes.

An object-oriented system could be implemented with just these three kinds of classes. But many methodologists include two other kinds of classes.

Persistence Classes

The attributes of the entity classes are generally persistent, meaning they continue to exist beyond when the system is running. The functionality to read and write attributes in a database could be built into the entity classes. But if that functionality put Into separate persistence (or data access) classes, the entity classes are kept Implementation neutral. That can allow the entity classes to be more reusable a major goal of object-oriented design.

System Classes

A final type of object class, the system class, isolates the other objects from operating system-specific functionality. If the system is ported to another operating system, only these classes and perhaps the interface classes have to be changed. Why all these kinds of classes? Structuring the system this way makes the maintenance and enhancement of those classes simpler and easier.

b. Explain the difference between coupling and cohesion.

Answer: Page Number 666 of Text Book

Q.9 a. Describe the systems construction and its implementation.

Answer:

Systems construction is the development installation and testing of system components. Unfortunately, systems development is a common synonym. (We dislike that synonym since it is more frequently used to describe the entire life cycle.) Systems Implementation is the delivery of that system into production (meaning day-to-day operation). Relative to the information systems building blocks, systems construction and implementation address IS building blocks primarily from the system builders' perspective (see the chapter home page). Figure 19-I illustrates the construction and implementation phases. Notice that the trigger for the systems construction phase is the approval of the physical design specifications resulting from the design phase. Given the design specifications, we can construct and test system components for that design. Eventually we will have built the functional system. The functional system can then be implemented or delivered as an operational system.

b. How you can Conduct System Test explain?

Answer:

Now that the software packages and in-house pro grains have been installed and tested, we need to conduct a final system test. All software packages, custom-built programs, and any existing programs that comprise the new system must be tested to ensure that they all work together. This task involves analysts, owners, users, and builders. The systems analyst facilitates the completion of this task. The systems analyst typically communicates testing problems and issues with the project team members. The system owners and system users hold the ultimate authority on whether or not a system is operating correctly. System builders, of various specialties, are involved in the systems testing. For example applications programmers, database programmers and networking specialists may need to resolve problems revealed during systems testing. The primary inputs to this task include the software packages. custom- built pro-grams and any existing programs comprising the new system. The system test is done using the system test data that was developed earlier by the systems analyst. As with previous tests that were performed. the system test may result in required modifications to programs, thus, once again, prompting the return to a construction phase task. This iteration would continue until a successful system test was experienced.

c. Discuss the term Conversion to the new system in system analysis and design.

Answer:

From the old system is a significant milestone after conversion the ownership of the system officially transfers from the analysts and programmers to the end users .The analyst completes this task by carefully carrying out the conversion plan. Recall that the conversion plan includes detailed installation strategies to follow for converting from the existing to the new production information system. This task also involves completing a systems audit. The task involves the systems owners users analysts, designers and builders .The project manager who will oversee the conversion process facilitates it. The system owners provide feedback regarding their experiences with the overall project. They may also provide feedback regarding the new system that has been placed into operation. The system users will provide valuable feedback pertaining to the actual use of the new system. They will be the source of the majority of the feedback used to measure the system's acceptance .The systems analysts, designers, and builders will assess the feedback received from the system owners and users once the system is in operation. In many cases, that feedback may stimulate actions to correct identified short-comings. Regardless, the feedback will be used to help benchmark new systems projects down the road. The key input to this activity is the conversion plan that was created in an earlier Implementation phase task. The principal deliverable is the operational system that is placed into production in the business.

Text Book

Systems Analysis and Design Methods, Jeffrey L Bentley, Seventh Edition, TMH, 2007