

Scope:

This course has been offered to make students proficient in systems specializations. This course takes into account the overview of the system, system analysis, its design, implementation and security aspects of system.

Objectives:

This course has been designed to provide a solid foundation of systems principles and an understanding of how business function is carried on, while heightening students to the issues of an analyst

Unit I

Overview: Introduction - Business Systems concepts, System development life cycle – Life cycle models - Prototyping model, Incremental model, Spiral model, RAD model - Role of systems analyst.

Unit II

System Analysis: System planning and Initial Investigation – Phases of system analysis - Information gathering - Tools of structured analysis - Feasibility study - Cost benefit analysis.

Unit III

System Design: The process and stages of system design - Major development activities - Input and output forms design - File organization and database design - Sequential - Logical and Physical views of data – Normalization – Different forms of normalization.

Unit IV

System Implementation: System testing and quality assurance – COCOMO model of testing -The nature of test data - The test plan-Quality Assurance - Role of the data processing auditor - Implementation and software maintenance hardware/software selection.

Unit V

System Security: Introduction - Definition and Threats to system – Security - Control measures – Recent trends in system security. Disaster/recovery planning: The plan - Ethics in system development - Ethical codes and standards of behavior.

Suggested Readings:

Text book:

1. Awad, E. M. (2010). *System analysis and design* (2nd edition). New Delhi: Galgotia publication.

References:

1. Jr Jain. V.K. (2009). *System Analysis and Design* (1st edition). New Delhi: Dream Tech Press.
2. Senn. (2009). *Analysis and Design of Information System*. New Delhi: McGraw Hill.
3. Alli Baharami. (2010). *Object Oriented Systems Development* (1st edition). New Delhi: McGraw Hill.
4. Kenneth, E. Kendall. (2013). *System Analysis and Design* (9th edition). New Delhi: Pearson publications.



KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established under section 3 of UGC Act 1956)

Coimbatore-641021

DEPARTMENT OF MANAGEMENT

Name: **Dr. S.VIDHYA (Assistant Professor)**

Department: **Management**

Subject Code: **17MBAPS303A**

Semester: **III**

Year: **2017-19 Batch**

Subject: **System Analysis and Design- Lesson Plan**

UNIT - 1			
S. No	Lecture Hours	Contents	References
1	1	Overview: Introduction - Business Systems concepts	R2 – Pg 2-17,18-25
2	1	System development life cycle	T1 – Pg 40-50
3	1	Life cycle models	T1 – Pg 50-56
4	1	Prototyping model	R2 – Pg 215-227
5	1	Incremental model	J1
6	1	Spiral model	W1
7	1	RAD model	W1
8	1	Role of systems analyst	T1 – Pg 67-72
9	1	Case Study	W2
10	1	Recapitulation and Discussion on important questions	-
Total no. of Hours planned for Unit 1			10
UNIT – 2			
1	1	System Analysis	T1 – Pg 90-94
2	1	System planning and Initial investigation	T1 – Pg 94-100
3	1	Phases of system analysis	T1 – Pg 99-103
4	1	Information gathering	T1 – Pg 134-143
5	1	Tools of structured analysis	R2 – Pg 155-182
6	1	Feasibility study	T1 – Pg 183-185, 202-208
7	1	Cost benefit analysis	T1 – Pg 234-248
8	1	Case Study	W2
9	1	Recapitulation and Discussion on important questions	-
Total no. of Hours planned for Unit 2			9
UNIT – 3			
1	1	System Design	T1 – Pg 262-274
2	1	The process and stages of system design	T1 – Pg 262-274
3	1	Major development activities	T1 – Pg 275-279
4	1	Input and output forms design	T1 – Pg 286-312
5	1	File organization and database design	T1 – Pg 322-324

S. No	Lecture Hours	Contents	References
6	1	Sequential – Index Sequential	R2 – Pg 543-560
7	1	Logical & Physical views of data	T1 – Pg 330-344
8	1	Normalization- Different forms of Normalization	R2 – Pg 592-597, 605
9	1	Case Study	J2
10	1	Recapitulation and Discussion on important questions	-
Total number of hours planned for Unit 3			10
UNIT – 4			
1	1	System Implementation	T1 – Pg 360-362
2	1	System Testing and quality assurance	T1 – Pg 362-368, 371-373
3	1	COCOMO Model of Testing	J3
4	1	The nature of test data - The test plan	T1 – Pg 371-373
5	1	Quality Assurance	T1 – Pg 369-371
6	1	Role of the data processing auditor	T1 – Pg 373-374
7	1	Implementation and software maintenance	T1 – Pg 420-436
8	1	Hardware/Software Selection	T1 – Pg 446-456
9	1	Recapitulation and Discussion on important questions	-
Total no. of Hours planned for Unit 4			9
UNIT – 5			
1	1	Introduction to system security, Definition and Threats to system – Security	T1 – Pg 474-481
2	1	Control measures- Recent trends in system security	T1 – Pg 481-486
3	1	Disaster/Recovery planning	T1 – Pg 486-490
4	1	The Plan- Ethics in system development	T1 – Pg 491-493
5	1	Ethical codes and standards of behavior	T1 – Pg 493-495
6	1	Case Study	W2
7	1	Recapitulation and Discussion on important questions	-
8	1	Revision of Previous Year Question Paper	-
9	1	Revision of Previous Year Question Paper	-
10	1	Revision of Previous Year Question Paper	-
Total no. of Hours planned for Unit 5			10

Suggested Readings:

Text Books:

T1. Elias M.Awad, (2010). *System analysis and design*. 2nd Edition, Galgotia publication. New Delhi.

Reference Books:

R2. James A.Senn,(2009). *Analysis and Design of Information Systems*. 2nd Edition. McGraw Hill Education India Pvt. Ltd, New Delhi.

Journals:

- J1.** Journal of Interactive Learning Research
- J2.** Journal of the International Academy for Case Studies
- J3.** University of Southern California

Websites:

- W1.** www.tutorialspoint.com
- W2.** www.gantecusa.com

UNIT-I

SYLLABUS

Overview: Introduction - Business Systems concepts, System development life cycle – Life cycle models - Prototyping model, Incremental model, Spiral model, RAD model - Role of systems analyst.

INTRODUCTION TO SYSTEM ANALYSIS AND DESIGN

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design (SAD) mainly deals with the software development activities.

System

The term system is derived from the Greek word *systema*, which means an organized relationship among functioning units' or components. A system exists because it is designed to achieve one or more objectives. A system is an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective.

SYSTEM DESIGN

It's the process of planning a new business system or one to replace or complement an existing system. Before it we must to understand the old system and how it's used.

SYSTEM ANALYSIS

It's the process of gathering and interpreting facts, diagnosing problems, and using the information to recommend improvements to the system.

DEFINING A SYSTEM

A collection of components that work together to realize some objectives forms a system. Basically there are three major components in every system, namely input, processing and output.

Fig. 1.1: Basic System Components

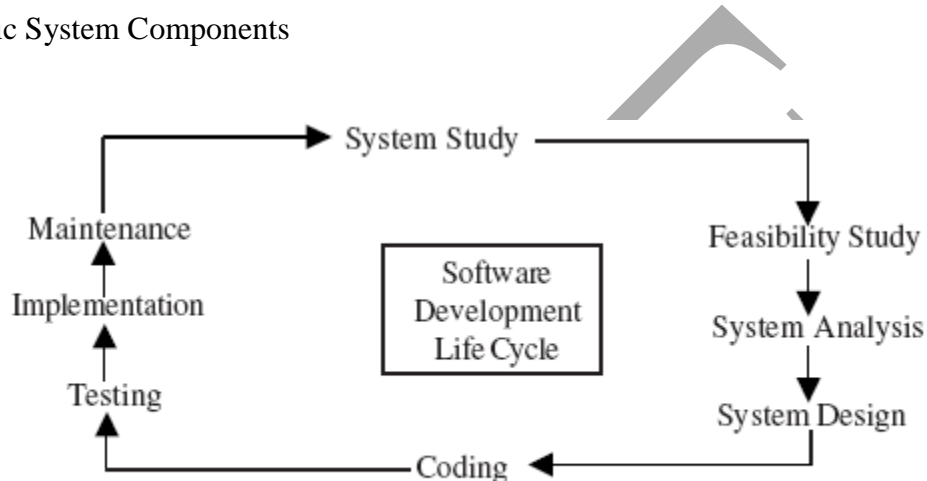


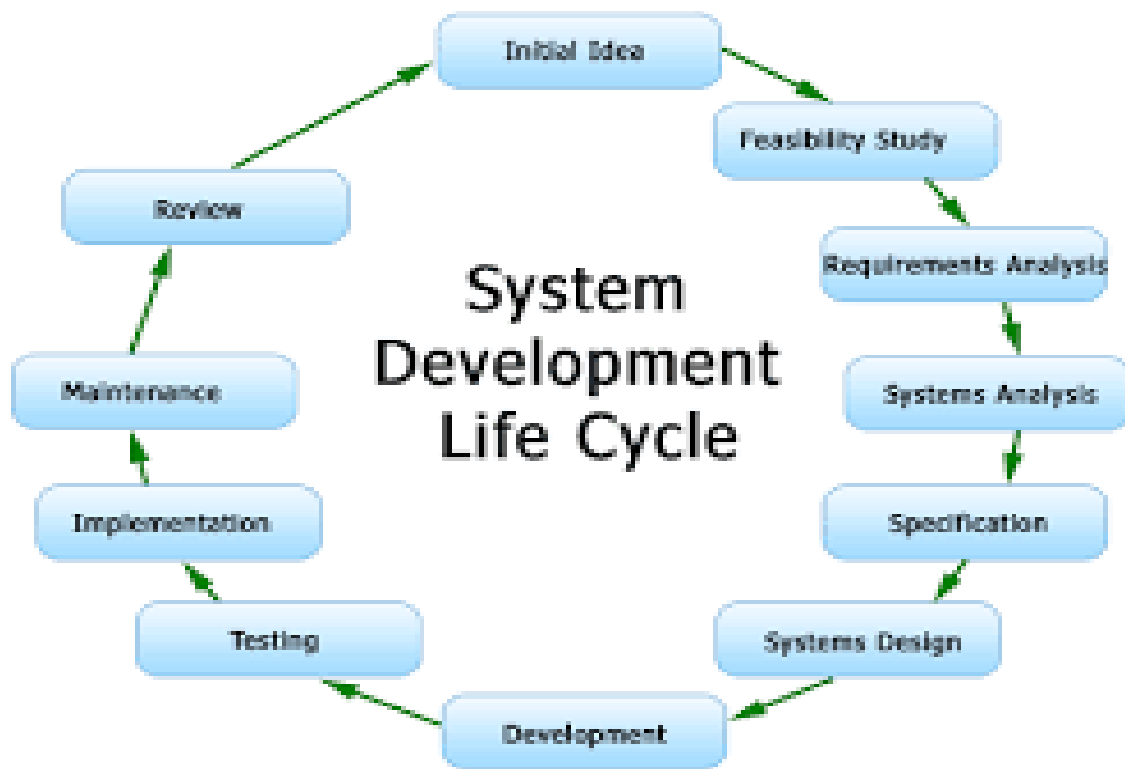
Fig. 1.2: Phases of System Development Life Cycle

Computer Applications

In a system the different components are connected with each other and they are interdependent. For example, human body represents a complete natural system. We are also bound by many natural systems such as political system, economic system, educational system and so forth. The objective of the system demands that some output is produced as a result of processing the suitable inputs. A well-designed system also includes an additional element referred to as 'control' that provides a feedback to achieve desired objectives of the system.

SYSTEM LIFE CYCLE

System life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system. System development life cycle means combination of various activities.



In other words we can say that various activities put together are referred as system development life cycle. In the System Analysis and Design terminology, the system development life cycle also means software development life cycle.

Following are the different phases of system development life cycle:

- Preliminary study
- Feasibility study
- Detailed system study
- System analysis
- System design
- Coding
- Testing
- Implementation
- Maintenance

PHASES OF SYSTEM DEVELOPMENT LIFE CYCLE

The different phases and related activities of system development life cycle.

(a) Preliminary System Study

Preliminary system study is the first stage of system development life cycle. This is a brief investigation of the system under consideration and gives a clear picture of what actually the physical system is? In practice, the initial system study involves the preparation of a 'System Proposal' which lists the Problem Definition, Objectives of the Study, Terms of reference for Study, Constraints, and Expected benefits of the new system, etc. in the light of the user requirements.

The system proposal is prepared by the System Analyst (who studies the system) and places it before the user management. The management may accept the proposal and the cycle proceeds to the next stage. The management may also reject the proposal or request some modifications in the proposal. In summary, we would say that system study phase passes through the following steps:

- problem identification and project initiation
- background analysis
- inference or findings (system proposal)

(b) Feasibility Study

In case the system proposal is acceptable to the management, the next phase is to examine the feasibility of the system. The feasibility study is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and of course, the cost effectiveness. These are categorized as technical, operational, economic and schedule feasibility. The main goal of feasibility study is not to solve the problem but to achieve the scope. In the process of feasibility study, the cost and benefits are estimated with greater accuracy to find the Return on Investment(ROI). This also defines the resources needed to complete the detailed investigation.

The result is a feasibility report submitted to the management. This may be accepted or accepted with modifications or rejected. The system cycle proceeds only if the management accepts it.

(c) Detailed System Study

The detailed investigation of the system is carried out in accordance with the objectives of the proposed system. This involves detailed study of various operations performed by a system and their relationships within and outside the system. During this process, data are collected on the available files, decision points and transactions handled by the present system. Interviews, on-site observation and questionnaire are the tools used for detailed system study. Using the following steps it becomes easy to draw the exact boundary of the new system under consideration:

- Keeping in view the problems and new requirements
- Workout the pros and cons including new areas of the system

All the data and the findings must be documented in the form of detailed data flow diagrams (DFDs), data dictionary, logical data structures and miniature specification.

The main points to be discussed in this stage are:

- Specification of what the new system is to accomplish based on the user requirements.
- Functional hierarchy showing the functions to be performed by the new system and their relationship with each other.
- Functional network, which are similar to function hierarchy but they highlight the functions which are common to more than one procedure.
- List of attributes of the entities – these are the data items which need to be held about each entity (record)

(d) System Analysis

Systems analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning.

This involves studying the business processes, gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weaknesses of the system so as to achieve the organizational goals. System Analysis also includes subdividing of complex process involving the entire system, identification of data store and manual processes.

The major objectives of systems analysis are to find answers for each business process: What is being done How is it being done, who is doing it, When is he doing it, Why is it being done and How can it be improved? It is more of a thinking process and involves the creative skills of the System Analyst.

It attempts to give birth to a new efficient system that satisfies the current needs of the user and has scope for future growth within the organizational constraints. The result of this process is a logical system design. Systems analysis is an iterative process that continues until a preferred and acceptable solution emerges.

(e) System Design

Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. It is the most crucial phase in the developments of a system. The logical system design arrived at as a result of systems analysis is converted into physical system design. Normally, the design proceeds in two stages:

- Preliminary or General Design
- Structured or Detailed Design

Preliminary or General Design: In the preliminary or general design, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated. If the project is still considered to be feasible, we move to the detailed design stage.

Structured or Detailed Design: In the detailed design stage, computer oriented work begins in earnest. At this stage, the design of the system becomes more structured.

Structure design is a blue print of a computer system solution to a given problem having the same components and inter-relationships among the same components as the original problem. Input, output, databases, forms, codification schemes and processing specifications are drawn up in detail.

In the design stage, the programming language and the hardware and software platform in which the new system will run are also decided. There are several tools and techniques used for describing the system design of the system. These tools and techniques are:

- Flowchart
- Data flow diagram (DFD)
- Data dictionary
- Structured English
- Decision table
- Decision tree

Each of the above tools for designing will be discussed in detailed in the next lesson.

The system design involves:

- i. Defining precisely the required system output
- ii. Determining the data requirement for producing the output
- iii. Determining the medium and format of files and databases
- iv. Devising processing methods and use of software to produce output
- v. Determine the methods of data capture and data input
- vi. Designing Input forms
- vii. Designing Codification Schemes
- viii. Detailed manual procedures
- ix. Documenting the Design

(f) Coding

The system design needs to be implemented to make it a workable system. This demands the coding of design into computer understandable language, i.e., programming language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer to as programs. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future changes, if required.

(g) Testing

Before actually implementing the new system into operation, a test run of the system is done for removing the bugs, if any. It is an important phase of a successful system. After codifying the whole programs of the system, a test plan should be developed and run on a given set of test data. The output of the test run should match the expected results. Sometimes, system testing is considered a part of implementation process.

Using the test data following test run are carried out:

- Program test
- System test

Program test: When the programs have been coded, compiled and brought to working conditions, they must be individually tested with the prepared test data. Any undesirable happening must be noted and debugged (error corrections)

System Test: After carrying out the program test for each of the programs of the system and errors removed, then system test is done. At this stage the test is done on actual data. The complete system is executed on the actual data.

At each stage of the execution, the results or output of the system is analysed. During the result analysis, it may be found that the outputs are not matching the expected output of the system. In such case, the errors in the particular programs are identified and are fixed and further tested for the expected output. When it is ensured that the system is running error-free, the users are called with their own actual data so that the system could be shown running as per their requirements.

(h) Implementation

After having the user acceptance of the new system developed, the implementation phase begins. Implementation is the stage of a project during which theory is turned into practice. The major steps involved in this phase are:

- Acquisition and Installation of Hardware and Software
- Conversion
- User Training
- Documentation

The hardware and the relevant software required for running the system must be made fully operational before implementation.

The conversion is also one of the most critical and expensive activities in the system development life cycle. The data from the old system needs to be converted to operate in the new format of the new system. The database needs to be setup with security and recovery procedures fully defined. During this phase, all the programs of the system are loaded onto the user's computer.

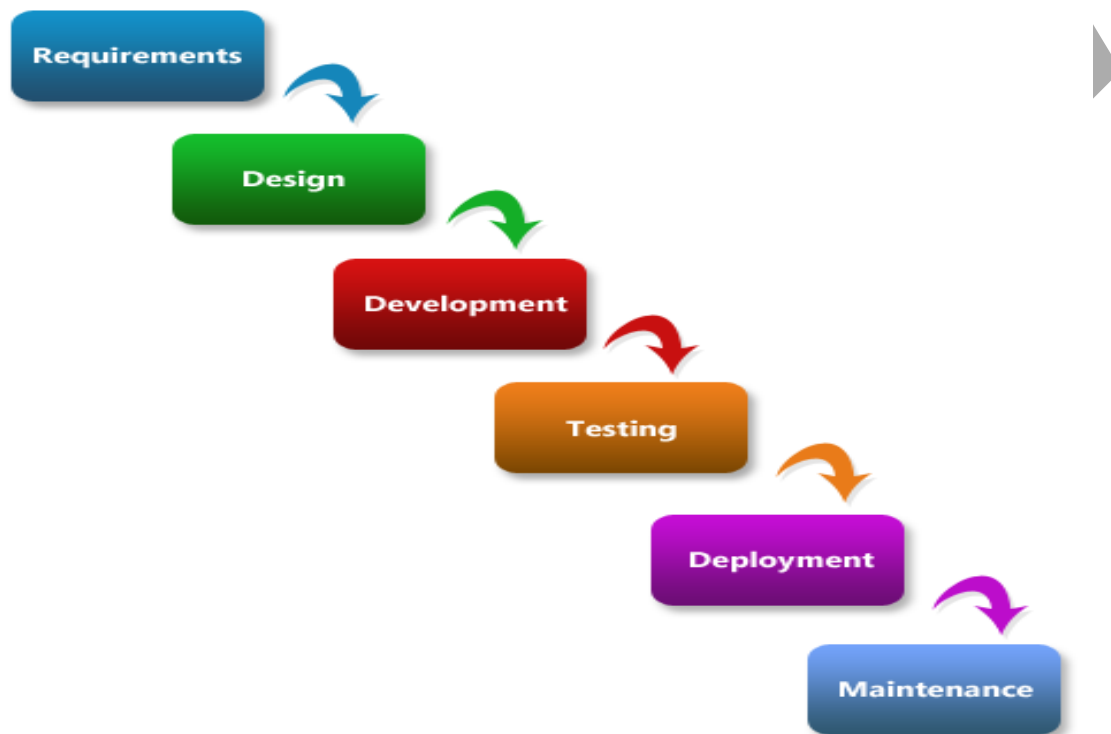
Waterfall model

The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance.

The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.

The first known presentation describing use of similar phases in software engineering was held by Herbert D. Benington at Symposium on advanced programming methods for digital computers on 29 June 1956. This presentation was about the development of software for SAGE.

Waterfall Methodology



In 1983 the paper was republished with a foreword by Benington pointing out that the process was not in fact performed in a strict top-down fashion, but depended on a prototype.

In Royce's original waterfall model, the following phases are followed in order:

1. Requirements specification resulting in the product requirements document
2. Design resulting in the Software architecture

3. Construction (implementation or coding) resulting in the actual software
4. Integration
5. Testing and debugging
6. Installation
7. Maintenance

Thus the waterfall model maintains that one should move to a phase only when its preceding phase is reviewed and verified. Various modified waterfall models (including Royce's final model), however, can include slight or major variations on this process

Prototyping approach, also known as evolutionary approach, came to picture because of failures that occurred in the final version of the software application developed using the waterfall approach.

The failure generally occurs because of the changes in the requirement of the proposed system or because of the gap in understanding the customer requirement by the development team. A gap in the first version of the developed application, inevitably leads to the need for redoing the application. To overcome these limitations, the concept of prototyping was introduced.

Prototyping Approach

A prototype is the sample implementation of the system that shows limited and main functional capabilities of the proposed system. After a prototype is built, it is delivered to the customer for the evaluation. The prototype helps the customer determine how the feature will function in the final software. The customer provides suggestion and improvements on the prototype. The development team implements the suggestion in the new prototype, which is again evaluated by the customer. The process continues until the customer and the development team understands the exact requirement of the proposed system. When the final prototype is developed, the requirement is considered to be frozen

The prototyping approach is used in the requirement gathering and in the analysis phase to capture the exact requirement of the proposed system. After the requirements are frozen, the remaining phases of the development process needs to be executed to complete the development of the software system.

An e-commerce website, such as shopping site is an example where you can implement the prototyping approach. You can develop the prototype of the various web pages of the shopping site such as catalogue page, product order page etc., and present it to the customer for approval. If the customer approves the prototype of the site, requirements are states again and the design of the web site is initiated. If the customer does not approve the web site, the development team revisits the prototype and resubmits it to the customer for approval. This process continues until the prototype is approved.

Prototypes are of two types:

(i) Throwaway prototypes: Prototypes that are eventually discarded rather than becoming a part of the finally delivered software. Examples of throwaway prototypes include screen mock-ups and story boards.

(ii) Evolutionary Prototypes: prototypes that evolve into the final system through iterative incorporation of user feedback.

Although prototyping is a very useful technique to obtain accurate requirements of the system and to speed up the development process, it has some disadvantage associated with it. Some disadvantages of Prototyping are:

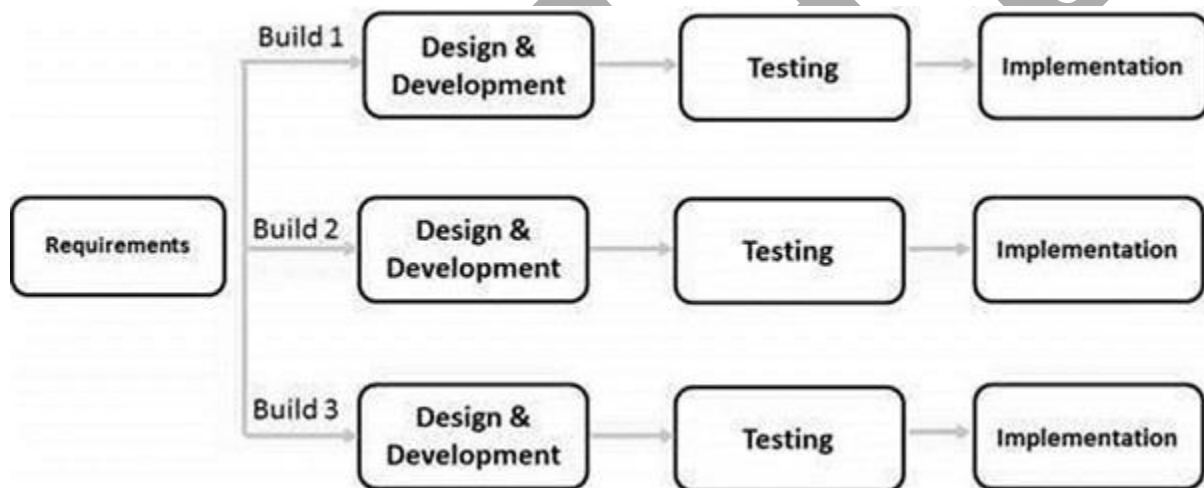
It gives client a false impression that a few minor changes to the prototype will give them the required system. It may comprise on the overall quality of the software in the rush to develop the prototype.

For example, the developer may use insufficient algorithm or inappropriate programming languages for developing the prototype quickly and the same may find place in the final application thus leading to insufficient code running in the final application.

Incremental Model

The incremental Model is an evolution of the waterfall model, where the waterfall model is incrementally applied.

The series of releases is referred to as “increments”, with each increment providing more functionality to the customers. After the first increment, a core product is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly. This process continues, with increments being delivered until the complete product is delivered. The incremental philosophy is also used in the agile process model.

**Advantages**

After every iteration, regression testing should be conducted. During this testing, faulty elements of the software can be quickly identified because few changes are made within any single iteration.

1. It is generally easier to test and debug than other methods of software development because relatively smaller changes are made during each iteration. This allows for more targeted and rigorous testing of each element within the overall product.

2. Customer can respond to features and review the product for any needful changes.
3. Initial product delivery is faster and costs lower.

Disadvantages

Resulting cost may exceed the cost of the organization.

1. As additional functionality is added to the product, problems may arise related to system architecture which was not evident in earlier prototypes.

Spiral Model

The spiral model is a risk-driven process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. Spiral model is a combination of iterative development process model and sequential linear development model i.e. waterfall model with very high emphasis on risk analysis. It allows for incremental releases of the product, or incremental refinement through each iteration around the spiral.

Spiral Model design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

- **Identification:** This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

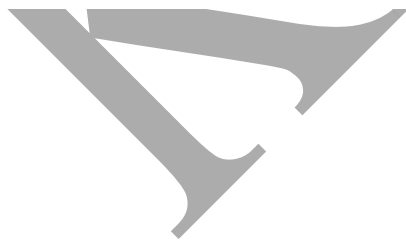
This also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral the product is deployed in the identified market.

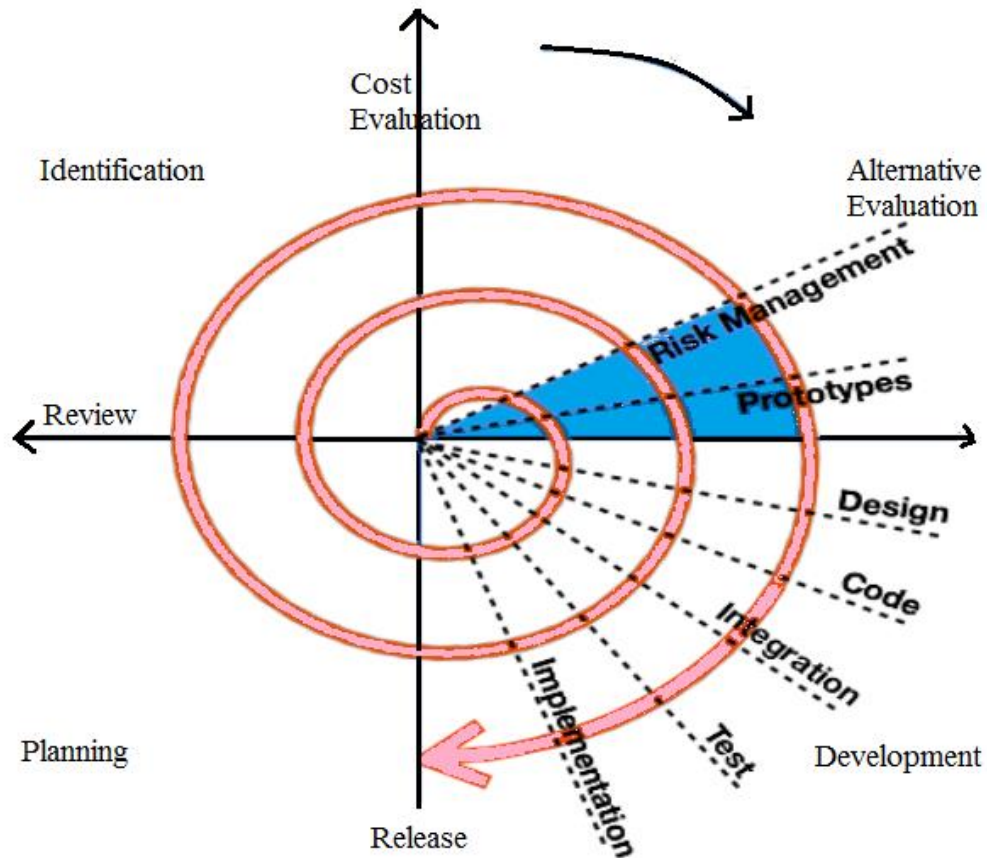
- **Design:** Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and final design in the subsequent spirals.
- **Construct or Build:** Construct phase refers to production of the actual software product at every spiral. In the baseline spiral when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to customer for feedback.

- **Evaluation and Risk Analysis:** Risk Analysis includes identifying, estimating, and monitoring technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

Following is a diagrammatic representation of spiral model listing the activities in each phase:





Based on the customer evaluation, software development process enters into the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Spiral Model Application

Spiral Model is very widely used in the software industry as it is in synch with the natural development process of any product i.e. learning with maturity and also involves minimum risk for the customer as well as the development firms. Following are the typical uses of Spiral model:

- When costs there are a budget constraint and risk evaluation is important.
- For medium to high-risk projects.

- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which are usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.



Spiral Model Pros and Cons

The advantage of spiral lifecycle model is that it allows for elements of the product to be added in when they become available or known. This assures that there is no conflict with previous requirements and design.

This method is consistent with approaches that have multiple software builds and releases and allows for making an orderly transition to a maintenance activity. Another positive aspect is that the spiral model forces early user involvement in the system development effort.

On the other side, it takes very strict management to complete such products and there is a risk of running the spiral in indefinite loop. So the discipline of change and the extent of taking change requests is very important to develop and deploy the product successfully.

The following table lists out the pros and cons of Spiral SDLC Model:

Pros	Cons
<ul style="list-style-type: none">• Changing requirements can be accommodated.• Allows for extensive use of prototypes• Requirements can be captured more accurately.• Users see the system early.• Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.	<ul style="list-style-type: none">• Management is more complex.• End of project may not be known early.• Not suitable for small or low risk projects and could be expensive for small projects.• Process is complex• Spiral may go indefinitely.• Large number of intermediate stages requires excessive documentation.

Rapid Application Development

The RAD (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

What is RAD?

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In RAD model the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

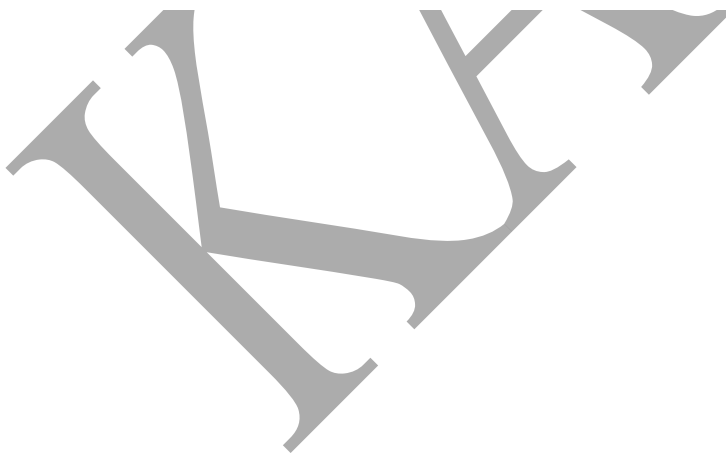
RAD Model Design

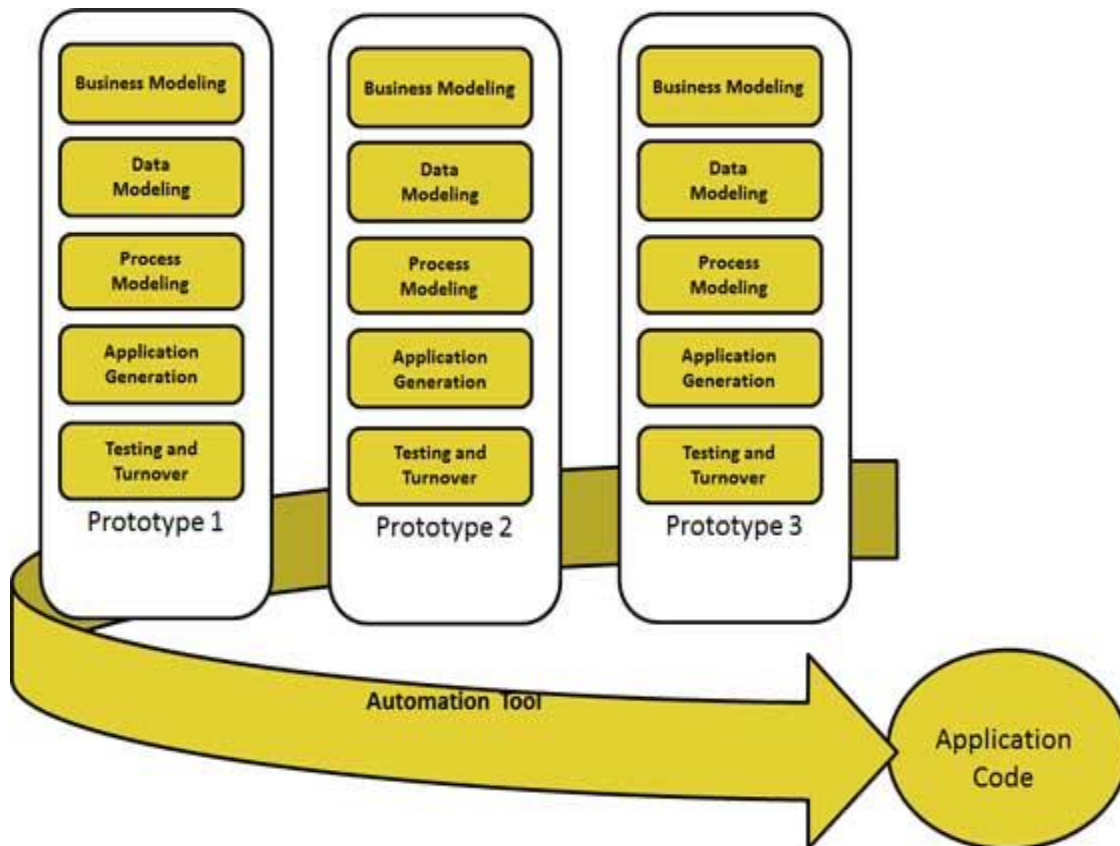
RAD model distributes the analysis, design, build, and test phases into a series of short, iterative development cycles. Following are the phases of RAD Model:

- **Business Modeling:** The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.
- **Data Modeling:** The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.

- **Process Modeling:** The data object sets defined in the Data Modeling phase is converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding, deleting, retrieving or modifying a data object are given.
- **Application Generation:** The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.
- **Testing and Turnover:** The overall testing time is reduced in RAD model as the prototypes are independently tested every iteration. However the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

Following image illustrates the RAD Model:





RAD Model Vs Traditional SDLC

The traditional SDLC follows a rigid process models with high emphasis on requirement analysis and gathering before the coding starts. It puts a pressure on the customer to sign off the requirements before the project starts and the customer doesn't get the feel of the product as there is no working build available for a long time.

The customer may need some changes after he actually gets to see the software, however the change process is quite rigid and it may not be feasible to incorporate major changes in the product in traditional SDLC.

RAD model focuses on iterative and incremental delivery of working models to the customer. This results in rapid delivery to the customer and customer involvement during the complete development cycle of product reducing the risk of non conformance with the actual user requirements.

RAD Model Application

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail. Following are the typical scenarios where RAD can be used:

- RAD should be used only when a system can be modularized to be delivered in incremental manner.
- It should be used if there's high availability of designers for modeling.
- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the course of the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

RAD Model Pros and Cons

RAD model enables rapid delivery as it reduces the overall development time due to reusability of the components and parallel development.

RAD works well only if high skilled engineers are available and the customer is also committed to achieve the targeted prototype in the given time frame. If there is commitment lacking on either side the model may fail.

Following table lists out the pros and cons of RAD Model:

Pros	Cons
<ul style="list-style-type: none">• Changing requirements can be accommodated.• Progress can be measured.• Iteration time can be short with use of powerful RAD tools.• Productivity with fewer people in short time.• Reduced development time.• Increases reusability of components• Quick initial reviews occur• Encourages customer feedback• Integration from very beginning solves a lot of integration issues.	<ul style="list-style-type: none">• Dependency on technically strong team members for identifying business requirements.• Only system that can be modularized can be built using RAD.• Requires highly skilled developers/designers.• High dependency on modeling skills.• Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.• Management complexity is more.• Suitable for systems that are component based and scalable.• Requires user involvement throughout the life cycle.• Suitable for project requiring shorter development times.

Systems Analyst

A **systems analyst** is an IT professional who specializes in analyzing, designing and implementing information systems. System analysts assess the suitability of information systems in terms of their intended outcomes and liaise with end users, software vendors and programmers in order to achieve these outcomes.

Although they may be familiar with a variety of programming languages, operating systems, and computer hardware platforms, they do not normally involve themselves in the actual hardware or software development. They may be responsible for developing cost analysis, design considerations, staff impact amelioration, and implementation time-lines.

A systems analyst may:

- Identify, understand and plan for organizational and human impacts of planned systems, and ensure that new technical requirements are properly integrated with existing processes and skill sets.
- Plan a system flow from the ground up.
- Interact with internal users and customers to learn and document requirements that are then used to produce business requirements documents.
- Write technical requirements from a critical phase.
- Interact with designers to understand software limitations.
- Help programmers during system development, ex: provide use cases, flowcharts or even database design.
- Perform system testing.
- Deploy the completed system.
- Document requirements or contribute to user manuals.
- Whenever a development process is conducted, the system analyst is responsible for designing components and providing that information to the developer.

The system development life cycle (SDLC) is the traditional system development method that organizations use for large-scale IT Projects. The SDLC is a structured framework that consists of sequential processes by which information systems are developed.

1. System Investigation
2. System Analysis
3. System Design
4. Programming and Testing
5. Implementation
6. Operation and Maintenance

Once a development project has the necessary approvals from all participants, the systems analysis stage begins. System analysis is the examination of the business problem that organizations plan to solve with an information system. The main purpose of the systems analysis stage is to gather information about the existing system in order to determine the requirements for an enhanced system or a new system. The end product of this stage, known as the deliverable, is a set of system requirements.

Perhaps the most difficult task in system analysis is identifying the specific requirements that the system must satisfy. These requirements often are called user requirements because users provide them. When the system developers have accumulated the user requirements for the new system, they proceed to the system design stage.

A computer systems analyst is an occupation in the field of information technology. A computer systems analyst works to solve problems related to computer technology. Many analysts set up new computer systems, both the hardware and software, add new software applications to increase computer productivity. Others act as system developers or system architects, but most analysts specialize in a specific type of system such as business systems, accounting systems, financial systems, or scientific systems.

Part A (ONE Mark)

Multiple Choice Questions

Online Examination

Part B (2 Marks)

1. Define system & list its various characteristics.
2. Discuss the primary characteristics of open systems. In what way is a system entropic?
3. How important is the informal information system in system analysis?
4. Explain the various categories of information relevant to decision making in business?
5. Discuss the concepts of MIS & DSS. How are they related? How do they differ?
6. Distinguish between initial investigation & feasibility study. In what way are they related?
7. A number of activities are carried out under implementation. Elaborate.
8. What are the considerations in deciding on a candidate system and why are they important?
9. Explain briefly the levels of structuring work units in system development.
10. Discuss top-down approach to system planning.

Part C (5 Marks)

1. Write in brief about various elements of a system.
2. Distinguish between: Interaction & interdependence Physical & Abstract Systems Open & Closed systems Schematic & static systems models.
3. Describe the role and tasks of system analyst.
4. What is system development life cycle .How does it relate to system analysis?
5. List all the phases of system development life cycle in order & explain them in detail.
6. Explain what is prototyping, its basic steps and benefits?
7. Discuss Spiral Model and explain the pros and cons of the model?

KARPAGAM ACADEMY OF HIGHER EDUCATION**Department of Management****Unit 1- System Analysis and Design -Multiple Choice Questions- Each Question Carry ONE Mark**

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	Data entry are	personnel	production	occupational	management	personnel
2	The term system is derived from the greek word---	systema	system	scientist	sub-systems	systema
3refers to the manner in which each component functions	interdependence	interacton	integration	information	interaction
4 means that parts of the organization or computer system depend on one another	interdependence	interaction	integration	information	interdependence
5implies structure and order	production	organization	sales	objective	organization
6	How much points in the key elements of a system	four	ten	six	five	six
7refers to the holism of systems	integration	interaction	interdependence	information	integration
8is a major objective of a system	action	Inputs / output	Components	process	Input/Output
9elements guides the system	feedback	interface	control	environment	control
10in a dynamic system is achieved by a feedback	control	interface	environment	none	control
11is an art and science	information gathering	information describing	information structure	information need	information gathering
12is a face-to-face interpersonnel role situation in which a person called the interviewer	self-interviewer	interviewer	personnel interviewer	customer	interviewer
13is a way of thinking about organisation and their problems	system concept	system	system develop	none	system
14	User training was	poor	best	fast	medium	poor
15	The user was not directly involved in the crucial phases of	system approach	system management	system development	system design	system development

16	Users are changed their	requirements	departments	feedback	environment	requirements
17	In organization requirements (or)environmental factors also called for system	procedure	enhancements	analyst	programmer	enhancements
18	Like any system,there is an aging process that requires periodic of hardware and software	maintenance	debug	procedure	none of the above	maintenance
19	The most creative and challenging phase of the system life cycle is	system design	system analysis	system cycle	all the above	system design
20	During analysis ,data are collected on the available	papers	documents	tool	files	files
21	Is a detailed study of the various operationa performed by a system	feasibility	strategy	analysis	design	analysis
22	Is a careflly worded that led to analysis	statement of the problem	methods of the problem	management of the problem	none	statement of the problem
23	In specific recommendation regarding the candidate system ,including assignments	formal	personnel	interpersonnel	semi-personnel	personnel
24	In sources of system ideas how much type based?	two	one	five	ten	two
25	In finished goods also,prompt to initiate an investigation	labour management	budget management	bottom management	top management	top management
26	A word of caution regarding activities?	prototype cycle	lifecycle	candidate cycle	target	lifecycle
27	How much types are involved in SDLC?	six	ten	five	three	six
28	Human problem solving?	behaviour	attitudes	character	descripline	behaviour
29	A approach to system planning	data	down-top	top-down	right-down	top-down
30	Need identification and analysis are concerned	user	output	inputs	all the above	user

31	Planning information systems in business has become important during the past decade	increasingly	decreasingly	medium	none	increasingly
32	Is the strategic planning for MIS	george	ephraim R.mclean	harvey poppel	A.striner	ephraim R.mclean
33	Must support organizational MIS objectives	system planning	system control	system development	all the above	system development
34	Is a one way of handling standard procedures.	initial investigation	submitted	data support	user request	initial investigation
35	Planning is an orderly approach that determinants the basic objectives for the user to achieve,their policies	strategic	technology	development	analysis	strateegic
36	A is a working tool and an excellent way to keep track of the data collected for a system	structure chart	structure pie chart	structure triangullar	structure rectangullar	structure chart
37	In information originate are gathered from principal sources	three	four	five	two	two
38	A observation occurs in a setting such as the empolyees place work	natural	physical	direct	indirect	natural
39	An observation takes place when the respondent knows he\she is being observed	contrived	natural	obtrusive	direct	obtrusive
40	Is a set of techniques and graphical tools that allow the analyst to develop a new kind of system	structured analysis	structured graphical	structured tools	structured systems	structured tools
41	The focus on the tools listed earlier in data dictionary	structured graphical	structured analysis	structured tools	structured systems	structured graphical
42	The objective is to build a new document ,called	system planning	system control	system development	system specifications	system development
43	In structured analysis ,focus on functions rather than the physical are	data dictionary	decision trees	decision tables	none of the above	data dictionary

44	Expand DFD?	define focus on data	derive fuctions in data	data flow diagram	data force design	data force design
45	In DFD ,how much symbols are involved?	four	two	five	none	four
46	An identifies data flow in data in motion	circle	arrow	open rectangle box	none	arrow
47	A circle or a represents a process that transforms incoming data flow(s) into outgoing data flow(s)	square	circle	arrow	bubble	circle
48	An is a data storedata at rest,or a temporary repository of data	open rectangle	open circle	open arrow	open bubble	open rectangle
49	A DFD,also known as a?	bubble	bubble chart	bubble circle	bubble rectangle	bubble chart
50	A is a structured repository of data about data2	data focus	data diagram	data dictionary	data design	data dictionary
51	Has a limited knowledge of computers ,although it has several applications on the computer	management	checking and savings	none	objective	checking and savings
52	Is atha most frequently used method for evaluting the effectiveness of a candidate system	economic feasibility	economic analysis	economic considerations	Cost analysis	economic considerations
53	Centers around the existing computer system hardware\software,etc?	technical feasibility	behavioral feasibility	economic feasibility	C & B analysis	behavioral feasibility
54	Many feasibility studies are disillusioning for both users and?	analysis	analytical	analysts	none of the above	analysts
55	How much steps in feasibility analysis?	four	ten	eight	six	four
56	Data analysis is a prerequisite to analysis?	cost \benefit	cost\effectiveness	cost\analysis	none	cost\effectiveness
57 Relate to the actual purchase or lease of the computer and peripherals?	hardware costs	software costs	personnel costs	system costs	software costs
58 Include EDP staff salaries and benefits in vacation time	personnel costs	hardware costs	software costs	system costs	personnel costs

59 Are expenses incurred in the preparation of the physical site ?	operating costs	personnel costs	facility costs	none	operating costs
60 Include all costs associated with the day-to-day operation systems	facility costs	operating costs	system costs	Benefit analysis	facility costs

UNIT-II

SYLLABUS

System Analysis: System planning and Initial Investigation – Phases of system analysis - Information gathering - Tools of structured analysis - Feasibility study - Cost benefit analysis.

Systems analysis is the study of sets of interacting entities, including computer systems analysis. According to the Merriam-Webster dictionary, systems analysis is "the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way". Analysis and synthesis, as scientific methods, always go hand in hand; they complement one another. Every synthesis is built upon the results of a preceding analysis, and every analysis requires a subsequent synthesis in order to verify and correct its results.

Systems Planning and the Initial Investigation

Initial Investigation

- This is the first phase of SDLC and is known as identification of need.
- This is a user's request to change, improve or enhance an existing system.
- The objective is to determine whether the request is valid or feasible
- The user request identifies the need for change and authorizes the initial investigation.

User's Request Form

- User assigned title of work requested.
- Nature of work requested (problem definition)
- Date request was submitted
- Date job should be completed

- Purpose of job requested
- Expected benefits
- Input/output description
- Requester's signature title, department, and phone number.
- Signature, title and department of person approving the request.

Needs Identification

- The success of a system depends largely on how accurately a problem is defined, thoroughly investigated and properly carried out through the choice of solution.
- It is concerned with what the user needs rather than what he/she wants.

Phases of system analysis

The analysis phase involves gathering requirements for the system. At this stage, business needs are studied with the intention of making business processes more efficient. The system analysis phase focuses on what the system will do in an effort that views all stakeholders, as viable sources of information.

In the analysis phase, a significant amount of time is spent talking with stakeholders and reviewing the stakeholder's input. Common stakeholders for IT projects are:

- Architecture office
- Testing & certification office
- Records management team
- Application support group

Once stakeholders have been recognized, the gathering and analysis of the requirements can begin. Requirement gathering must be related to business needs or opportunities. Requirement analysis involves capturing requirements and analyzing requirements. Capturing requirements is communicating with stakeholders to agree on what the requirements are. Analyzing requirements is using standard tools to produce a baseline of the requirements. Once the stakeholders concur on the requirements, the baseline is created and becomes the formal requirement source.

Within this analysis phase, the analyst is discovering and fact finding. Along with meeting with stakeholders, the analyst must meet with end users to understand what the user's needs are and to learn about problems that affect the current system in order to assist with designing a new and more efficient system. There are several activities that must occur within the analysis phase:

- Gather Information
- Define the new system's requirements
- Build prototypes for the new system
- Prioritize requirements
- Evaluate alternatives
- Meet with management to discuss new options

Information gathering / Strategies used by the Users

Kitchen Sink Strategy- user throws everything into the requirement definition, overstatement of needs such as an overabundance of reports .This approach usually reflects the user's lack of experience in the area

Smoking Strategy - It sets up a smoke screen by requesting several system features when only one or two are needed. Requests have to be reduced to one that is realistic, manageable and achievable

Same Thing Strategy -This strategy indicates the user's laziness, lack of knowledge or both. "Give me the same thing but in a better format through the computer" is a typical statement. The analyst has little chance of succeeding because only the user can fully discover the real needs and problems.

Human's limitations- Humans as information processors .Human bias in data selection and use .Human problem solving behavior. Strategies for Determining Information Requirements,

- Asking
- Getting Information from the existing information system
- Prototyping

1. Asking

This strategy obtains information from users by simply asking them about their requirements.

The three methods of asking are:

- Questions (open-ended or closed)
- Brainstorming
- Group Consensus (Delphi Technique)

2. Getting Information from the existing information system

(i) Data Analysis

Determining Information from existing system. It simply asks the user what information is currently received and what other information is required. Ideal for Structured Decisions.

(ii) Decision Analysis

In this problem is broken down into parts, so that user can focus separately on the critical issues. It is used for Unstructured Decisions.

(iii) Problem Definition and Project Initiation

The problem must be stated clearly, understood, and agreed upon by the user and the analyst.

(iv) Background Analysis

Once the project is initiated, the analyst begins to learn about the setting, the existing System and the physical processes related to the revised system.

(v)Fact Finding

After obtaining the background knowledge, the analyst begins to collect data on the existing system's outputs, inputs and costs, The tools used in data collection are:

- Review of written documents
- On site observations
- Interviews
- Questionnaires

What is Structured Analysis?

Structured Analysis is a development method that allows the analyst to understand the system and its activities in a logical way.

It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understandable by user.

It has following attributes –

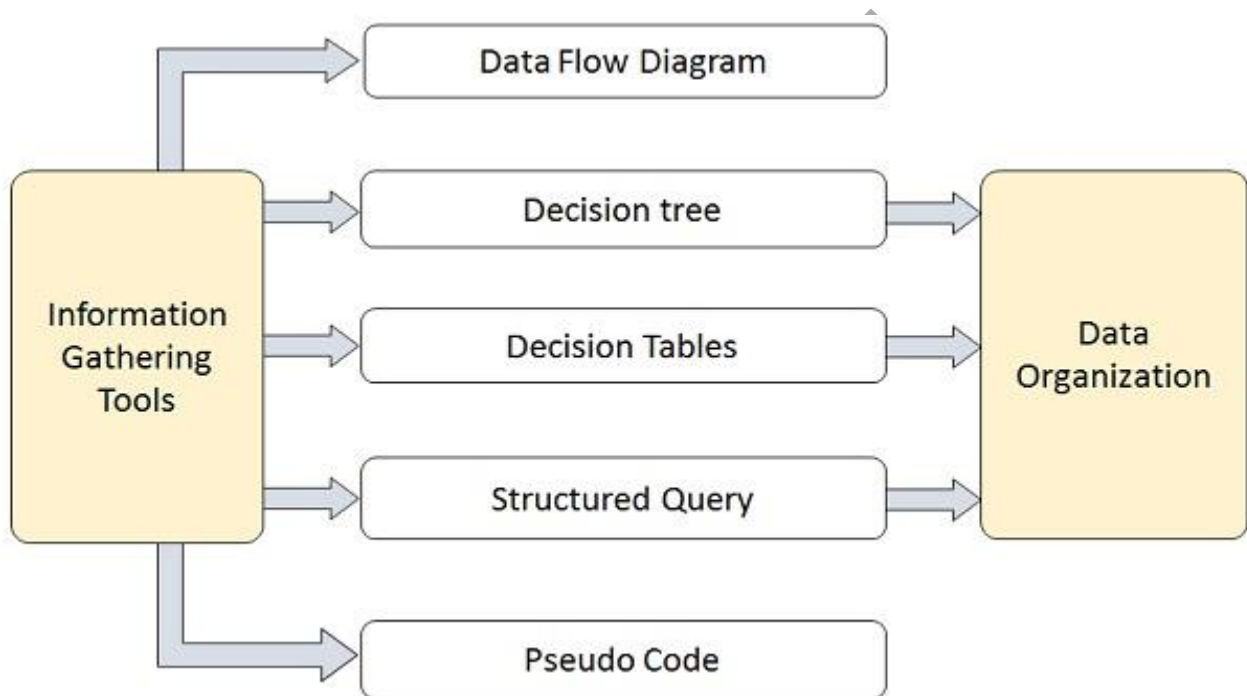
- It is graphic which specifies the presentation of application.
- It divides the processes so that it gives a clear picture of system flow.
- It is logical rather than physical i.e., the elements of system do not depend on vendor or hardware.
- It is an approach that works from high-level overviews to lower-level details.

Structured Analysis Tools

During Structured Analysis, various tools and techniques are used for system development. They are –

- Data Flow Diagrams
- Data Dictionary
- Decision Trees

- Decision Tables
- Structured English
- Pseudocode



Data Flow Diagrams (DFD) or Bubble Chart

It is a technique developed by Larry Constantine to express the requirements of system in a graphical form.


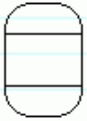


- It shows the flow of data between various functions of system and specifies how the current system is implemented.
- It is an initial stage of design phase that functionally divides the requirement specifications down to the lowest level of detail.
- Its graphical nature makes it a good communication tool between user and analyst or analyst and system designer.

- It gives an overview of what data a system processes, what transformations are performed, what data are stored, what results are produced and where they flow.

Basic Elements of DFD

DFD is easy to understand and quite effective when the required design is not clear and the user want a notational language for communication. However, it requires a large number of iterations for obtaining the most accurate and complete solution.

The following table shows the symbols used in designing a DFD and their significance –

	<i>External entity</i>	Source or destination of data that is external to the system
	<i>Process</i>	Manual or computer process that changes data. In the following text a circle is used to indicate a process
	<i>Data flow</i>	Data transfer in the direction indicated by the arrow. Each arrow should be labeled to indicate what data is being transferred
	<i>Data store</i>	Manual or computer storage of data

Types of DFD

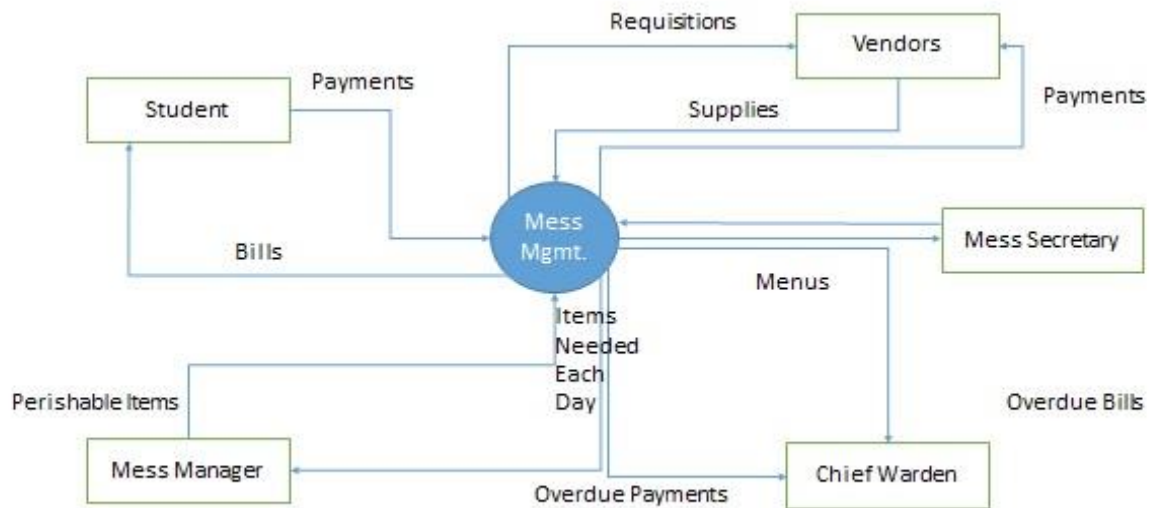
DFDs are of two types: Physical DFD and Logical DFD. The following table lists the points that differentiate a physical DFD from a logical DFD.

Physical DFD	Logical DFD
It is implementation dependent. It shows which functions are performed.	It is implementation independent. It focuses only on the flow of data between processes.
It provides low level details of hardware, software, files, and people.	It explains events of systems and data required by each event.
It depicts how the current system operates and how a system will be implemented.	It shows how business operates; not how the system can be implemented.

Context Diagram

A context diagram helps in understanding the entire system by one DFD which gives the overview of a system. It starts with mentioning major processes with little details and then goes onto giving more details of the processes with the top-down approach.

The context diagram of mess management is shown below.



A data dictionary is a structured repository of data elements in the system. It stores the descriptions of all DFD data elements that is, details and definitions of data flows, data stores, and data stored in data stores, and the processes.

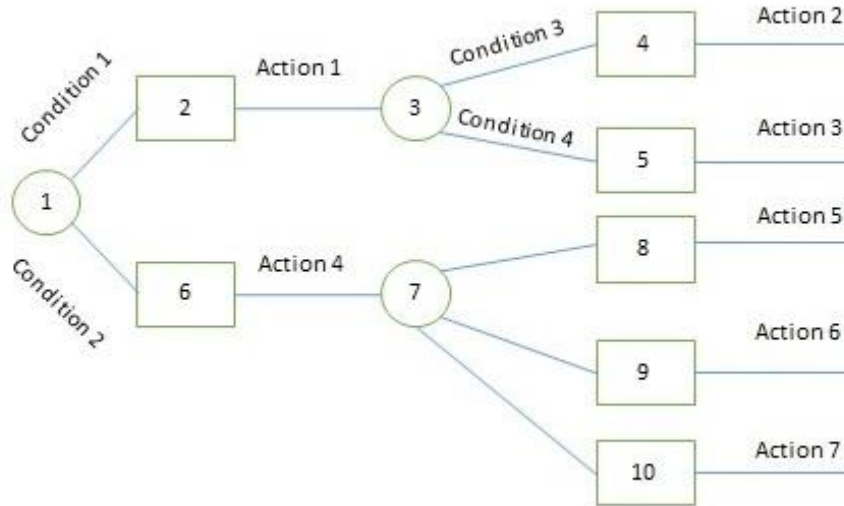
A data dictionary improves the communication between the analyst and the user. It plays an important role in building a database. Most DBMSs have a data dictionary as a standard feature. For example, refer the following table:

S.No.	Data Name	Description	No. of Characters
1	ISBN	ISBN Number	10
2	TITLE	Title	60
3	SUB	Book Subjects	80
4	ANAME	Author Name	15

Decision Trees

Decision trees are a method for defining complex relationships by describing decisions and avoiding the problems in communication. A decision tree is a diagram that shows alternative actions and conditions within horizontal tree framework. Thus, it depicts which conditions to consider first, second, and so on.

Decision trees depict the relationship of each condition and their permissible actions. A square node indicates an action and a circle indicates a condition. It forces analysts to consider the sequence of decisions and identifies the actual decision that must be made.



The major limitation of a decision tree is that it lacks information in its format to describe what other combinations of conditions you can take for testing. It is a single representation of the relationships between conditions and actions.

For example, refer the following decision tree –



Decision Tables

Decision tables are a method of describing the complex logical relationship in a precise manner which is easily understandable.

- It is useful in situations where the resulting actions depend on the occurrence of one or several combinations of independent conditions.
- It is a matrix containing row or columns for defining a problem and the actions.

Components of a Decision Table

- **Condition Stub** – It is in the upper left quadrant which lists all the condition to be checked.
- **Action Stub** – It is in the lower left quadrant which outlines all the action to be carried out to meet such condition.
- **Condition Entry** – It is in upper right quadrant which provides answers to questions asked in condition stub quadrant.
- **Action Entry** – It is in lower right quadrant which indicates the appropriate action resulting from the answers to the conditions in the condition entry quadrant.

The entries in decision table are given by Decision Rules which define the relationships between combinations of conditions and courses of action. In rules section,

- Y shows the existence of a condition.
- N represents the condition, which is not satisfied.
- A blank - against action states it is to be ignored.
- X (or a check mark will do) against action states it is to be carried out.

For example, refer the following table –

CONDITIONS	Rule 1	Rule 2	Rule 3	Rule 4
Advance payment made	Y	N	N	N
Purchase amount = Rs 10,000/-	-	Y	Y	N
Regular Customer	-	Y	N	-
ACTIONS				
Give 5% discount	X	X	-	-
Give no discount	-	-	X	X

Structured English

Structure English is derived from structured programming language which gives more understandable and precise description of process. It is based on procedural logic that uses construction and imperative sentences designed to perform operation for action.

- It is best used when sequences and loops in a program must be considered and the problem needs sequences of actions with decisions.
- It does not have strict syntax rule. It expresses all logic in terms of sequential decision structures and iterations.

Pseudocode

A pseudocode does not conform to any programming language and expresses logic in plain English.

- It may specify the physical programming logic without actual coding during and after the physical design.
- It is used in conjunction with structured programming.
- It replaces the flowcharts of a program.

Guidelines for Selecting Appropriate Tools

Use the following guidelines for selecting the most appropriate tool that would suit your requirements –

- Use DFD at high or low level analysis for providing good system documentations.
- Use data dictionary to simplify the structure for meeting the data requirement of the system.
- Use structured English if there are many loops and actions are complex.
- Use decision tables when there are a large number of conditions to check and logic is complex.
- Use decision trees when sequencing of conditions is important and if there are few conditions to be tested.

Structured Analysis (SA) in software engineering and its allied technique, **Structured Design (SD)**, are methods for analyzing and converting business requirements into specifications and ultimately, computer programs, hardware configurations and related manual procedures. Structured analysis and design techniques are fundamental tools of systems analysis, and developed from classical systems analysis of the 1960s and 1970s.

Objectives of Structured Analysis

Structured Analysis became popular in the 1980s and is still used by many. The analysis consists of interpreting the system concept (or real world) into data and control terminology, that is into data flow diagrams. The flow of data and control from bubble to data store to bubble can be very hard to track and the number of bubbles can get to be extremely large. One approach is to first define events from the outside world that require the system to react, then assign a bubble to that event, bubbles that need to interact are then connected until the system is defined. This can be rather overwhelming and so the bubbles are usually grouped into higher level bubbles.

Data Dictionaries are needed to describe the data and command flows and a process specification is needed to capture the transaction/transformation information. SA and SD were accompanied by notational methods including structure charts, data flow diagrams and data model diagrams, of which there were many variations, including those developed by Tom DeMarco, Ken Orr, Larry Constantine, Vaughn Frick, Ed Yourdon, Steven Ward, Peter Chen, and others.

These techniques were combined in various published System Development Methodologies, including Structured Systems Analysis and Design Method, Profitable Information by Design (PRIDE), Nastec Structured Analysis & Design, SDM/70 and the Spectrum Structured system development methodology.

History

Structured analysis is part of a series of structured methods, that "represent a collection of analysis, design, and programming techniques that were developed in response to the problems facing the software world from the 1960s to the 1980s. In this timeframe most commercial programming was done in Cobol and Fortran, then C and BASIC. There was little guidance on "good" design and programming techniques, and there were no standard techniques for documenting requirements and designs.

Systems were getting larger and more complex, and the information system development became harder and harder to do so". As a way to help manage large and complex software, the following structured methods emerged.

Since the end of the 1960s multiple Structured Methods emerged:

- Structured programming in circa 1967 with Edsger Dijkstra - "Go To Statement Considered Harmful"
- Niklaus Wirth Stepwise design in 1971
- Nassi-Shneiderman diagram in 1972
- Warnier/Orr diagram in 1974 - "Logical Construction of Programs"
- HIPO in 1974 - IBM Hierarchy input-process-output (though this should really be output-input-process)
- Structured Design around 1975 with Larry Constantine, Ed Yourdon and Wayne Stevens.
- Jackson Structured Programming in circa 1975 developed by Michael A. Jackson
- Structured Analysis in circa 1978 with Tom DeMarco, Yourdon, Gane & Sarson, McMenamin & Palmer.
- Structured Analysis and Design Technique (SADT) developed by Douglas T. Ross
- Yourdon Structured Method developed by Edward Yourdon.
- Structured Analysis and System Specification published in 1979 by Tom DeMarco.
- Structured Systems Analysis and Design Method (SSADM) first presented in 1983 developed by the UK Office of Government Commerce.
- IDEF0 based on SADT, developed by Douglas T. Ross in 1985.
- Hatley-Pirbhai modeling, defined in "Strategies for Real-Time System Specification" by Derek J. Hatley and Imtiaz A. Pirbhai in 1988.
- Information Engineering in circa 1990 with Finkelstein and popularised by James Martin.

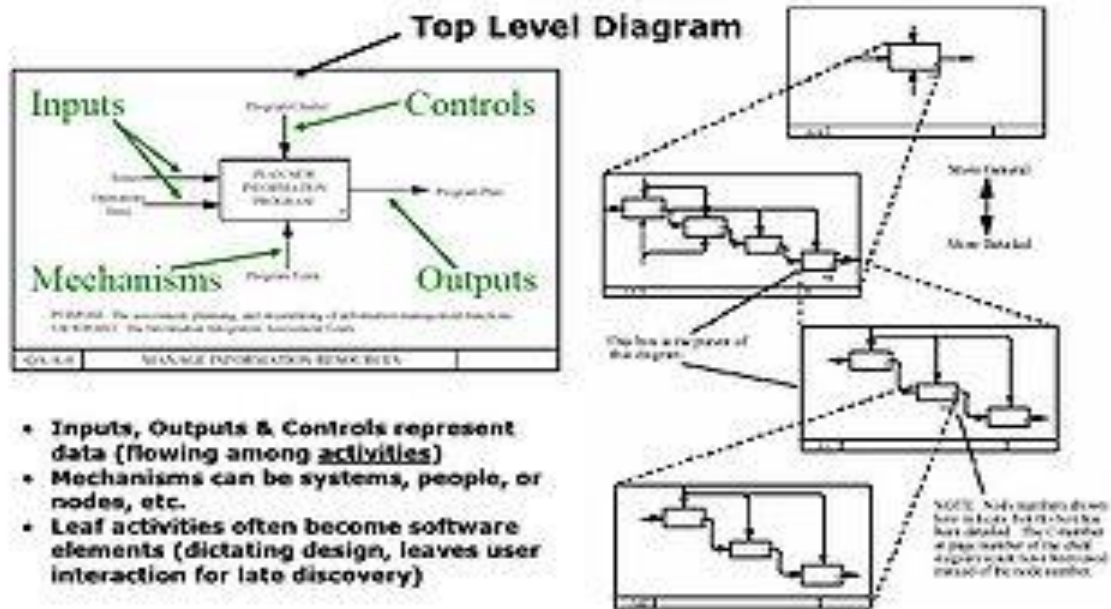
According to Hay(1999) "information engineering was a logical extension of the structured techniques that were developed during the 1970's. Structured programming led to structured design, which in turn led to structured systems analysis.

These techniques were characterized by their use of diagrams: structure charts for structured design, and data flow diagrams for structured analysis, both to aid in communication between users and developers, and to improve the analyst's and the designer's discipline.

During the 1980's, tools began to appear which both automated the drawing of the diagrams, and kept track of the things drawn in a data dictionary". After the example of computer-aided design and computer-aided manufacturing (CAD/CAM), the use of these tools was named Computer-aided software engineering (CASE).

Structured analysis topics

Single abstraction mechanism



Structured Analysis example

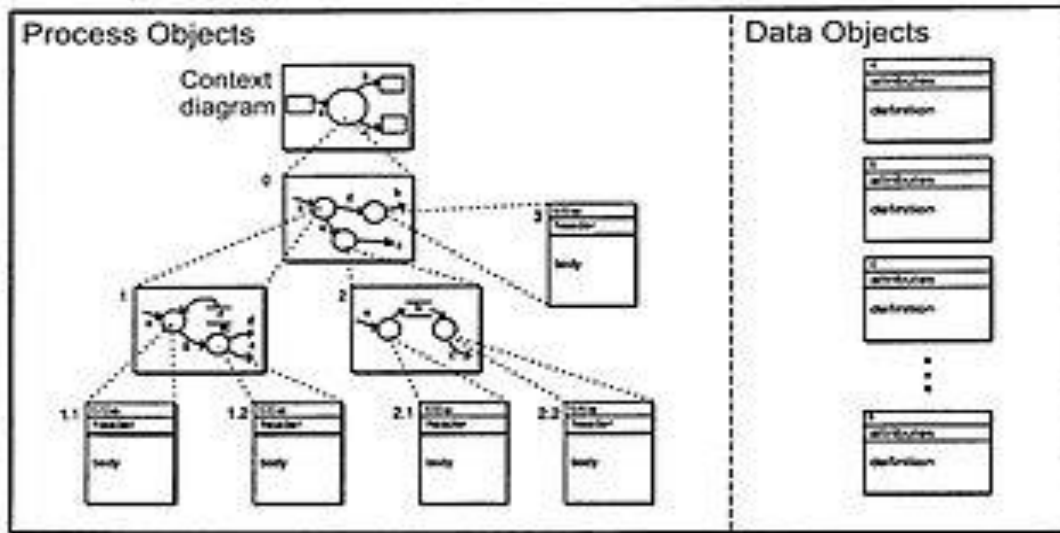
Structured analysis typically creates a hierarchy employing a single abstraction mechanism. The structured analysis method can employ IDEF (see figure), is process driven, and starts with a purpose and a viewpoint.

This method identifies the overall function and iteratively divides functions into smaller functions, preserving inputs, outputs, controls, and mechanisms necessary to optimize processes. Also known as a functional decomposition approach, it focuses on cohesion within functions and coupling between functions leading to structured data.

The functional decomposition of the structured method describes the process without delineating system behavior and dictates system structure in the form of required functions. The method identifies inputs and outputs as related to the activities. One reason for the popularity of structured analysis is its intuitive ability to communicate high-level processes and concepts, whether single system or enterprise levels. Discovering how objects might support functions for commercially prevalent object-oriented development is unclear. In contrast to IDEF, the UML is interface driven with multiple abstraction mechanisms useful in describing service-oriented architectures (SOAs).

Approach

Structured Analysis views a system from the perspective of the data flowing through it. The function of the system is described by processes that transform the data flows. Structured analysis takes advantage of information hiding through successive decomposition (or top down) analysis. This allows attention to be focused on pertinent details and avoids confusion from looking at irrelevant details. As the level of detail increases, the breadth of information is reduced. The result of structured analysis is a set of related graphical diagrams, process descriptions, and data definitions. They describe the transformations that need to take place and the data required to meet a system's functional requirements.

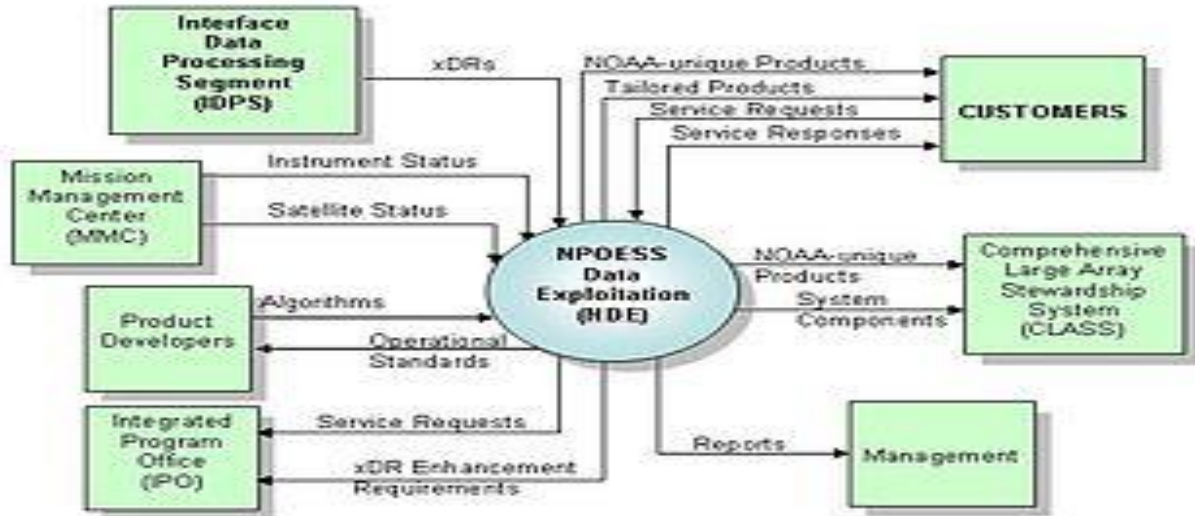


The structured analyse approach develops perspectives on both process objects and data objects.

De Marco's approach consists of the following objects (see figure):

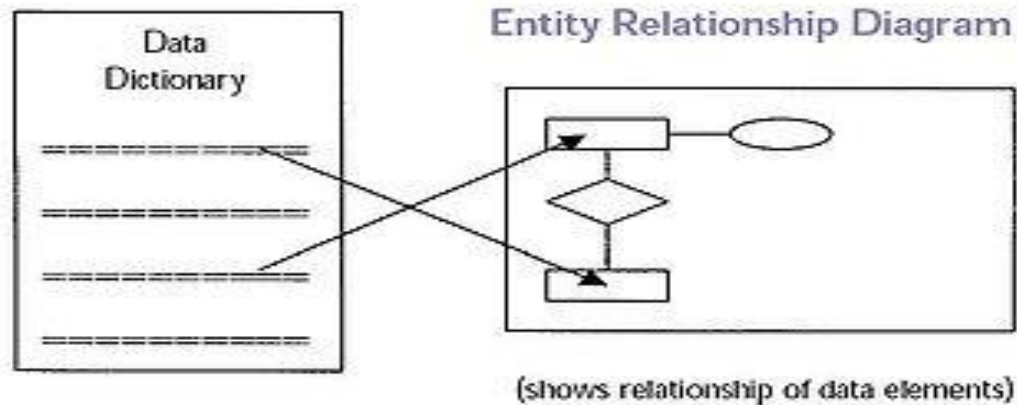
- Context diagram
- dataflow diagram,
- process specifications, and
- a data dictionary,

Hereby the Data flow diagrams (DFDs) are directed graphs. The arcs represent data, and the nodes (circles or bubbles) represent processes that transform the data. A process can be further decomposed to a more detailed DFD which shows the sub processes and data flows within it. The sub processes can in turn be decomposed further with another set of DFDs until their functions can be easily understood. Functional primitives are processes which do not need to be decomposed further. Functional primitives are described by a process specification (or mini-spec). The process specification can consist of pseudo-code, flowcharts, or structured English. The DFDs model the structure of the system as a network of interconnected processes composed of functional primitives. The data dictionary is a set of entries (definitions) of data flows, data elements, files and data bases. The data dictionary enmesh are partitioned in a top down manner. They can be referenced in other data dictionary entries and in data flow diagrams.

Context diagram

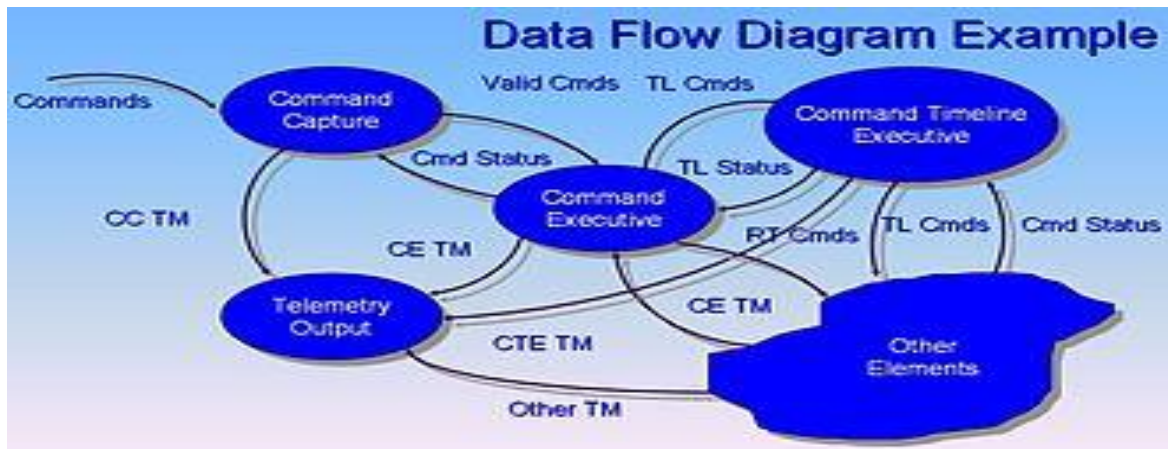
Context diagrams are diagrams that represent the actors outside a system that could interact with that system. This diagram is the highest level view of a system, similar to Block Diagram, showing a, possibly software-based, system as a whole and its inputs and outputs from/to external factors.

This type of diagram according to Kossiakoff (2003) usually "pictures the system at the center, with no details of its interior structure, surrounded by all its interacting systems, environment and activities. The objective of a system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of system requirements and constraints". System context diagram are related to Data Flow Diagram, and show the interactions between a system and other actors with which the system is designed to face. System context diagrams can be helpful in understanding the context in which the system will be part of software engineering.

Data dictionary

Entity relationship diagram, essential for the design of database tables, extracts, and metadata. A data dictionary or *database dictionary* is a file that defines the basic organization of a database. A database dictionary contains a list of all files in the database, the number of records in each file, and the names and types of each data field. Most database management systems keep the data dictionary hidden from users to prevent them from accidentally destroying its contents. Data dictionaries do not contain any actual data from the database, only book keeping information for managing it. Without a data dictionary, however, a database management system cannot access data from the database.

Database users and application developers can benefit from an authoritative data dictionary document that catalogs the organization, contents, and conventions of one or more databases. This typically includes the names and descriptions of various tables and fields in each database, plus additional details, like the type and length of each data element. There is no universal standard as to the level of detail in such a document, but it is primarily a distillation of metadata about database structure, not the data itself. A data dictionary document also may include further information describing how data elements are encoded. One of the advantages of well-designed data dictionary documentation is that it helps to establish consistency throughout a complex database, or across a large collection of federated databases.

Data Flow Diagrams*Data Flow Diagram example*

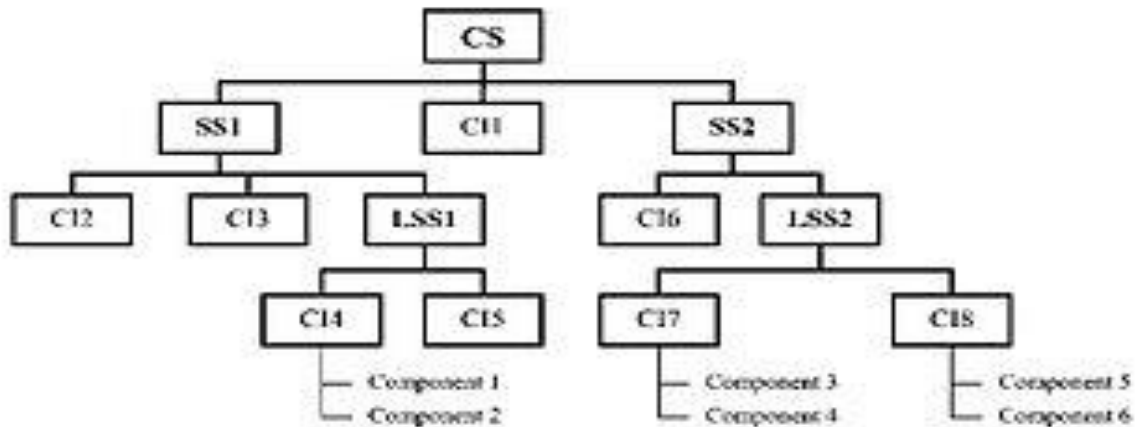
A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system. It differs from the system flowchart as it shows the flow of data through processes instead of computer hardware. Data flow diagrams were invented by Larry Constantine, developer of structured design, based on Martin and Estrin's "data flow graph" model of computation.

It is common practice to draw a System Context Diagram first which shows the interaction between the system and outside entities. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts. This context-level Data flow diagram is then "exploded" to show more detail of the system being modeled.

Data flow diagrams (DFDs) are one of the three essential perspectives of Structured Systems Analysis and Design Method (SSADM). The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a dataflow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's dataflow diagrams to draw comparisons to implement a more efficient system.

Dataflow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to re cook. How any system is developed can be determined through a dataflow diagram.

Structure Chart



A Configuration System Structure Chart

A Structure Chart (SC) is a chart, that shows the breakdown of the configuration system to the lowest manageable levels. This chart is used in structured programming to arrange the program modules in a tree structure. Each module is represented by a box which contains the name of the modules. The tree structure visualizes the relationships between the modules.

In structured analysis structure charts are used to specify the high-level design, or architecture, of a computer program. As a design tool, they aid the programmer in dividing and conquering a large software problem, that is, recursively breaking a problem down into parts that are small enough to be understood by a human brain. The process is called top-down design, or functional decomposition. Programmers use a structure chart to build a program in a manner similar to how an architect uses a blueprint to build a house. In the design stage, the chart is drawn and used as a way for the client and the various software designers to communicate. During the actual building of the program (implementation), the chart is continually referred to as the master-plan.

Structured Design

Structured Design (SD) is concerned with the development of modules and the synthesis of these modules in a so-called "module hierarchy". In order to design optimal module structure and interfaces two principles are crucial:

- *Cohesion* which is "concerned with the grouping of functionally related processes into a particular module", and
- *Coupling* relates to "the flow of information, or parameters, passed between modules. Optimal coupling reduces the interfaces of modules, and the resulting complexity of the software".

Structured Design was developed by Larry Constantine in the late 1960s, then refined and published with collaborators in the 1970s; see Larry Constantine: Structured Design for details. Page-Jones (1980) has proposed his own approach, which consists of three main objects: structure charts, module specifications and a data dictionary. The structure chart aims to show "the module hierarchy or calling sequence relationship of modules. There is a module specification for each module shown on the structure chart. The module specifications can be composed of pseudo-code or a program design language. The data dictionary is like that of structured analysis. At this stage in the software development lifecycle, after analysis and design have been performed, it is possible to automatically generate data type declarations",¹ and procedure or subroutine templates.^[10]

Structured query language

The structured query language (SQL) is a standardized language for querying information from a database. SQL was first introduced as a commercial database system in 1979 and has since been the favorite query language for database management systems running on minicomputers and mainframes. Increasingly, however, SQL is being supported by PC database systems because it supports distributed databases (see definition of distributed database). This enables several users on a computer network to access the same database simultaneously. Although there are different dialects of SQL, it is nevertheless the closest thing to a standard query language that currently exists.

Criticisms

Problems with data flow diagrams have been:

1. choosing bubbles appropriately,
2. partitioning those bubbles in a meaningful and mutually agreed upon manner,
3. the size of the documentation needed to understand the Data Flows,
4. still strongly functional in nature and thus subject to frequent change,
5. though "data" flow is emphasized, "data" modeling is not, so there is little understanding of just what the subject matter of the system is about, and
6. not only is it hard for the customer to follow how the concept is mapped into these data flows and bubbles, it has also been very hard for the designers who must shift the DFD organization into an implementable format

Feasibility Study

Many feasibility studies are disillusioning for both users and analysts. First, the study often presupposes that when the feasibility document is being prepared, the analyst is in a position to evaluate solutions. Second most studies tend to overlook the confusion inherent in system development-the constraints and the assumed attitudes. If the feasibility study is to serve as decision document it must answer three key questions:

1. Is there a new and better way to do the job that will benefit the user?
2. What are the costs and savings of the alternative (s)?
3. What is recommended?

The most successful system projects are not necessarily the biggest or most visible in a business but rather those that truly meets user expectations. More projects fail because of inflated expectations than for any other reason.

Feasibility Considerations

Three key considerations are involved in the feasibility analysis: economic, technical and behavioral. Let's briefly review each consideration and how it relates to the systems effort.

Economic Feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of a candidate system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. Otherwise, further justification or alterations in the proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the system life cycle.

Technical Feasibility

Technical feasibility centers on the existing computer system (hardware, software, etc.) and to what extent it can support the proposed addition. For example, if the current computer is operating at 80 percent capacity-an arbitrary ceiling- then running another application could overload the system or require additional hardware. This involves financial considerations to accommodate technical enhancements. If the budget is a serious constraint, then the project is judged not feasible.

Behavioral Feasibility

People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development of a computerized system. It is common knowledge that computer installations have something to do with turnover, transfers, retraining and changes in employee job status. Therefore, it is understandable that the introduction of a candidate system requires special effort to educate, sell and train the staff on new ways of conducting business.

In safe deposit example, three employees are more than 50 years old and have been with the bank over 14 years, four of which have been in safe deposit. The remaining two employees are in their early thirties. They joined safe deposit about two years before the study. Based on data gathered from extensive interviews, the younger employees want the programmable aspects of safe deposit (essentially billing) put on a computer.

Two of the three older employees have voiced resistance to the idea. Their view is that billing is no problem. The main emphasis is customer service – personal contact with customers. The decision in this case was to go ahead and pursue the project.

Steps in Feasibility Analysis

Feasibility analysis involves eight steps:

1. Form a project team and appoint a project leader.
2. Prepare system flowcharts.
3. Enumerate potential candidate systems.
4. Describe and identify characteristics of candidate systems.
5. Determine and evaluate performance and cost effectiveness of each candidate system.
6. Weight system performance and cost data.
7. Select the best candidate system.
8. Prepare and report final project directive to management.

1. Form a project Team and Appoint a Project Leader

The concept behind a project team is that future system users should be involved in its design and implementation. Their knowledge and experience in the operations area are essential to the success of the system. For small projects, the analyst and an assistant usually suffice; however, more complex studies require a project team. The team consists of analysts and user staff - enough collective expertise to devise a solution to the problem. In many cases, an outside consultant and an information specialist join the team until the job is completed.

Projects are planned to occupy a specific time period, ranging from several weeks to months. The senior systems analyst is generally appointed as project leader. He/she is usually the most experienced analyst in the team. The appointment is temporary, lasting as long as the project. Regular meetings take place to keep up the momentum and accomplish the mission – selection of the best candidate system. A record is kept of the progress made in each meeting. Regarding the safe deposit case, since the whole user area consists of five employees, the analyst handled most of the work.

2. Prepare System Flowcharts

The next step in the feasibility study is to prepare generalized system flowcharts for the system. Information – oriented charts and data flow diagrams prepared in the initial investigation are also reviewed at this time. The charts bring up the importance of inputs, outputs and data flow among key points in the existing system. All other flowcharts needed for detailed evaluation are completed at this point.

3. Enumerate Potential Candidate Systems

This step identifies the candidate systems that are capable of producing the outputs included in the generalized flowcharts. This requires a transformation from logical to physical system models. Another aspect of this step is consideration of the hardware that can handle the total system requirements. In the safe deposit case, it was found that virtually any microcomputer system with more than 128k –byte memory and dual disk drive will do the job. It was also learned that a microcomputer can be designed to interface with the bank's mainframe. In this design, actual processing is handled by the microcomputer, whereas information such as payments and credits are transmitted to the main computer files for proper adjustment through the customer's checking account.

An important aspect of hardware is processing and main memory. There are a large number of computers with differing processing sizes, main memory capabilities and software support. The project team may contact vendors for information on the processing capabilities of the system available.

4. Describe and Identify Characteristics of Candidate System

From the candidate systems considered, the team begins a preliminary evaluation in an attempt to reduce them to a manageable number. These packages were the result of a preliminary evaluation of more than 15 other packages – all purporting to meet the requirements, of the safe deposit billing system. When the number is reduced to three key packages, the next step is to describe in some detail the characteristics of each package.

For example, the first candidate system runs on an IBM PC with a minimum of 128K-bytes of memory. The software is written in Oracle, a relatively new language. In case of enhancements, change has to be made through the software house, since the source code is not available to the user.

5. Determine and Evaluate Performance and Cost Effectiveness of Each Candidate System

Each candidate system's performance is evaluated against the system performance requirements set prior to the feasibility study. Whatever the criteria, there has to be as close a match as practicable, although trade-offs are often necessary to select the best system. In the safe deposit case, the criteria chosen in advance were accuracy, growth potential, and response time less than five seconds, expandable main and auxiliary storage, and user-friendly software. Often these characteristics do not lend themselves to quantitative measures. They are usually evaluated in qualitative terms (excellent, good, etc.) based on the subjective judgment of the project team.

The cost encompasses both designing and installing the system. It includes user training, updating the physical facilities and documenting.

System performance criteria are evaluated against the cost of each system to determine which system is likely to be the most cost effective and also meets the performance requirements. The safe deposit problem is easy. The analyst can plot performance criteria and costs for each system to determine how each fare.

Costs are more easily determined when the benefits of the system are tangible and measurable. An additional factor to consider is the cost of the study design and development. The cost estimate of each phase of the safe deposit project was determined for the candidate system (IBM PC). In many respects, the cost of the study phase is a “sunk cost” (fixed cost). Including it in the project cost estimate is optional.

6. Weight System Performance and Cost Data

In some cases, the performance and cost data for each candidate system show which system is the best choice? This outcome terminates the feasibility study. Many times, however, the situation is not so clear – cut. The performance / cost evaluation matrix at times does not clearly identify the best system, so the next step is to weight the importance of each criterion by applying a rating figure. Then the candidate system with the highest total score is selected.

The procedure for weighting candidate systems is simple: -

1. Assign a weighting after factor to each evaluation criterion based on the criterion' effect on the success of the system. For example, if the usability criterion is twice as important as the accuracy factor, usability is assigned weight 4 and accuracy is assigned weight 2.
2. Assign a quantitative rating to each criterion's qualitative rating. For example, ratings (poor, fair, good, very good, excellent) may be assigned respective values (1,2,3,4,5).
3. Multiply the weight assigned to each category by the relative rating to determine the score.
4. Sum the score column for each candidate system.

Thus, the weighted candidate evaluation matrix is prepared using these steps, which in itself helps in the next step.

7. Select the Best Candidate System

The system with highest total score is judged the best system. This assumes the weighting factors are fair and the rating of each evaluation criterion is accurate. The criterion of growth potential is generally given the maximum weight, thus the greatest effect on the total score. Additionally, system development and user training are also given high weights. Most feasibility studies select from more candidate systems than we have mentioned in our example. The criteria chosen and the constraints are also more complex. In any case, management should not make the selection without having the experience to do so. Management cooperation and comments, however, are encouraged.

8. Feasibility Report

The culmination of the feasibility study is a feasibility report directed to management; it evaluates the impact of the proposed changes on the area(s) in question. The report is a formal document for management use, brief enough and sufficiently non technical to be understandable, yet detailed enough to provide the basis for system design. There is no standard format for preparing feasibility reports. Analysts usually decide on a format that suits the particular user and system. Most reports, however, begin with a summary of findings and recommendations, followed by document details. Starting with summary information highlights the essence of the report, giving management the option of reviewing the details later. The report contains the following sections:

1. Cover letter formally presents the report and briefly indicates to management the nature, general findings and recommendations to be considered.
2. Table of content specifies the location of the various parts of the report. Management quickly refers to the sections that concern them.

3. Overview is a narrative explanation of the purpose scope of the project, the reason for undertaking the feasibility study and the department(s) involved or affected by the candidate system. Also included are the names of the persons who conducted the study, when it began, and other information that explains the circumstance surrounding the study.

4. Detailed findings outline the methods used in the present system. The system's effectiveness and efficiency as well as operating costs are emphasized. The section also provides a description of the objectives and general procedures of the candidate system. A discussion of output reports, costs and benefits gives management a feel for the pros and cons of the candidate system.

5. Economic justification details point-by-point cost comparisons and preliminary cost estimates for the development and operation of the candidate system. A return on investment (ROI) analysis of the project is also included.

6. Recommendations and conclusions suggest to management the most beneficial and cost-effective system. They are written only as a recommendation, not a command. Following the recommendations, any conclusions from the study may be included.

7. Appendixes document all memos and data compiled during the investigation. They are placed at the end of the report for reference.

Disapproval of the feasibility report is rare if it has been conducted properly.

When a feasibility team has maintained good rapport with the user and his/ her staff it makes the recommendations easier to approve. Technically, the report is only a recommendation, but it is an authoritative one. Management has the final say. Its approval is required before system design is initiated.

Cost and Benefit Categories

In developing cost estimates for a system, we need to consider several cost elements. Among them is hardware, personnel, facility, operating and supply costs.

1. **Hardware costs** relate to the actual purchase or lease of the computer and peripherals (for example, printer, disk drive, tape unit). Determining the actual cost of hardware is generally more difficult when the system is shared by various users than for a dedicated stand-alone system. In some cases, the best way to control for this cost is to treat it as an operating cost.

2. **Personnel costs** include EDP staff salaries and benefits (health insurance, vacation time, sick pay, etc.) as well as pay for those involved in developing the system. Costs incurred during development of a system are one-time costs and are labeled developmental costs. Once the system is installed, the costs of operating and maintaining the system become recurring costs.

3. **Facility costs** are expenses incurred in the preparation of the physical site where the application or the computer will be in operation. This includes wiring, flooring, acoustics, lighting and air conditioning. These costs are treated as onetime costs and are incorporated into the overall cost estimate of the candidate system.

4. **Operating costs** include all costs associated with the day-to-day operation of the system; the amount depends on the number of shifts, the nature of the applications, and the caliber of the operating staff. There are various ways of covering operating costs. One approach is to treat operating costs as overhead. Another approach is to charge each authorized user for the amount of processing they request from the system. The amount charged is based on computer time, staff time and volume of the output produced. In any case, some accounting is necessary to determine how operating costs should be handled.

5. **Supply costs** are variable costs that increase with increased use of paper, ribbons, disks, and the like. They should be estimated and included in the overall cost of the system. A system is also expected to provide benefits. The first task is to identify each benefit and then assign a monetary value to it for cost/ benefit analysis. Benefits may be tangible and intangible, direct or indirect.

The two major benefits are improving performance and minimizing the cost of processing. The performance category emphasizes improvement in the accuracy of or access to information and easier access to the system by authorized users. Minimizing costs through an efficient system – error control or reduction of staff- is a benefit that should be measured and included in cost/benefit analysis.

Procedure for Cost/ Benefit Determination

There is a difference between expenditure and investment. We spend to get what we need, but we invest to realize a return on the investment. Building a computer – based system is an investment. Costs are incurred throughout its life cycle. Benefits are realized in the form of reduced operating costs, improved corporate image, staff efficiency, or revenues. To what extent benefits outweigh costs is the function of cost /benefit analysis. Cost/ benefit analysis is a procedure that gives a picture of the various costs, benefits and rules associated with a system.

The determination of costs and benefits entails the following steps:

1. Identify the costs and benefits pertaining to given project.
2. Categorize the various costs and benefits for analysis.
3. Select a method of evaluation.
4. Interpret the results of the analysis.
5. Take action.

Costs and Benefits Identification

Certain costs and benefits are more easily identifiable than others. For example, direct costs, such as the price of a hard disk, are easily identified from company invoice payments or canceled checks. Direct benefits often relate one-to-one to direct costs, especially savings from reducing costs in the activity in question. Other direct costs and benefits, however, may not be well defined, since they represent estimated costs or benefits that have some uncertainty. An example of such costs is reserve for bad debt. It is a discerned real cost, although its exact amount is not so immediate.

A category of costs or benefits that is not easily discernible is opportunity costs and opportunity benefits. These are the costs or benefits forgone by selecting one alternative over another. They do not show in the organization's accounts and therefore are not easy to identify.

Classifications of Costs and Benefits

The next step in cost and benefit determination is to categorize costs and benefits. They may be tangible or intangible, direct or indirect, fixed or variable. Each category is reviewed as follows:

Tangible or Intangible Costs and Benefits

Tangibility refers to the ease with which costs or benefits can be measured. An outlay of cash for a specific item or activity is referred to as a tangible cost. They are usually shown as disbursements on the books. The purchase of hardware or software, personnel training and employee salaries are examples of tangible costs. They are readily identified and measured.

Costs that are known to exist but whose financial value cannot be accurately measured are referred to as intangible costs. For example, employee morale problems caused by a new system or lowered company image is an intangible cost.

In some cases, intangible costs may be easy to identify but difficult to measure. For example, the cost of the breakdown of an online system during banking hours will cause the bank to lose deposits and waste human resources. The problem is by how much? In other cases, intangible costs may be difficult even to identify, such as an improvement in customer satisfaction stemming from a real-time order entry system.

Benefits are also classified as tangible or intangible. Like costs, they are often difficult to specify accurately. Tangible benefits, such as completing jobs in fewer hours or producing reports with no errors, are quantifiable. Intangible benefits, such as more satisfied customers or an improved corporate image, are not easily quantified. Both tangible and intangible costs and benefits, however, should be considered in the evaluation process. Management often tends to deal irrationally with intangibles by ignoring them.

Direct or Indirect Costs and Benefits

From a cost accounting point of view, costs are handled differently depending on whether they are direct or indirect. Direct costs are those with which an exact figure can be directly associated in a project. They are applied directly to the operation. For example, the purchase of a box of diskettes is a direct cost because we can associate the diskettes with the money spent.

Direct benefits also can be specifically attributable to a given project. For example, a new system that can handle 25 percent more transactions per day is a direct benefit. Indirect costs are the results of operations that are not directly associated with a given system or activity. They are often referred to as overhead. A system that reduces overhead realizes a saving. If it increases overhead, it incurs an additional cost. Insurance, maintenance, protection of the computer center, heat, light and air conditioning are all tangible costs, but it is difficult to determine the proportion of each attributable to a specific activity such as a report. They are overhead and are allocated among users, according to a formula.

Indirect benefits are realized as by-product of another activity or system. For example, our proposed safe deposit billing system that provides profiles showing vacant boxes by size, location and price, will help management decide on how much advertising to do for box rental. Information about vacant boxes becomes an indirect benefit of the billing even though it is difficult to specify its value. Direct and indirect costs and benefits are readily identified for tangible costs and benefits, respectively.

Fixed or Variable- Costs and Benefits

Some costs and benefits are constant, regardless of how well a system is used. Fixed costs are sunk costs. They are constant and do not change. Once encountered, they will not recur. Examples are straight – line depreciation of hardware, and insurance. In contrast, variable costs are incurred on a regular (weekly, monthly) basis. They are usually proportional to work volume and continue as long as the system is in operation.

For example, the costs of computer forms vary in proportion to the amount of processing or the length of the reports required. Fixed benefits are also constant and do not change. An example is a decrease in the number of personnel by 20 percent resulting from the use of a new computer. The benefit of personnel savings may recur every month. Variable benefits, on the other hand, are realized on a regular basis. For example, consider a safe deposit tracking system that saves 20 minutes preparing customer notices compared with the manual system. The amount of time saved varies with the number of notices produced.

Savings versus Cost Advantages

Savings are realized when there is some kind of cost advantage. A cost advantage reduces or eliminates expenditures. So we can say that a true savings reduces or eliminates various costs being incurred. There are savings; however, those do not directly reduce existing costs. To illustrate, examine the following case:

A systems analyst designed an online teller system that requires 14 new terminals. No reduction in personnel is immediately planned. Renovation of the bank lobby and the teller cages will be required. The primary benefits are:

1. Savings in teller's time to update account and post transaction.
2. Faster access and retrieval of customer account balances.
3. Available of additional data for tellers when needed.
4. Reduction of transaction processing errors.
5. Higher employee morale.
6. Capability to absorb 34 percent of additional transactions.

This is a case where no money can be realized as a result of the costs incurred for the new installation. There might be potential savings if additional transactions help another department reduce its personnel.

Similarly, management might set a value (in terms of savings) on the improved accuracy of teller activity, on quicker customer service, or on the psychological benefits from installing an online teller system. Given the profit motive, savings (or benefits) would ultimately be tied to cost reductions. Management has the final say on how well the benefits can be cost-justified.

Select Evaluation Method

When all financial data have been identified and broken down into cost categories, the analyst must select a method of evaluation. Several evaluation methods are available, each with pros and cons. The common methods are:

1. Net benefit analysis.
2. Present value analysis.
3. Net Present value.
4. Payback analysis.
5. Break- even analysis.
6. Cash-flow analysis

1. Net Benefit Analysis:- Net benefit analysis simply involves subtracting total costs from total benefits. It is easy to calculate easy to interpret, and easy to present. The main drawback is that it does not account for the time value of money and does not discount future cash flow. Period 0 is used to represent the present period. The negative numbers represent cash outlays. The time value of money is extremely important in evaluation processes. What is suggested here is that money has a time value. Today's dollar and tomorrow's dollar are not the same. The time lag accounts for the time value of money.

The time value of money is usually expressed in the form of interest on the funds invested to realize the future value. Assuming compounded interest, the formula is:

$$F = P (1 + i)^n$$

Where

F= Future value of an investment

P= Present value of the investment.

I= Interest rate per compounding period.

N= Number of years.

2. Present Value Analysis:- In developing long-term projects, it is often difficult to compare today's costs with the full value of tomorrow's benefits. As we have seen, the time value of money allows for interest rates, inflation and other factors that alter the value of the investment. Furthermore certain investments offer benefit periods that varies with different projects.

Presents value analysis controls for these problems by calculating the costs and benefits of the system in terms of today's value of the investment and then comparing across alternatives.

A critical factor to consider in computing present value is a discount rate equivalent to the forgone amount that the money could earn if it were invested in a different project. It is similar to the opportunity cost of the funds being considered for the project.

Suppose that Rs. 3,000 is to be invested in a microcomputer for our safe deposit tracking system and the average annual benefit is Rs. 1,500 for the four-year life of the system. The investment has to be made today, whereas the benefits are in the future. We compare present values to future values by considering the time value of money to be invested. The amount that we are willing to invest today is determined by the value of the benefits at the end of a given period (year). The amount is called the present value of the benefit.

3. Net Present Value:- The net present value is equal to discounted benefits minus discounted costs. Our Rs. 3,000 microcomputer investment yields a cumulative benefit of Rs. 4,758.51 or a net present gain of Rs.1,758.51. This value is relatively easy to calculate and accounts for the time value of money. The net present value is expressed as a percentage of the investment- in our example:
 $1,758.51/3,000 = 0.55$ percent

4. Payback Analysis: - The payback method is a common measure of the relative time value of a project. It determines the time it takes for the accumulated benefits to equal the initial investment. Obviously the shorter the payback period, the sooner a profit is realized and the more attractive is the investment.

The payback method is easy to calculate and allows two or more activities to be ranked. Like the net profit, though, it does not allow for the time value of money.

The payback period may be computed by the following formula:

$$\text{Overall cost outlay/ Annual cash return} \\ = (A*B) + (C*D) / (5+2) = \text{Years} + \text{Installation time (G) / Years to recover}$$

5. Break –even Analysis:- Break –even is the point where the cost of the candidate system and that of the current one are equal. Unlike the payback method that compares costs and benefits of the candidate system, break-even compares the costs of the current and candidate systems. When a candidate system is developed, initial costs usually exceed those of the current system.

This is an investment period. When both costs are equal, it is break-even. Beyond that point, the candidate system provides greater benefit (profit) than the old one--a return period.

A break-even chart compares the costs of the current and candidate systems. The attributes are processing cost and processing volume. Straight lines are used to show the model's relationships in terms of the variable, fixed and total costs of the two processing methods and their economic benefits. Intersection indicates the point where the total cost of processing transactions by the current system is equal to the total cost of using the candidate system. Beyond that point is the return period. Before the intersection is the investment period. According to the chart, then, it would be more economical to process manually when volume is below the number of breakeven point transactions. Higher processing volume favors the candidate system.

6. Cash – Flow Analysis: -

Some projects, such as those carried out by computer and word processing services, produce revenues from an investment in computer systems. Cash-flow analysis keeps track of accumulated costs and revenues on a regular basis. The “spread sheet” format also provides break – even and payback information. It is revenue minus expense on a period by period basis.

Drawbacks of the Cash flow analysis are: It ignores time value of money. For a limited period, it does not take into account the profitability of the project. It ignores behavioral implications of the numbers in the financial statement. However the major advantage of the cash flow analysis is that it combines benefits of break even and payback methods.

Interpret Results of the Analysis and Final Action

When the evaluation of the project is complete, the results have to be interpreted. This entails comparing actual results against a standard or the result of an alternative investment. The interpretation phase as well as the subsequent decision phase is subjective, requiring judgment and intuition. Depending on the level of uncertainty, the analyst may be confronted with a single known value or a range of values.

In either case, simpler measures such as net benefit analysis are easier to calculate and present than other measures, although they do not discount future cash flows. If it can be modified to include the time value of money, the net benefit method would be comparable to the net present value method. More complex measures such as net present value account for the time value of money but are more difficult to evaluate and present. The decision to adopt an alternative candidate system can be highly subjective, depending on the analyst's or end user's confidence in the estimated costs and benefits and the magnitude of the investment.

In summary, cost/ benefit analysis is a tool for evaluating projects rather than a replacement of the decision-maker. In real-life business situations, whenever a choice among alternatives is considered, cost / benefit analysis is an important tool. Like any tool, however, it has problems:

1. **Valuation problems:-** Intangible costs and benefits are difficult to quantify and tangible costs are generally more pronounced than tangible benefits. In most cases, then, a project must have substantial intangible benefits to be accepted.
2. **Distortion problems:-** There are two ways of distorting the results of cost/benefit analysis. One is the intentional favoritism of an alternative for political reasons. The second is when data are incomplete or missing from the analysis.
3. **Completeness problems:-** Occasionally an alternative is overlooked that compromises the quality of the final choice. Furthermore, the costs related to cost/ benefit analysis may be on the high side or not enough costs may be considered to do a complete analysis. In either case, the reliability of the final choice is in doubt.

The System Proposal

The final decision following cost/benefit analysis is to select the most cost-effective and beneficial system for the user. At this time, the analyst prepares a feasibility report on the major findings and recommendations. It outlines the options and recommendations. It is presented to management for determining whether a candidate system should be designed. Effective reports follow carefully planned formats that management can understand and evaluate without having to read the entire document.

Part A (ONE Mark)

Multiple Choice Questions

Online Examination

Part B (2 Marks)

1. What do you mean by Strategic and operational planning?
2. What fact-finding techniques would use for investigating the information requirement of a large organization?
3. Where are brain storming & Delphi methods used? Discuss.
4. What do you mean by prototyping?
5. What are the data flow diagrams? How are they different from structure charts?
6. Discuss the behavioral issues involved in understanding the analyst/user interface.

Part C (5 Marks)

1. A question may be open or closed-ended. Illustrate the difference.
2. Elaborate on the pros and cons of data analysis & decision analysis method.
3. Describe the process of design. Also discuss Logical & physical design.
4. Describe the various goals considered for input & output design?
5. What important information does the user's request form provide? Why is it so important in the initial investigation? Explain in detail.
6. List four situations that make use of questionnaire appropriate.
7. Describe the concept and procedure used in constructing DFDs. Give an example.
8. Discuss the procedure for constructing a questionnaire.
9. Discuss the utility of Decision table & Structure chart.

	KARPAGAM ACADEMY OF HIGHER EDUCATION					
	Department of Management					
	Unit 2- System Analysis and Design -Multiple Choice Questions- Each Question Carry ONE Mark					
S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	In system design how much phases of development?	one	two	three	four	two
2	An HIPO diagram derived fromdiagram?	hierarchy	structured	information structure	un- structured	structured
3refers to the numbers of connections between a "calling" and a "called" module	module	module chart	moduleing	module coupling	moduleing
4 Are variable costs that increase with increased use of paper	supply costs	hardware costs	software costs	production cost	supply costs
5 Refers to the ease with which costs or benefits can be measured?	intangible	tangible	tangible	intangibility	tangible
6	An outlay of cash a specific item or activity is referred to as a	intangible costs	tangible costs	tangibility cost	intangibility costs	tangible costs
7 are the results of operations that are not directly associated with a given system or activity	direct costs	indirect costs	direct benefits	indirect benefits	direct benefits
8 are realized as a by-product of another activity or system	direct benefits	indirect benefits	direct costs	indirect costs	direct benefits
9are also constant and do not change	fixed benefits	unfixed benefits	variable benefits	none	unfixed benefits
10is a data-flow based methodology	structured design	structured system	structured tools	structured systems	structured tools
11 Refers to forms driven technique in that standard forms are used to document the information	hipo	hoip	hipo	hiop	hipo
12 activity deals with the design of the physical data base	data design	program design	data base design	program base design	data base design

13conjunction with data base design is a decision on the programming language?	data design	program design	data base design	program base design	program design
14 Check ensures that all fields in a record are present and a read in the proper sequence?	consistency	completencies	reasonableness	sequenceness	reasonableness
15refers to the relevance of one type of data to another	consistency	completencies	reasonableness	sequenceness	completencies
16 Check evaluates a transaction against a standard to test?	consistency	completencies	reasonableness	sequenceness	sequenceness
17verifies that data records are in suquence prior to processing	sequence check	lifecycle	variable benefits	all the above	all the above
18 Design is the process of converting user-originated computer based format?	input	output	input data	output data	input data
19 Are the either 80 or 96 columns wide.	input cards	output cards	punch cards	unpunch cards	unpunch cards
20	Expand OCR?	optical code role	optial character recognition	optical convert role	optical constrant role	optical convert role
21	Expand CRT	cathode ray tube	cathode role test	cathode entry test	none of the above	cathode role test
22	in requirements how much points are involved in forms design	five	nine	ten	three	nine
23forms may be printed on paper of different colors,grades,and weights?	paper color	paper code	paper selection	paper	paper selection
24 is generally classified as onionskin,bond,duplicator,ledger,ect,,,.....	paper color	paper code	paper selection	paper	paper code
25	A Is a organised to ensure that records are available for procesing	documents	papers	file	Folder	papers

26	A record maintains a logical relationship among all the data items in the record	file	documents	paper selection	logical	documents
27	How much points are involved in organizing files?	two	three	four	five	three
28	Thearea contains records added to the file that cannot be placed in logical sequence in the prime area?	physical	logical	overflow	none	logical
29	A data base is a collection of interrelated data stored with Redundancy	minimum	maximum	medium	range	maximum
30	Which of the following Information systems are aimed at improving the routine business activities on which all organizations depend?	Management Information systems) Decision support systems	Transaction processing systems	Management support systems	Transaction processing systems
31	Which of the following strategies are adopted if information requirements are not well-defined?	Rapid application development method) Structured analysis development method	Systems development life cycle method	Prototyping method	Prototyping method
32	Structured Programming involves:	functional modularization	localization of errors	decentralized programming	stress on analysis	functional modularization
33	Which of the following is not a fact-finding technique?	Third party enquiry	Interview	Questionnaire	Record reviews	Third party enquiry

34	Which of the following questions are useful in evaluating data flow diagrams?	Are there any unnamed components in the data flow diagram?	Are there any processes that do not receive input?	Are there any data stores that are input but never referenced?	All (a), (b) and (c) above.	All (a), (b) and (c) above.
35	In system design and development field what does spaghetti code mean?	programs written in unstructured languages.	well structured and well documented code.	program code that has many GOTO statements.	Both (a) and (c) above	Both (a) and (c) above
36	Which of the following statements is false with respect to a Data Dictionary?	It is a repository of the elements in a system.) data dictionary and data store both are same	It manages detail	It communicates the common meanings for system elements and activities.	data dictionary and data store both are same
37	Cost-Benefit Analysis is performed during	Analysis phase) Design phase	Feasibility study phase	Implementation phase	Feasibility study phase
38	Which of the following technique detects transposition errors?	check digit) automatic correction	existence test	duplicate processing	check digit
39	The structure chart derived by studying the flow through the system supports the activity of	File design	Program design	Database design	Internal controls design	Internal controls design

40	Which of the following suggests phased implementation of the system?	introduce a new system gradually either by functions or by organizational units.) procure resources in stages and then introduce the system at once	Withdraw the existing system gradually	allow the new system and old system to run parallel for sometime	Withdraw the existing system gradually
41	Class is analogous to	object) blue print	instance	record	blue print
42	Which of the following represents the condition of an object at a specific moment in time?	behavior) properties	instance	state	state
43	Some object-oriented systems permit a class to inherit its state(attributes) and behaviors from more than one super class. This is called	multiple inheritance	inheritance	hybrid inheritance	specialization	multiple inheritance
44	Identify the following who presented the object modeling technique (OMT)	Booch	Jim Rumbaugh ET AL	Jacobson ET AL	J. Johnson	Jim Rumbaugh ET AL
45	Which of the following statements is false with respect to a use case?	scenario for understanding	between the users and the system	responsibility of the system to its	the flow of activities of	flow of activities of
46	Which type of association does the following diagram depict?	aggregation	composition	specialization	simple association	aggregation
47	Which of the following statements is false with respect to the diagram	The building is composed) An aggregation relationship exists	A room can have many rooms	There is a recursive	An aggregation relationship
48	What category of information system determines the sale of an item and a withdrawal from an	Management Information	Executive Information System	Transaction Processing	Communication Support	Transaction Processing

49	Which of the following is not true regarding the waterfall method?	Fairly rigid approach	Can easily go back to previous phases	Good for traditional type	Not as good for many of the	Can easily go back to
50	Which feasibility determines the availability of team and support staff?	Economic Feasibility	Cultural Feasibility	Resource Feasibility.	Schedule feasibility	Resource Feasibility.
51	Which among the following is an intangible benefit?	Maintaining constant staff	Decreasing operating expenses	Survival	Reducing error rates	Survival
52	Which chart is represented by vertical bars?	PERT	ROI	GANTT	NPV	GANTT
53	Which of the model is used for system components?	PERT chart	Gantt chart	Organizational hierarchy chart	DFD.	DFD.
54	Which is not used in context level diagram?	Source	Destination	Data flow	Data Store	Data Store
57	What will help the system analyst to work with users to determine system usage?	Use case	Actor	Class	Component	Use case
58	Which UML diagram provides a variety of symbols and encompasses a number of ideas, all to model the changes which just one object goes through?	Package	Object	State	Class	State
59	Which relationship specifies an optional behavior?	An extend.	A generalization	An inheritance	An include	An extend
60	Which among the following literally means 'many forms', the concept that different objects can respond to the same message in different ways?	Composition	Encapsulation	Polymorphism	Aggregation	Polymorphism

UNIT-III

SYLLABUS

System Design: The process and stages of system design - Major development activities - Input and output forms design - File organization and database design - Sequential - Logical and Physical views of data – Normalization – Different forms of normalization.

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering

Architectural design:

The architectural design of a system emphasizes on the design of the systems architecture which describes the structure, behavior, and more views of that system.

Logical design:

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams.

Physical design:

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified/ authenticated, how it is processed, and how it is displayed as In Physical design; the following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

Explain the stages of System Design?

System Design involves the transformation of the plans obtained in the system analysis phase into a function system. The actual work of the Design stage are,

1. Design of the data that needs to be processed, how data will be organized and how output data will be presented. Design of the User Interface. Algorithms for the main program modules are produced using techniques such as flowcharts, pseudo code, JSP and HIPO charts etc.
2. Hardware and software selection and their specifications.
3. Design of testing strategies.

Process Steps for Designing the System Dataflow

Designing the system dataflow consists of identifying the data sources, objects, and attributes that are used in the met directory deployment and describing the policies that define the actions taken when these objects are added, modified, or deleted. Dataflow design is the discovery and recording of data source information, of relevant objects in each data source for synchronization, of real-world object policies and actions, and of which attributes from each object should push out from the met averse. This information is recorded on the following worksheets, which are provided for you:

- Real-World Identity Objects
- Connected Data Sources
- Object-Level Policies
- Included Attributes
- Outbound Attribute Flow
- Met averse Object Design

Begin to fill out these worksheets as you collect the information described below. Some of the information will be easily apparent; however, other information may require persistent investigation or decisions not yet made.

Therefore, you might have to return to these worksheets when the information is available, particularly during the dataflow design process. When you prepare these new worksheets, begin by adding the initial data. Then return to the worksheets as needed, gradually refining your logical design.

These steps outline this process and the subsections that follow provide details of each step:

- **Step 1: Identify real-world identity types.** Identify each real-world identity type and begin filling out a Real-World Identity Objects worksheet for each one.
- **Step 2: Identify your data sources.** For each management agent listed on the Real-World Identity Objects worksheets, begin filling out a Connected Data Sources worksheet.
- **Step 3: Identify the system objects.** For each object, begin filling out an Object-Level Policies worksheet. List all business policies that would apply to each type of object activity, such as adding, changing, or removing an object.
- **Step 4: Identify system attributes.** For each connected data source object, fill out an Included Attributes worksheet. List all attributes that need to provide data to the met directory.
- **Step 5: Diagram your design.** Draw a diagram of your proposed design.
- **Step 6: Establish a met averse object type.** Represent each real-world identity type, by completing a Met averse Object Design worksheet for each object type.
- **Step 7: Obtain stakeholder approval for your dataflow model.**

Before you begin, review the sample worksheets to see detailed examples of how to use them. Then review the worksheet samples together with the detailed step descriptions that follow. The step subsections cover the process of creating the met directory-related worksheets. They include suggestions and helpful questions to consider as you complete the worksheets. Additionally, consider diagramming the processes.

Input-Output Analysis:

Input-output analysis is “a technique used in economics for tracing resources and products within an economy. The system of producers and consumers is divided into different branches, which are defined in terms of the resources they require as inputs and what they produce as outputs. The quantities of input and output for a given time period, usually expressed in monetary terms, are entered into an input-output matrix within which one can analyze what happens within and across various sectors of an economy where growth and decline takes place and what effects various subsidies may have” (Krippendorff).

Inputs, Outputs and Process:

All systems have inputs, outputs and processes.

A simple version of a factory system would be

1. raw materials go in (inputs)
2. They are worked upon and changed (processes)
3. Manufactured goods come out (outputs).

Of course a system is more complicated and often involves several inputs, processes and outputs which do not necessarily happen within a short space of time in the same country.

Using Indexes and Sequences

An index is an optional structure designed to help you gain faster access to data. Just like the index in this book, an Oracle index is logically and physically independent of the data in the associated table or cluster. You can use the index to speed access to the data or you can retrieve the data independently from the index by searching the tables for it. When optimally configured and used, indexes can significantly reduce I/O to the data files and greatly improve performance.

The presence of an index is transparent to the user or application and requires no application changes. However, if you are aware of an index, you should be able to design your applications to take better advantage of those indexes. The only indication of an index might be an improved access time to data.

The index itself should be created with some knowledge of the application and data-access patterns. If indexes are created on columns that are not used to access the data, the index is useless. After an index has been created for a table, Oracle automatically maintains that index. Insertions, updates, and deletions of rows in the table automatically update the related indexes.

A table can have any number of indexes, but the more indexes there are, the more overhead is incurred during table updates, insertions, and deletions. This overhead is incurred because all associated indexes must be updated whenever table data is altered.

What is the difference between logical scheme and physical scheme?

A logical schema won't exist in your database. A logical schema is a design-centric database structure built to meet your business requirements. It is a model that exists on a white board or in a diagramming tool. It is like the architect's drawings of your database.

A physical model is what is *actually implemented* in your DBMS.

The two can be different for a variety of reasons and in several ways:

- Your logical model should be properly normalized, but your physical model may have de normalization which you've added deliberately and for all the right reasons.
- Your physical model may have different naming conventions. Some people use plain English (or the language of choice) for their logical models and impose a more "system-ish" naming convention in their physical model.
- Your logical model may have many-to-many relationships. Physical models implement using *intersection tables*.

- Your logical model may only use *natural* or business keys. Physical models may also add *surrogate* keys.

You should have a logical model because it lets you think about your database design without having to get bogged down in physical constraints. You need to have the physical model because that will be where your data eventually lives.

Logical view and physical view

A database management system provides the ability for many different users to share data and process resources. But as there can be many different users, there are many different database needs. The question now is: How can a single, unified database meet the differing requirement of so many users?

A DBMS minimizes these problems by providing two views of the database data: a physical view and a logical view. The physical view deals with the actual, physical arrangement and location of data in the direct access storage devices(DASDs). Database specialists use the physical view to make efficient use of storage and processing resources. Users, however, may wish to see data differently from how they are stored, and they do not want to know all the technical details of physical storage. After all, a business user is primarily interested in using the information, not in how it is stored.

The logical view/user's view, of a database program represents data in a format that is meaningful to a user and to the software programs that process those data. That is, the logical view tells the user, in user terms, what is in the database. One strength of a DBMS is that while there is only one physical view of the data, there can be an endless number of different logical views. This feature allows users to see database information in a more business-related way rather than from a technical, processing viewpoint. Thus the logical view refers to the way user views data, and the physical view to the way the data are physically stored and processed.

Integrated Design Process:

The intent of this guide is to explain the numerous advantages of the Integrated Design Process (IDP); to provide enough information to start applying it to your design projects;

and, to help you find additional useful sources of IDP tools and information.

After reading this article you will:

- _ Understand what IDP is and how it is different from traditional approaches;
- _ Understand the benefits of IDP and why it is critical for achieving sustainable design;
- _ Understand how to generally structure an IDP process;
- _ Understand who needs to be involved, when and why;
- _ Understand key success factors to apply IDP; and
- _ Be able to find additional tools and resources to apply IDP.

IDP—What is it?

Integrated Design Process (IDP) was used in the early 1990s, by Canada's C-20001 program and IDEAS challenge2 competition to describe a more holistic approach to building design. This design process has been shown to produce more significant results than did investment in capital equipment3. There is now no single "right" definition for IDP. Rather, IDP describes a different, intentional way of approaching sustainable building and community design that offers a much higher likelihood of success than any other approach. There are an increasing number of practitioners of IDP.

Each has a different, and valid, perspective on how to do it, based on their experiences and practices. Most would agree that there are common elements to every definition. _ *Goal-driven* with the primary goal being sustainability, but with explicit subsidiary goals, objectives and targets set as a means to get there. _ *Facilitated* by someone whose primary role is not to produce the building design or parts of it, but to be accountable for the *process* of design. _ *Structured* to deal with issues and decisions in the right order, to avoid locking in bad performance by making non-reversible decisions with incomplete input or information. _ *Clear decision-making* for a clearly understood methodology for making decisions and resolving critical conflicts.

Inclusive—everyone, from the owner to the operator, has something critical to contribute to the design and everyone must be heard.

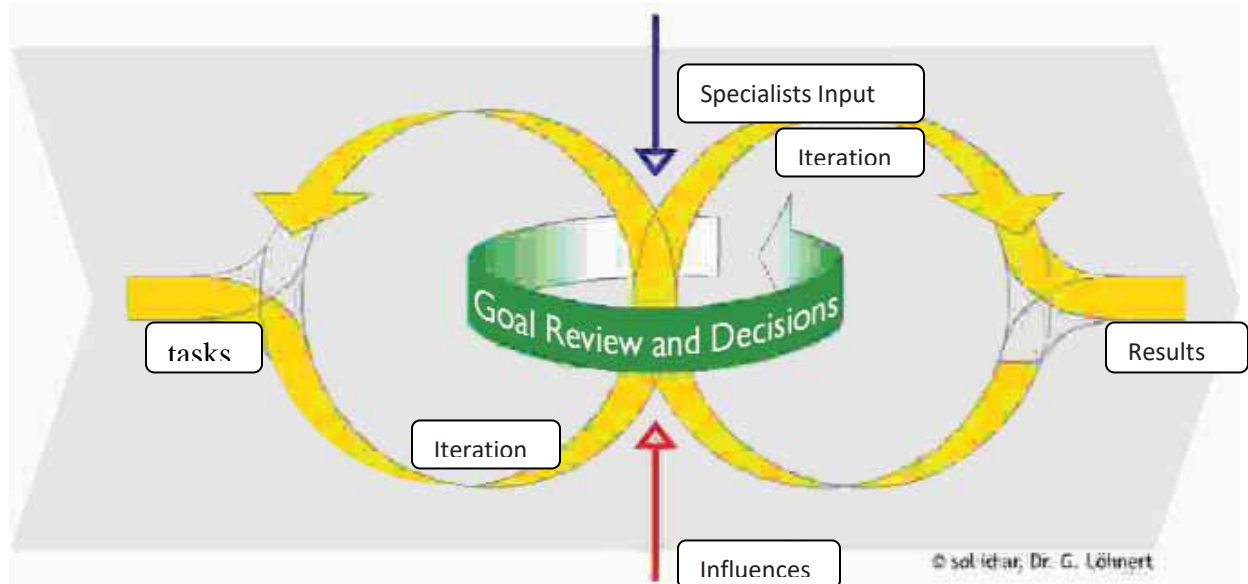
Collaborative so that the architect is not simply the form-giver, but more the leader of a broader team collaboration with additional active roles earlier in the process.

Holistic or systemic thinking with the intent of producing something where the whole is greater than the sum of the parts, and which may even be more economic. _ *Whole-building budget setting*—allows financial trade-offs, so money is spent where it is most beneficial when a holistic solution is found.

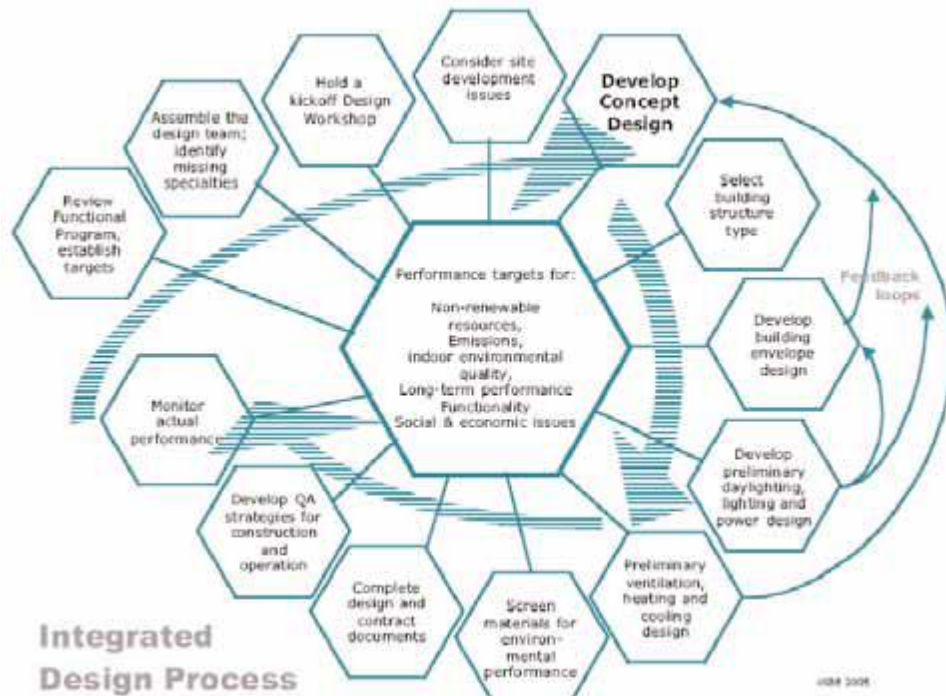
Iterative—to allow for new information to inform or refine previous decisions.

Non-traditional expertise—on the team, as needed, or brought in at non-traditional times to contribute to the process.

Sustainability is one of the most important issues facing human society today. The challenges as they relate to buildings are complex and the solutions are not simple. Framing the challenge in terms of *motivation* and *means* is one way of clarifying our thinking. Motivation proceeds from a source or ground, towards a goal. The means require some tools and a direction to apply them.



It is also important to follow through on the iterations in the IDP process by explicitly identifying subsequent IDP tasks and group meetings, interwoven with the overall project schedule. If this is not done upfront, it is too easy for the design team to revert to familiar, business-as-usual, linear design processes after the excitement and energy of the initial kickoff charrette begins to wane. An explicit IDP process schedule is a key tool to managing the IDP process.



What does an IDP process look like?

At the beginning of this article, it was pointed out that there are a number of different practitioners of IDP, each with a different perspective on how to carry it out. A number of people and organizations have identified the steps involved in applying IDP to a design. It may be useful to summarize three different approaches. Links are provided to explore the details of each of these. The 3 are 1. The approach by Nils Larsson of International Initiative for Sustainable Built Environment (iisBE). 2. The Integrative Design Collaborative²⁰ approach by Bill Reed. 3. The process definition developed at national workshop held in Toronto in 2001.²¹ In addition, IEA Task 23 has published quite a detailed guideline and accompanying software²² that are also useful tools..

Objections to IDP

It is worth discussing some of the standard objections that are raised to the Integrated Design Process. The objections usually are phrased as the following: **“We’ve always done IDP”** –

That may be true, and if so, keep doing it. Usually the people who say this, however, have remarkably few green buildings to show as evidence. **“If you want me to do something different, that implies I’ve been doing it wrong all these years”** – Well no, it doesn’t. This is the 21st century, with an entirely new situation for human society, and new problems and demands for the profession to respond to. Think of IDP as a new tool to add to the toolbox to address this new situation. **“The client won’t pay for it”** – Possibly not, especially the first time when the value has not been demonstrated to the client, but that’s exactly what is the intent of NRCan’s Commercial Building Incentive Program.

Integrated Design Process

Database Design Process

There are six stages in the design of a database:

1. requirement analysis
2. conceptual database design
3. choice of the DBMS
4. data model mapping
5. physical design
6. implementation

Requirement Collection and Analysis

Purpose: to document the data requirements of the users

– *functional requirements* are the operations that will be applied to the database, including queries and update the specification will then be used as the basis for the design of the

Database typical activities:

- identification of application areas and user groups
- analysis of existing documentation of application areas, e.g. policy documents, forms, reports, organisation charts

- analysis of current operating environments and the planned use of the information, e.g. information flow, types of transactions, frequency of transaction types
- responses to user questionnaires are analysed ... in summary:
start from a description of the requirements which is:
 - poorly structured,
 - heterogeneous
 - informal and use a technique to transform that into a specification of the database requirements which is:
 - formal
 - homogeneous
 - consistent
 - complete

2. Conceptual Design

Two parallel activities

1. **schema design**– examines the data requirements of the database resulting from the analysis (phase 1) and produces a conceptual schema in a DBMS-independent high level data model
2. **transaction design**– examines the database applications whose requirements were analysed in phase 1 and produces high level specifications for these transactions

2.1. Conceptual Schema Design

Purpose: to produce a *conceptual schema* of the database

- expressed using concepts of the high level data model

not including implementation details (has to be understood by non-technical users)

but detailed in terms of the “objects” of the domain the database will represent independent of the DBMS to be used (no relational DB-oriented notions! Cannot be used directly to implement the database design is made in terms of a *semantic* or *conceptual* data model)

Output Design

The term output necessarily implies to information printed or displayed by an information system.

Following are the activities that are carried out in the output design stage:

- Identification of the specific outputs required to meet the information requirements.
- Selection of methods required for presenting information.
- Designing of reports, formats or other documents that acts as carrier of information.

Output Design Objectives

The output design of an information system must meet the following objectives.

- 1.) The output design should provide information about the past, present or future events. The operational control level outputs provide information of the past and present events. On the other hand, required at the strategic planning level provide information of the future events.
- 2.) The output design should indicate the important events, opportunities and problems.
- 3.) The output design should be designed keeping in mind that an action must be triggered in response to some event. A set of rules is pre-defined for such trigger.
- 4.) The output design should produce some action to the transaction. For example, when the telephone bill is received, a receipt is printed.

Presentation of Output

The next consideration in the output design is the presentation involved with an information system. The presentation of an output is regarded as an important feature of output design. The presentation of an output represented either in tabular or graphical form or in both forms. A tabular format is preferred in the following conditions:

- When the details dominate the content of the output

- When the contents of the output are classified in groups.
- When the output design are to be compared.

A tabular format is also preferred for the detailed reports.

Graphical representation is used to improve the effectiveness of the output because some users prefer to view information in graphic form rather than in rows and columns of the tables. The tabular and graphical formats may be combined together to enhance the presentation of the output.

Output Design Specifications

The specifications for the output design should be considered first while designing any output. The main points in the output design specifications are :

- Paper size: It is important for a system designer to specify the size of the paper to be used for the output. The size of the paper can be A4 or A3 size. It can also be 9.5 * 11 or 11 * 14.7/8 inches.
- Special forms: Outputs can be designed on the pre-printed form. The pre-printed form requires the standards print headings or titles for the output design. Let us say, an organisation wants to display the name and logo of an organisation at the top of the output document produced by the information organisations at the top of the output document produced by the information system. Other organisations may require the address of an organisation should also be displayed along with the name and logo of the organisation. The output display depends on the choice of organisation and it varies in different organisation. The different ideas can be helpful in enhancing the presentation skills of the output document of that organisation.
- Multiple copies of output: At times, more than one copy of an output is required and in such cases, multipart forms can be used to produce multiple copies of the output. Multiple papers are available in carbon and carbonless forms.
- Turnaround documents : The output can be produced as a turnaround document. In this specification, the output can be used as an input document also.

- The turnaround documents can be used in the organisations where the optical scanners are used for reading data from the forms.
- Output layout: The output layout may be defined as the arrangement of items on the output medium. The layout design guides a programmer in the development of codes.
- Headings and date
- Data and details
- Summaries and totals
- Page title, number and date
- Notes and comments
- Column headings and data type

The designers usually use N[n] for numeric data type and X[n] for alpha data type, where n specifies the width of the column.

A system designer may design multiple screens or special windowing capabilities such as pop-up windows for designing screens. Such design will enhance readability for the visual displays.

Input Design

Similar to the output design, input design is equally important for a system designer. This is because output from a system is regarded as the foremost determinant for defining the performance of a system. The output of the system greatly affects the input design of the system.

Input Design Objectives

The input design of an information system must the following objectives:

- The input design of the system must attempt and try reducing the data requirements. It should also avoid capturing un necessary data such as constant and system-computable data.
- The input design must avoid processing delays during data entry. Capturing automatic data can reduce this kind of delay.

- The input design must avoid data entry errors. This can be achieved by checking the errors in data entry program. This technique of checking data entry program programs for errors is known as input validation technique.
- The input design must keep the process simple and easy to use.

Input layout

The layout of the input design must contain the following items.

1. Headings and date of data entry.
2. Data heading and value
3. Data type and width of the column
4. Initials of data entry operator

Normalization

Normalization process is the establishment of the database structure so that most of the ambiguity can be removed. Normalization stage, starting from the most mild (1NF) to most stringent (5NF). Usually only up to the level of 3NF or BCNF because already sufficient to generate the table—a table of good quality. Normalization is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics—insertion, update, and deletion anomalies that could lead to a loss of data integrity.

Normal form (normal form) is a database of the scheme class relations defined for the purpose of high integrity and maintainability. Creation of a normal form is called normalization. Normalization achieved with analysis dependence among the attributes of each individual associated with that relation.

Normalization Database

Normalization be used to optimizing table structures, increase speed, the income data is the same, more efficient in the use of storage media, reduce redundancy, avoid anomalies (insertion anomalies, deletion anomalies, update anomalies), improved data integrity.

A table saying good (efficient) or if the normal 3 to meet the following criteria:

- If there is decomposition (decomposition) table, it must be guaranteed safe the composition (Lossless-Join Decomposition). That is, after the table is described a new table-table, the table-table can generate a new table with the same exact.
- Maintain dependence on the functional changes in data (Dependency preservation).
- Does not violate Boyce-Codd Normal Form (BCNF).

If the three criteria (BCNF) can't be met, then at least the table does not violate the Normal Form of the third stage (3rd Normal Form / 3NF).

Functional Dependency

Functional Dependency attributes describe the relationship in a relationship. An attribute said functionally dependant on the other, if we use the value attribute to determine the value of the other attributes. Symbol that is used to represent functional dependency. Read the functional set. Notation: $A \rightarrow B$ A and B are attributes of a table. A means of determining the functional B or B depends on A, if and only if there are 2 rows of data with the same value of A, then B is also the same value Notation: $A \times \rightarrow B$

It is the opposite of the previous notation.

Dependency of the table value

Functional Dependency:

- $NRP \rightarrow Nama$
- $Mata_Kuliah, NRP \rightarrow Nilai$

Non Functional Dependency:

- $Mata_Kuliah \rightarrow NRP$
- $NRP \rightarrow Nilai$

Nrp \rightarrow Name

Because for each value Nrp the same, then the value of the same name (Mata_kuliah, NRP) \rightarrow Value Because the value of attributes depending on the NRP and Mata_kuliah together. In another sense Mata_kuliah for the NRP and the same, they also rated the same, because Mata_kuliah and the NRP is a key (is unique).


NRP \nRightarrow Mata_kuliah

Value \nRightarrow NRP

First Normal Form - 1NF

Relations, there is a condition in First Normal Form (1NF) if and only if all domains are covered only atomic value, for example, there is no recurrence group (domains) in a tuple. Advantage of the 1NF relation compared Un normalized (UNRs) is a simplification in the form of representation and ease of use in developing a query language. lack is the need to duplicate data, Most of the system relations (not all) require a form of relations in 1NF.

A table on the form said to be normal if I did not reside in the unnormalized form of a table, where there is a kind of field multiplication and field that allows a null (empty). May not be found:

- 
- Many attributes of value (Multi valued attributes).
 - Attributes composite or a combination of both.
 - Price is the domain attribute must be atomic rates.

Second Normal Form - 2NF

A super key is a set of one or more attribute, which, taken in particular where the possible for us to uniquely identify one entity or relationship.

A Candidate key is a subset of attributes-the attributes that super key is also super key and not reducible to the other super key. A primary key is selected from the set of candidate key for use in relation to an index of the ownership of one or more of the attribute that can be defined from the unique value of one or more of the attribute called functional dependency.

Given a relation (R), a set of (B) is functionally dependent on another set of attributes (A) if, at one time, each value A associated with a value B, this is a form of the FD $A \rightarrow B$ symbol with B. A relation is in second position to its normal form (2NF) if and only if the relationship is also in 1NF and every attribute non key fully dependent on its primary key. 2NF requires that any FD in the relation must include all components of the primary key as the determinant, either direct or transitive.

Why should 2NF, consider the benefits of the 1NF R. paper, pub-id, and editor-created duplicate id. For each author of the paper. If the editor of publications for a paper changes, some also in the tuple must be updated. Finally, if one paper on the take, all the symbol topple should be removed. This form will give the side effects on the deletion of information that a associated with the auth-id name and auth-auth-addr.

- One way to do this is to transform the relations into two or more of the 2NF relations.

example R1: paper-id, id-auth-name \rightarrow Inst, Inst-addr R2: auth-id \rightarrow auth-name, auth-addr R3: paper-pub-id \rightarrow id, editor-id. Normal form 2NF met in a table if it meets the form of 1NF, and all the attributes than the primary key, have a full Functional Dependency on primary key. A table does not meet 2NF, if there are attributes that dependency (Functional Dependency) is only partial (only depending on the part of the primary key).

If there are attributes that have no dependence on the primary key, then the attributes must be moved or removed. Y is full if it is said to delete an Functional dependency X attribute A from X means that Y is no longer dependent functional. Y said if deleting a partial attribute A from X means that Y is functionally dependent. Functional dependency. R depend on the full functional primary key R. Relation scheme R in the form 2NF if every non-primary key attribute A.

Third Normal Form - 3NF

Normal form 3NF fulfilled if the form meets 2NF, and if there are no non-primary key attribute that has a dependence on non-primary key attributes of the other (transitive dependencies).

A relationship in the Third Normal Form (3NF) if and only if the relation is in 2NF and every non key attribute is non transitive dependent on primary key

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form constraint has a stronger form of the Normal third. To be BCNF, relations must be in the form of first normal and forced each of the attributes depends on the function in the super key attributes. In the example below there is a relationship seminar, is the Primary Key NPM + Seminar. Students may take one or two seminars. Each seminar requires 2 each of the students and led by one of the 2 seminar. Each leader can only take one seminar course. NPM in this example and show a seminar guide.

Normal form of the fourth and fifth

Relations in the fourth normal form (NF 4) if the relation in BCNF and dependency tdk contains many values. To remove the dependency of many values from a relation, we divide the relationship into two new relations. Each relation contains two attributes that have a lot of relationship value. Relations in fifth normal form (5NF) deal with the property called the join without any loss of information (lossless join). Fifth normal form (also called the 5 NF PJNF (projection join normal form)). The case is very rare and appears difficult to detect in practice.

Rules of Data Normalization

1. Eliminate Repeating Groups

In the original member list, each member name is followed by any databases that the member has experience with. Some might know many, and others might not know any. To answer the question, "Who knows DB2?" we need to perform an awkward scan of the list looking for references to DB2. This is inefficient and an extremely untidy way to store information. Moving the known databases into a separate table helps a lot. Separating the repeating groups of databases from the member information results in **first normal form**.

The Member ID in the database table matches the primary key in the member table, providing a foreign key for relating the two tables with a join operation. Now we can answer the question by looking in the database table for "DB2" and getting the list of members.

2. **Eliminate Redundant Data**

In the Database Table, the primary key is made up of the MemberID and the DatabaseID. This makes sense for other attributes like "Where Learned" and "Skill Level" attributes, since they will be different for every member/database combination. But the database name depends only on the DatabaseID. The same database name will appear redundantly every time its associated ID appears in the Database Table.

Suppose you want to reclassify a database - give it a different DatabaseID. The change has to be made for every member that lists that database! If you miss some, you'll have several members with the same database under different IDs. This is an update anomaly. Or suppose the last member listing a particular database leaves the group. His records will be removed from the system, and the database will not be stored anywhere! This is a delete anomaly. To avoid these problems, we need **second normal form**.

To achieve this, separate the attributes depending on both parts of the key from those depending only on the DatabaseID. This results in two tables: "Database" which gives the name for each DatabaseID, and "MemberDatabase" which lists the databases for each member. Now we can reclassify a database in a single operation: look up the DatabaseID in the "Database" table and change its name. The result will instantly be available throughout the application.

3. **Eliminate Columns Not Dependent On Key**

The Member table satisfies first normal form - it contains no repeating groups. It satisfies second normal form - since it doesn't have a multi valued key. But the key is MemberID, and the company name and location describe only a company, not a member. To achieve third normal form, they must be moved into a separate table. Since they describe a company, Company Code becomes the key of the new "Company" table.

The motivation for this is the same for second normal form: we want to avoid update and delete anomalies. For example, suppose no members from the IBM were currently stored in the database. With the previous design, there would be no record of its existence, even though 20 past members were from IBM!

4. **BCNF. Boyce-Codd Normal Form**

Boyce-Codd Normal Form states mathematically that:

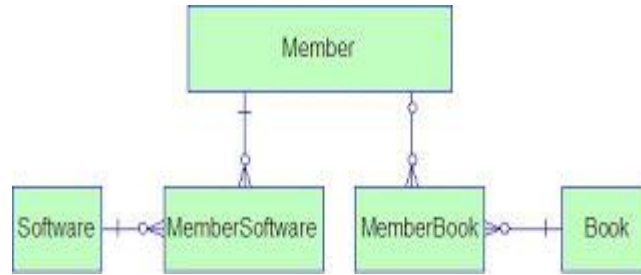
A relation R is said to be in BCNF if whenever $X \rightarrow A$ holds in R, and A is not in X, then X is a candidate key for R. BCNF covers very specific situations where 3NF misses interdependencies between non-key (but candidate key) attributes. Typically, any relation that is in 3NF is also in BCNF. However, a 3NF relation won't be in BCNF if (a) there are multiple candidate keys, (b) the keys are composed of multiple attributes, and (c) there are common attributes between the keys. Basically, a humorous way to remember BCNF is that all functional dependencies are: "The key, the whole key, and nothing but the key, so help me Codd."

5. **Isolate Independent Multiple Relationships**

This applies primarily to key-only associative tables, and appears as a ternary relationship, but has incorrectly merged 2 distinct, independent relationships. The way this situation starts is by a business request list the one shown below. This could be any 2 M:M relationships from a single entity. For instance, a member could know many software tools, and a software tool may be used by many members. Also, a member could have recommended many books, and a book could be recommended by many members.

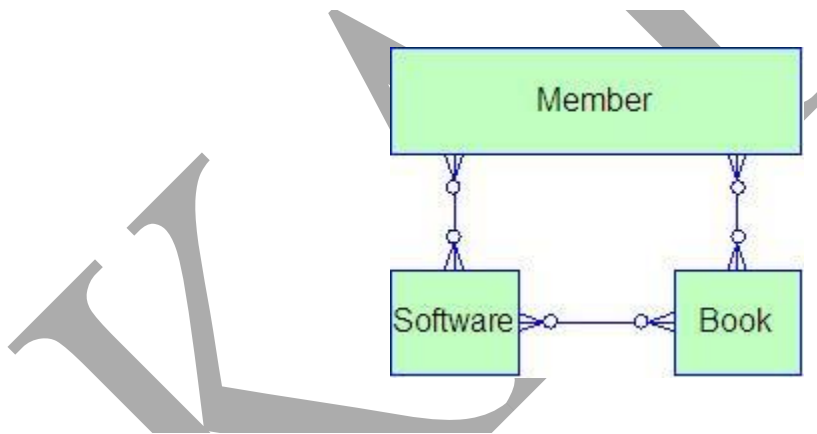
So, to resolve the two M:M relationships, we know that we should resolve them separately, and that would give us 4th normal form. But, if we were to combine them into a single table, it might look right (it is in 3rd normal form) at first. This is shown below, and violates 4th normal form. Get a picture of what is wrong, look at some sample data, shown below. The first few records look right, where Bill knows ERWin and recommends the ERWin Bible for everyone to read. But something is wrong with Mary and Steve.

Mary didn't recommend a book, and Steve doesn't know any software tools. Our solution has forced us to do strange things like create dummy records in both Book and Software to allow the record in the association, since it is key only table. The correct solution, to cause the model to be in 4th normal form, is to ensure that all M:M relationships are resolved independently if they are indeed independent, as shown below.

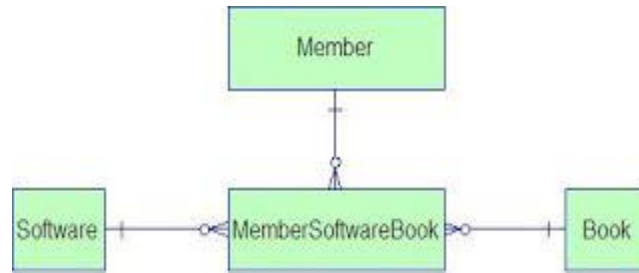


6. Isolate Semantically Related Multiple Relationships

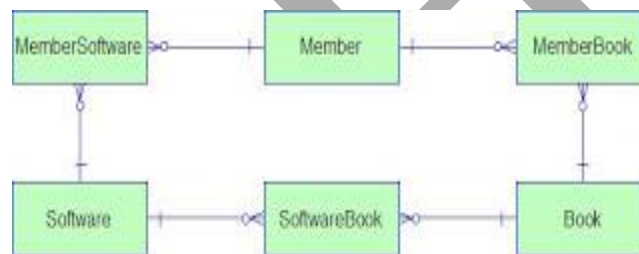
Modify the original business diagram and add a link between the books and the software tools, indicating which books deal with which software tools, as shown below.



This makes sense after the discussion on Rule 4, and again we may be tempted to resolve the multiple M:M relationships into a single association, which would now violate 5th normal form. The ternary association looks identical to the one shown in the 4th normal for example, and is also going to have trouble displaying the information correctly. This time we would have even more trouble because we can't show the relationships between books and software unless we have a member to link to, or we have to add our favorite dummy member record to allow the record in the association table.



The solution, as before, is to ensure that all M:M relationships that are independent are resolved independently, resulting in the model shown below. Now information about members and books, members and software, and books and software are all stored independently, even though they are all very much semantically related. It is very tempting in many situations to combine the multiple M:M relationships because they are so similar. Within complex business discussions, the lines can become blurred and the correct solution not so obvious.



7. Optimal Normal Form

At this point, we have done all we can with Entity-Relationship Diagrams (ERD). Most people will stop here because this is usually pretty good. However, another modeling style called Object Role Modeling (ORM) can display relationships that cannot be expressed in ERD. Therefore there are more normal forms beyond 5th. With Optimal Normal Form (OMF), it is defined as a model limited to only simple (elemental) facts, as expressed in ORM.

8. Domain-Key Normal Form

This level of normalization is simply a model taken to the point where there are no opportunities for modification anomalies. "if every constraint on the relation is a logical consequence of the definition of keys and domains" Constraint "a rule governing static values of attributes" Key "unique identifier of a tuple" Domain "description of an attribute's allowed values"

Trivial functional dependency

A trivial functional dependency is a functional dependency of an attribute on a superset of itself. {Employee ID, Employee Address} \rightarrow {Employee Address} is trivial, as is {Employee Address} \rightarrow {Employee Address}.

Full functional dependency

An attribute is fully functionally dependent on a set of attributes X if it is

→ Functionally dependent on X, and

→ Not functionally dependent on any proper subset of X. {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a *full* functional dependency, because it is also dependent on {Employee ID}.

Transitive dependency

A transitive dependency is an indirect functional dependency, one in which $X \rightarrow Z$ only by virtue of $X \rightarrow Y$ and $Y \rightarrow Z$.

Multi valued dependency

A multi valued dependency is a constraint according to which the presence of certain rows in a table implies the presence of certain other rows.

Join dependency

A table T is subject to a join dependency if T can always be recreated by joining multiple tables each having a subset of the attributes of T .

Super key

A super key is an attribute or set of attributes that uniquely identifies rows within a table; in other words, two distinct rows are always guaranteed to have distinct super keys.

{Employee ID, Employee Address, Skill} would be a super key for the "Employees' Skills" table; {Employee ID, Skill} would also be a super key.

Candidate key

A candidate key is a minimal super key, that is, a super key for which we can say that no proper subset of it is also a super key. {Employee Id, Skill} would be a candidate key for the "Employees' Skills" table.

Non-prime attribute

A non-prime attribute is an attribute that does not occur in any candidate key. Employee Address would be a non-prime attribute in the "Employees' Skills" table.

Primary key

Most DBMSs require a table to be defined as having a single unique key, rather than a number of possible unique keys. A primary key is a key which the database designer has designated for this purpose

Part A (ONE Mark)

Multiple Choice Questions

Online Examination

Part B (2 Marks)

1. What traditional information gathering tools are available for the analyst?
2. What points should be considered in constructing a data dictionary?
3. What basic rules are relevant to construct a DFD?
4. What do you mean by leveling in DFDs? Explain with the help of an example.
5. List and illustrate the primary uses and elements of a decision table.
6. What is meant by structured walk through? Why and how are they conducted?
7. What makes up a system performance definition? Explain the steps to prepare the definition.

Part C (5 Marks)

1. Under what circumstances would the analyst depend more heavily on external than internal information? Why?
2. Distinguish between validity & reliability. How are they related?
3. For what purpose would one use an interview rather than the data collection methods? Explain.
4. "A data dictionary is a structured repository of data about data". Discuss.
5. Discuss the advantages being offered by data dictionary in the documentation area.
6. In what way is a decision tree and DFD related? What about a decision tree and structured English?
7. Who are the important users that must be interviewed? Describe the various steps to be taken to carry out an interview.
8. What considerations are involved in feasibility analysis? Which consideration do you think is the most crucial? Why?

	KARPAGAM ACADEMY OF HIGHER EDUCATION					
	Department of Management					
	Unit 3- System Analysis and Design -Multiple Choice Questions- Each Question Carry ONE Mark					
S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	The problem solving approach usually incorporates the following general steps	Design and implement the 'best solution'.	Analyze and understand the problem.	Identify alternative solutions and decide on a course of action.	Identify the problem.	(IV), (II), (V), (III), (I), (VI)
2	A _____ is a tabular form of presentation that specifies a set of conditions and their corresponding actions.	Decision Table	Decision Tree	Structured English	Data Flow diagram	Decision Table
3	Which of the following tool is not used during system analysis?	Diagram	Structured English	Structured Chart	Decision Tree.	Structured Chart
4	What do you call, "where the Objects hide their inner workings of their operations from the outside world and from other objects"?	Composition	Encapsulation	Generalization	Polymorphism	Encapsulation
5	A data store is represented in data flow diagram as	Rectangle	Square	Open rectangle	Open square	Open rectangle
6	Creating an object model from an existing relational database layout is referred to as	Forward engineering	Backward engineering	Reverse engineering	Top-down engineering	Reverse engineering
7	The symbol represents	Aggregation	Aggregation	Dependency	Composition	Aggregation

8	Whole Part or Composition relationship represents	Aggregation	Inheritance	Polymorphism	Stereotype	Aggregation
9	Place or thing about which some information is gathered?	Entity	Attribute	Data Store	Data flow	Entity
10	Who is not involved in Requirements definition?	Client managers	System end-users	Client engineers	Software developers.	Software developers.
11	Which cohesion operates on the same input or output data?	Communicational	Temporal	Procedural	Functional	Communicational
12	Technical feasibility is an evaluation to determine whether	The system can provide the right information for the organization's personnel	The existing system can be upgraded to use the new technology	Any restructuring of jobs will be acceptable to the current users	The technology needed for the proposed system is available.	The technology needed for the proposed system is available.
13	When requirements go wrong, what could happen from among the following?	The system may cost more than projected	The system may be delivered later than promised	The system will become unreliable and prone to errors	All of the above.	All of the above.
14	Which of the following is a fact-finding technique?	Systems requirement specification	Quality assurance	Sampling of existing documents	Rapid application development	Sampling of existing documents
15	The data dictionary in SDLC contains descriptions of	DFD elements	E-R Diagram	Use case	Class Diagram	DFD elements
16	Cohesion is a qualitative indication of the degree to which a module?	Can be written more compactly	Is connected to other modules and the outside world	Is able to complete its function in a timely manner	Focuses on just one thing	Is connected to other modules and the outside world

17	UML graphical notations can be used not only to describe the system's components but also to describe a model itself; this is known as a	Model	Meta-Model	Stereotypes	Multiplicity.	Meta-Model
18	The requirement definition document is intended for	System end-users	Client managers	Software developers	System architects	System architects
19	Prototyping	Is most practical for large scale projects	Emphasizes getting the design right the first time	Cannot be used when the requirements are not clear.	Involves an iterative and interactive development process with extensive end use involvement	Involves an iterative and interactive development process with extensive end use involvement
20	Which of the following represents the correct sequence of testing activities?	Unit testing, system testing, module testing, integration testing, acceptance testing	Unit testing, volume testing, integration testing, system testing, acceptance testing	Unit testing, integration testing, system testing, module testing, acceptance testing	Unit testing, module testing, integration testing, system testing, acceptance testing	Unit testing, module testing, integration testing, system testing, acceptance testing
21	Which of the following is not a component of CASE Tool?	Diagramming Tools	Code Generators	Information Repository	Debugging Tools	Debugging Tools

22	The process of converting a new or revised system design into an operational one is known as _____.	Testing	Implementation	Quality Assurance	Design	Implementation
23	Which one the following is not a form of Decision Table?	Limited-Entry) Extended-Entry	Mixed-Entry	Double-Entry	Double-Entry
24	Application prototyping follows an organized process or steps that begins with _____.	Use prototype	Review prototype	Develop working model	Identify known requirements	Identify known requirements
25	The requirements model consists of four parts:	Use cases, interface descriptions, class diagram, project scope	Project scope, use cases, class diagram, context diagram	Interface descriptions, data model, context diagram, class diagram	Project scope, use cases, interface descriptions, context diagram	Project scope, use cases, interface descriptions, context diagram
26	_____ are used to group classes together for ease of use, maintainability, and reusability.	Objects	Use	States	Packages	Packages
27	An association must	Be described by nouns	Have attributes	Be described by a verb or nouns	Have a 1:M component	Be described by a verb or nouns

28	A data dictionary	Is a book used by programmers to find the definitions of technical terms	Is the central place where the components of a system are defined	Helps to avoid ambiguities among different development team members	Includes a range of acceptable values for data.	Includes a range of acceptable values for data.
29	Unified modelling language	Is an object oriented programming language	Is useful in describing object oriented design models graphically	Allows to represent multiple views of a system	Is an object oriented system development methodology	Is useful in describing object oriented design models graphically
30	_____ consists of objects with which the user interacts as well as the objects needed to manage.	Business Layer	View Layer	Physical Layer	Network Layer	View Layer
31	The process of looking for patterns to document is called	Pattern	Antipattern	Frameworks	Pattern Template	Frameworks
32	An entity class is a class that	Does not appear in the class diagram	Relates to the class diagram	Exists in the real world and in the class diagram	Exists in the real world but not in the class diagram	Exists in the real world and in the class diagram
33	cases) of the system based on user requirements.	Software Architect	System Analyst	Designer	End-user	System Analyst
34	Design patterns are	Generic problems	Generic solutions recurring problems	Common mistakes	Both (a (b) and e	Both (a (b) and e
35	_____ represent a built-in extensibility mechanism of the UML.	Note	Meta model	Stereotype	Class	Stereotype

36	The most important thing about a class card(CRC card) is that it be	Measured in centimeters	Used in portrait orientation to enhance its capacity	Carefully filled	Used to record responsibilities and collaborations	Used to record responsibilities and collaborations
37	A state machine is	The execution of a particular specified instance	The execution of a particular class of statechart diagram	The execution of a policy by a government behemoth	The execution of a statechart diagram by a specified instance	The execution of a statechart diagram by a specified instance
38	Which of the following statements accurately describe an Information System?	An Information System is an arrangement of people, data processes, information	An Information System is a contemporary term that describes the combination	An Information System is an arrangement of information representation	An Information System is an arrangement of data processes, information	An Information System is an arrangement of people, data processes, information
39	Which of the following skills is not required by a systems analyst?	Programming Language skills	Communication skills	Technical skills	Business process re-engineering skills	Programming Language skills
40	The statements given below are associated with the activity of outsourcing	Ownership of IT assets is not transferred to the outsourcer	Outsourcers do not assume responsibility for an organization's information systems development	Outsourcing is the act of contracting with the outside vendor	Outsourcers do not require a systems analyst.	Outsourcing is the act of contracting with the outside vendor

41	Which of the following is not a step in the linear system development cycle?	Testing design	Prototyping	Requirements definition	Development	Prototyping
42	Given below are some statements associated with the problem definition	definition phase produces a	statement is a document that	definition phase produces a	The problem definition	definition phase
43	Which of the following is not considered as feasibility factor when developing an information system?	Economic	Application	Schedule	Technical	Application
44	The four phases of the Systems Development Life Cycle are _____.	Analysis, gathering, modeling and diagramming	Construction, installation, testing and converting	Designing, charting, formatting and structuring	Planning, analysis, design and implementation	Planning, analysis, design and implementation
45	The _____ is generated by the department or person that has an idea for a new information system.	Feasibility analysis	Gradual refinement	Project sponsor	System request	System request
46	In which phase of the SDLC is the system proposal developed?	Analysis	Design	Implementation	Planning	Analysis
47	The primary advantage of the Waterfall Development methodology is _____.	A version of the system is quickly delivered into the users' hands	Requirements evolve through users' feedback about the system	Features and functionality of the system are explored through simple models	are completely specified and held relatively constant prior to programming	Requirements are completely specified and held relatively constant prior to programming
48	_____ is the process of examining the technical, economic, and organizational pros and cons of developing a new system.	Committee approval	Feasibility analysis	Functionality determination	Risk analysis	Feasibility analysis
49	The functionality of the system or what the information system will do is called the _____ of the system.	Business need	Intangibles	Sponsors	Requirements	Requirements

50	The four elements commonly found on a system request are _____.	Economic, organizational, technical and operational feasibility	Project sponsor, business need, requirements and business value	Risk analysis, familiarity, project size and cost-benefit analysis	Training, software, installation and equipment	Project sponsor, business need, requirements and business value
51 literally means ‘many forms’, the concept that different objects can respond to the same message in different ways.	Composition	Aggregation	Inheritance	Polymorphism	Polymorphism
52	Objects hide their inner workings of their operations from the outside world and from other objects. This is called	Composition	Aggregation	Encapsulation	Polymorphism	Encapsulation
53	Given below are some statements associated with data flow diagrams. Identify the correct statement from among them.	flows in a DFD may be bidirectional	0 DFD only consists of the main process	0 DFD is the same as the context diagram	a DFD must connect to two other processes.	flows in a DFD may be bidirectional
54	Consider the following statements related to UML 2.0.	Composite Structure Diagram, Interaction Overview Diagram and Timing	A composite structure diagram shows the components of a class as a diagram nested inside a large class rectangle.	UML 2.0 takes the interface concept a step further by allowing one to model	(I) and (III) above	(I) and (III) above

55	Consider the following statements in relation to <i>Role names</i> in associations between two classes:	Role names have to be used with association names.	. If there is a relationship between <i>Company</i> and <i>Person</i> then one could use <i>Employee</i> or <i>Employer</i> as role names to convey an employment	The role name is placed on the association near the class that it modifies	Both (II) and (III) above	Both (II) and (III) above
56	The following statements are related to <i>Role/Role names</i> in associations.	When one class associates with another, each one usually plays a role within that association.	A Role name may be placed on one or both ends of an association line.	If <i>Company</i> and <i>Person</i> are two classes in a class diagram that has an association relationship, one could use an association name called “employs	Both (II) and (III) above	Both (II) and (III) above
57	Multiplicity indicators and their meanings.	. 0..* Zero or More	1..* One or More	. 0..1 Zero or One	4..7,9 4,5,6,7 or 9	0..1 Zero or One
58	RAD Stands for	Rapid application development tool	Raid application development tool	Research application Development	Research aid design tool	Rapid application development tool
59	Which of the following are not CASE facilities?	Diagramming tools	Prototyping tools	Quality management tools	System analysing tools.	System analysing tools.

60	The is a group of attributes used to identify a single entity instance.	Concatenated key	Alternate key	Primary key	Candidate key	Concatenated key
----	---	------------------	---------------	-------------	---------------	------------------

[illegible]

[illegible]

UNIT-IV

SYLLABUS

System Implementation: System testing and quality assurance – COCOMO model of testing -The nature of test data - The test plan-Quality Assurance - Role of the data processing auditor - Implementation and software maintenance hardware/software selection.

Systems implementation & evaluation

Systems implementation is the process of:

1. defining how the information system should be built (i.e., physical system design),
2. ensuring that the information system is operational and used,
3. ensuring that the information system meets quality standard (i.e., quality assurance).

Systems design

Conceptual design – what the system should do

Logical design – what the system should look to the user

Physical design – how the system should be built

Physical system design using structured design approach:

→ To produce a system that is easy to read, code, and maintain

1. Factoring: decomposition
2. Span of control: 9 subordinate modules

3. Reasonable size: 50-100 LOC
4. Coupling: minimize inter-module dependency
5. Cohesion: single module functionality
6. Shared use: multiple calls to lower level modules from different bosses

Structured design tools

- Organization of programs and program modules (structure chart)
- Processing logic specification in each module (pseudocode)

Structure chart – to show graphically

1. how the various program parts/modules of an information system are physically organized hierarchically
2. how the modules communicate with each other through data couple (data exchange) and flag (control/message)
3. how the modules are related to each other in terms of sequence, selection, and repetition

Symbols

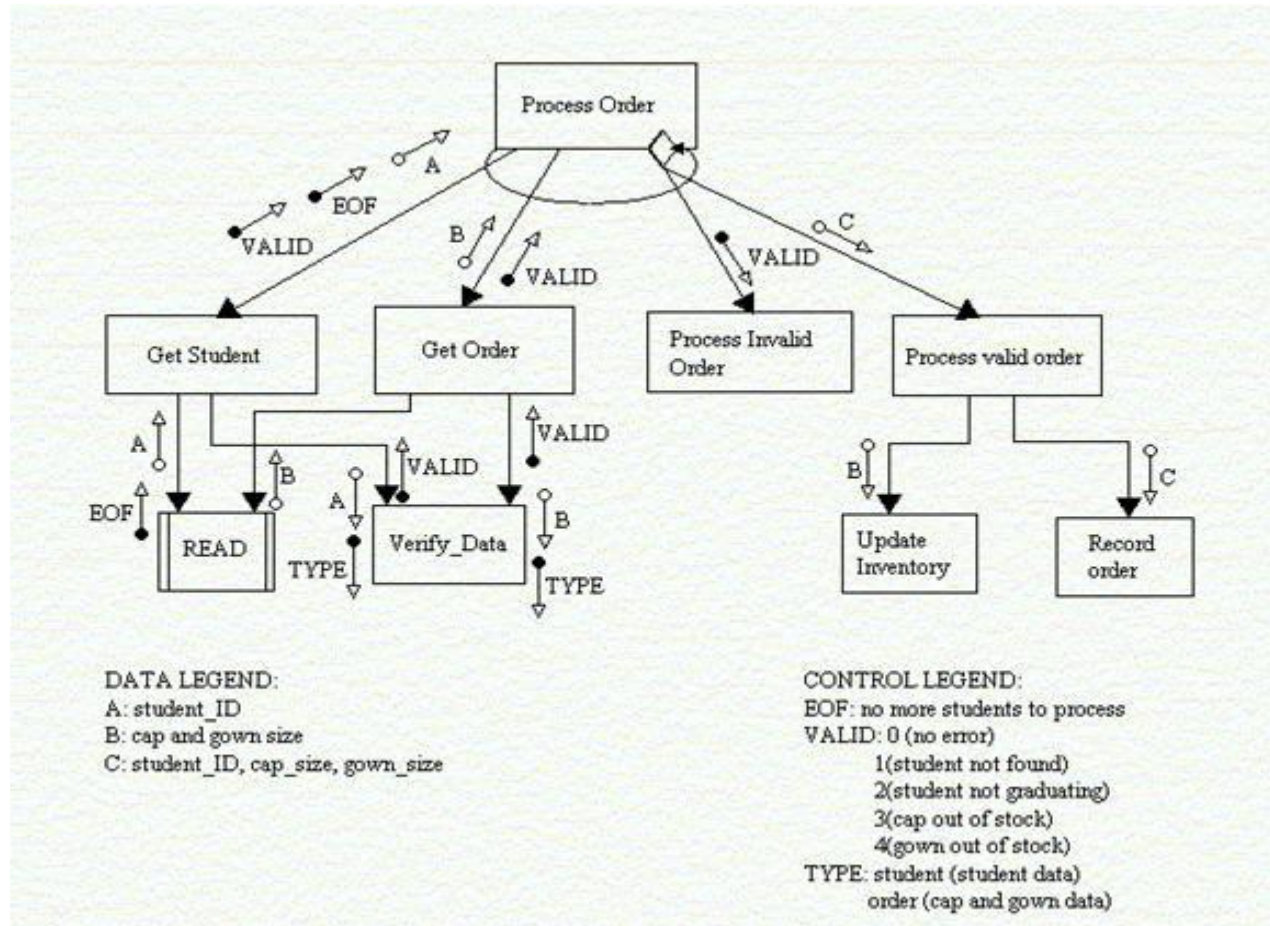
Components	Symbol
Module	Rectangle
Data couple	Clear circle without arrow
Control flag	Filled circle without arrow
Conditional processing/selection	Diamond
Repetition	A curved line intersecting the connection to the modules
Predefined module	Rectangle with a vertical bar on each side

Converting DFDs to structure charts

1. Locate the central transform/transaction center
2. Find a coordinating module for the top of the chart
3. Identify the primary input and output data flows
4. Draw a top-level chart (consists of two hierarchical level)
5. Refine the chart until the data origin, system function, and output dispositions are defined

Cap and gown ordering system example

Step	Example
Center	Process #3: Validate Order
Root module	Process Order
Primary inputs	Student ID, cap size, gown size
Primary outputs	Receipt, Order detail, Inventory update
Top-level	Get student, Get order, Process invalid order, Process valid order
Refinement	Get student, Get order, Process valid order



Systems Testing and Quality Assurance

Overview

It should be a goal of every system development project to deliver a system free of defects, on time and in a cost-effective manner. This course is designed to give students the practical skills and the specific techniques to build the zero defect concept into their methods of building and maintaining systems. The primary emphasis of the workshop is to avoid errors rather than do re-work to correct them. Our approach is straightforward.

We believe there is a direct and demonstrable relationship between the quality of the requirements analysis, the quality of the test plans that result, and the quality of the operational system.

Here are some of the benefits your organization will receive: slashed time and cost of re-work and systems maintenance, systems built more quickly, increased user confidence in system quality, and reduced or eliminated errors in operational systems.

Objectives

Upon completion of this workshop, students will be able:

- Incorporate the Zero Defect concept into their methods of building and maintaining systems
- Develop an integrated design, construction and test plan
- Develop practices that prevent errors from occurring prior to system testing
- Verify & validate the quality of the system during each phase of a system's life cycle
- Conduct inspections and structured walkthroughs
- Explain the role of smoke testing
- Document and use test results (metrics) to pinpoint functions of the systems development process that need improvement
- Design effective test cases using white-box and black-box techniques
- Use various systems tests: stress, volume, regression, etc.
- Create test objectives and acceptance criteria

Testing should involve a thorough auditing to introduce control elements unique to a system. The Data Processing (DP) Auditor should be involved in testing. The role of the auditor is to judge the controls and make any recommendations to change or enhance the project. The user department also has to be involved in this process. For testing programs, test data that violate control procedures and test data that do not violate control procedures are to be included. At the time of testing all the required files are collected for testing.

Audit trail:

One important function of control is to provide audit trail. Some of the problems encountered by the auditor are as follows

- Once the source documents are transferred into a machine-readable form, they are filed in areas that make it difficult to access.
- Files stored on tapes or disks require a computer to read it.
- Processing activities are difficult to observe since they are inside the system.

To solve this detailed file of the transactions can be kept and they can be used as an input for an audit program. The audit program then selects a transaction for selected accounts for the auditor to view. Three important steps are to be considered while evaluating the project

1. Define the control objectives as separate design and test requirements.
2. Re examine budget costs
3. Review specification

COCOMO Model

In this section the model of COCOMO I also called COCOMO'81 is presented. The underlying software life cycle is a waterfall life cycle. Detailed information about the ratings as well as the cost drivers can be found in [Boehm81].

Boehm proposed three levels of the model: basic, intermediate, detailed.

The basic COCOMO'81 model is a single-valued, static model that computes software development effort (and cost) as a function of program size expressed in estimated thousand delivered source instructions (KDSI).

The intermediate COCOMO'81 model computes software development effort as a function of program size and a set of fifteen "cost drivers" that include subjective assessments of product, hardware, personnel, and project attributes.

The advanced or detailed COCOMO'81 model incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

ADVANCED, DETAILED COCOMO

The Advanced COCOMO model computes effort as a function of program size and a set of cost drivers weighted according to each phase of the software lifecycle. The Advanced model applies the Intermediate model at the component level, and then a phase-based approach is used to consolidate the estimate [Fenton, 1997].

The four phases used in the detailed COCOMO model are:

- requirements planning and product
- design (RPD), detailed design (DD),
- code and unit test (CUT), and integration and test (IT).

ADA COCOMO

COCOMO has been continued to evolve and improve since its introduction. Beneath the "ADA COCOMO" model:

The model named "ADA_87" assumes that the ADA programming language is being used. The use of ADA made it easier to develop highly reliable systems and to manage complex applications. The APM_88 model is based upon a different "process model" called the ADA Process Model. Among the assumptions are:

- ADA is used to produce compiler checked package specifications by the Product Design Review (PDR)

- Small design teams are used
- More effort is spent in requirements analysis and design
- Less effort is spent in coding, integration, and testing
- The APM_88 model incorporates changes to the equations, several cost drivers, and a variety of other tables. For further information please contact [Boehm 81].

ADVANTAGES OF COCOMO'81

- COCOMO is transparent; one can see how it works unlike other models such as SLIM
- Drivers are particularly helpful to the estimator to understand the impact of different factors that affect
- project costs

DRAWBACKS OF COCOMO'81

- It is hard to accurately estimate KDSI early on in the project, when most effort estimates are required
- KDSI, actually, is not a size measure it is a length measure
- Extremely vulnerable to mis-classification of the development mode
- Success depends largely on tuning the model to the needs of the organization, using historical data which is not always available

The Nature of test data

The proper choice of test data is as important as the test itself. If test data as input are not valid or representative of the data to be provided by the user, then the reliability of the output is suspect.

Test data may be artificial (created solely for test purposes) or live (taken from the user's actual files). Properly created artificial data should provide all combinations of values and formats and make it possible to test all logic and transaction path subroutines. Unlike live data, which are biased toward typical values, artificial data provide extreme values for testing the limits of the candidate system:

For large, complex systems, a computer program is used to generate necessary test data. Data-generating programs save substantial time for both the programmer and the test itself. A familiarity with system files and parameters, however, is necessary for writing an effective data-genera program.

The test plan

A **TEST PLAN** is a document describing software testing scope and activities. It is the basis for formally testing any software/product in a project.

ISTQB Definition

- **test plan:** A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.
- **master test plan:** A test plan that typically addresses multiple test levels.
- **phase test plan:** A test plan that typically addresses one test phase.

Test Plan Types

One can have the following types of test plans:

- **Master Test Plan:** A single high-level test plan for a project/product that unifies all other test plans.
- **Testing Level Specific Test Plans:** Plans for each level of testing.
 - Unit Test Plan
 - Integration Test Plan
 - System Test Plan
 - Acceptance Test Plan
- **Testing Type Specific Test Plans:** Plans for major types of testing like Performance Test Plan and Security Test Plan.

Test Plan Guidelines

- Make the plan concise. Avoid redundancy and superfluousness. If you think you do not need a section that has been mentioned in the template above, go ahead and delete that section in your test plan.
- Be specific. For example, when you specify an operating system as a property of a test environment, mention the OS Edition/Version as well, not just the OS Name.
- Make use of lists and tables wherever possible. Avoid lengthy paragraphs.
- Have the test plan reviewed a number of times prior to baselining it or sending it for approval. The quality of your test plan speaks volumes about the quality of the testing you or your team is going to perform.
- Update the plan as and when necessary. An out-dated and unused document stinks and is worse than not having the document in the first place.

Quality Assurance

QA is a way of preventing mistakes and defects in manufactured products and avoiding problems when delivering solutions or services to customers; which ISO 9000 defines as "part of quality management focused on providing confidence that quality requirements will be fulfilled". This defect prevention in quality assurance differs subtly from defect detection and rejection in quality control, and has been referred to as a *shift left* as it focuses on quality earlier in the process i.e. to the left of a linear process diagram reading left to right.

The terms "quality assurance" and "quality control" are often used interchangeably to refer to ways of ensuring the quality of a service or product.^[3] For instance, the term "assurance" is often used as follows: *Implementation of inspection and structured testing as a measure of quality assurance in a television set software project at Philips Semiconductors is described.* The term "control", however, is used to describe the fifth phase of the Define, Measure, Analyze, Improve, Control (DMAIC) model. DMAIC is a data-driven quality strategy used to *improve* processes.

Quality assurance comprises administrative and procedural activities implemented in a quality system so that requirements and goals for a product, service or activity will be fulfilled. It is the systematic measurement, comparison with a standard, monitoring of processes and an associated feedback loop that confers error prevention. This can be contrasted with quality control, which is focused on process output.

Quality assurance includes two principles: "Fit for purpose" (the product should be suitable for the intended purpose); and "right first time" (mistakes should be eliminated). QA includes management of the quality of raw materials, assemblies, products and components, services related to production, and management, production and inspection processes. The two principles also manifest before the background of developing (engineering) a novel technical product: The task of engineering is to make it work once, while the task of quality assurance is to make it work all the time.

Historically, defining what suitable product or service quality means has been a more difficult process, determined in many ways, from the subjective user-based approach that contains "the different weights that individuals normally attach to quality characteristics," to the value-based approach which finds consumers linking quality to price and making overall conclusions of quality based on such a relationship.

Role of the data processing auditor

System Audit

It is an investigation to review the performance of an operational system. The objectives of conducting a system audit are as follows –

- To compare actual and planned performance.
- To verify that the stated objectives of system are still valid in current environment.
- To evaluate the achievement of stated objectives.
- To ensure the reliability of computer based financial and other information.
- To ensure all records included while processing.
- To ensure protection from frauds.

Audit of Computer System Usage

Data processing auditors audits the usage of computer system in order to control it. The auditor need control data which is obtained by computer system itself.

System Auditor

The role of auditor begins at the initial stage of system development so that resulting system is secure. It describes an idea of utilization of system that can be recorded which helps in load planning and deciding on hardware and software specifications. It gives an indication of wise use of the computer system and possible misuse of the system.

Audit Trial

An audit trial or audit log is a security record which is comprised of who has accessed a computer system and what operations are performed during a given period of time. Audit trials are used to do detailed tracing of how data on the system has changed.

It provides documentary evidence of various control techniques that a transaction is subject to during its processing. Audit trials do not exist independently. They are carried out as a part of accounting for recovering lost transactions.

Audit Methods

Auditing can be done in two different ways –

Auditing around the Computer

- Take sample inputs and manually apply processing rules.
- Compare outputs with computer outputs.

Auditing through the Computer

- Establish audit trail which allows examining selected intermediate results.
- Control totals provide intermediate checks.

Audit Considerations

Audit considerations examine the results of the analysis by using both the narratives and models to identify the problems caused due to misplaced functions, split processes or functions, broken data flows, missing data, redundant or incomplete processing, and non addressed automation opportunities.

The activities under this phase are as follows –

- Identification of the current environment problems
- Identification of problem causes
- Identification of alternative solutions
- Evaluation and feasibility analysis of each solution

- Selection and recommendation of most practical and appropriate solution
- Project cost estimation and cost benefit analysis

Security

System security refers to protecting the system from theft, unauthorized access and modifications, and accidental or unintentional damage. In computerized systems, security involves protecting all the parts of computer system which includes data, software, and hardware. Systems security includes system privacy and system integrity.

- **System privacy** deals with protecting individuals systems from being accessed and used without the permission/knowledge of the concerned individuals.
- **System integrity** is concerned with the quality and reliability of raw as well as processed data in the system.

Control Measures

There are varieties of control measures which can be broadly classified as follows –

Backup

- Regular backup of databases daily/weekly depending on the time criticality and size.
- Incremental back up at shorter intervals.
- Backup copies kept in safe remote location particularly necessary for disaster recovery.
- Duplicate systems run and all transactions mirrored if it is a very critical system and cannot tolerate any disruption before storing in disk.

Physical Access Control to Facilities

- Physical locks and Biometric authentication. For example, finger print
- ID cards or entry passes being checked by security staff.
- Identification of all persons who read or modify data and logging it in a file.

Using Logical or Software Control

- Password system.
- Encrypting sensitive data/programs.
- Training employees on data care/handling and security.
- Antivirus software and Firewall protection while connected to internet.

Risk Analysis

A risk is the possibility of losing something of value. Risk analysis starts with planning for secure system by identifying the vulnerability of system and impact of this. The plan is then made to manage the risk and cope with disaster. It is done to access the probability of possible disaster and their cost.

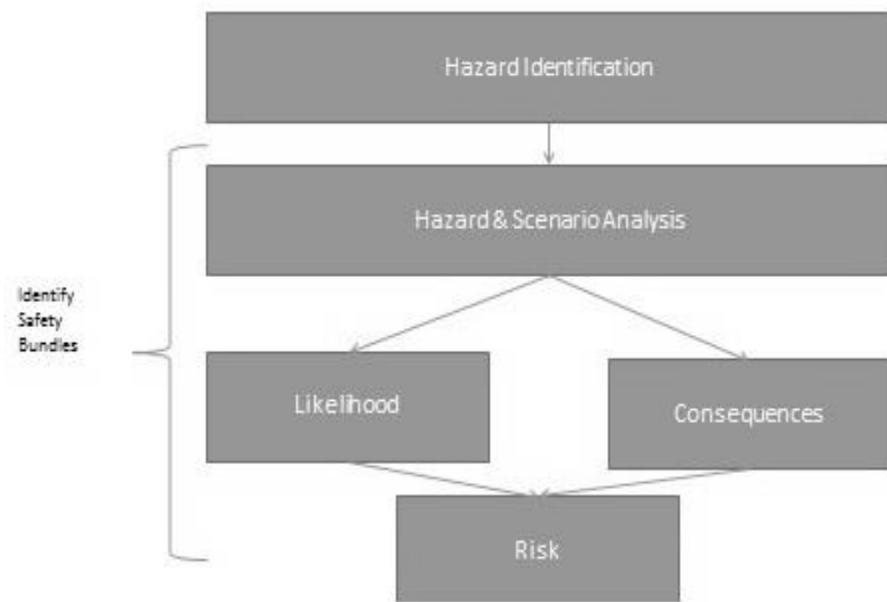
Risk analysis is a team work of experts with different backgrounds like chemicals, human error, and process equipment.

The following steps are to be followed while conducting risk analysis –

- Identification of all the components of computer system.
- Identification of all the threats and hazards that each of the components faces.
- Quantify risks i.e. assessment of loss in the case threats become reality.

Risk Analysis – Main Steps

As the risks or threats are changing and the potential loss are also changing, management of risk should be performed on periodic basis by senior managers.



Risk management is a continuous process and it involves the following steps –

- Identification of security measures.
- Calculation of the cost of implementation of security measures.
- Comparison of the cost of security measures with the loss and probability of threats.
- Selection and implementation of security measures.
- Review of the implementation of security measures.

Software Implementation and Maintenance:

Software Implementation and Maintenance Implementation means the process of converting a new or a revised system design into an operational one. The other aspects are the post-implementation review and software maintenance. There are three types of implementation: Implementation of a computer system to replace a manual system.

Implementation of a new computer system to replace an existing one. Implementation of a modified application to replace an existing one.

Conversion: The objective is to put the tested system into operation. Conversion should be exciting because it is the last step before system begins to show results. If not, properly planned, then, things can go wrong? For many first-time users, this theme is familiar? What went wrong? Conversions are often fiasco?

Activities for Conversion:

An activity for Conversion Plan Testing System Conversion begins with a review of project plan and system test documentation and implementation plan. The parties involved are: - User - Project team - Programmers - Data capture – form filling and checking

Files conversion: File conversion involves capturing data and creating computerized files from existing files. Data entry staff training Database is prime concern. Creating test files: programs should be checked on test data files.

Train Personnel: Training aids User manual Screen menu Data dictionary Job aids Wall charts. Conversion of physical Facilities: Communication network Hardware

Conversion of Administrative procedures

Resistance to change: People will resist changes, how they will react: Hostility: Non-cooperation, fear Withdrawal from the cause of stress-supervisor going sick Refusing to accept the computer Adverse effect on employee's status, job satisfaction May even sabotage the system Unable to understand the new system- communication gap.

There are several ways to reduce resistance to system change Identify and discuss the defects in the present system. Convince them that changeover will improve the quality of work and also it will help them all. Establish open communication between user and project team. Invite and use employee participation in all phases of conversion process.

Different Ways of Conversion: Parallel Running: new and old system Direct conversion: Totally change-over i.e. only new system Gradual switch over: Partial conversion: step by step conversion

Main activities: Programming Test data preparation and system testing Data collection/ data capture : i) filling of forms ii) Data entry Training of users Conversion: Change-over methods – direct, gradual, parallel.

Post Implementation Review:

Post Implementation Review Operational systems are quickly taken for granted. Every system requires periodic evaluation after implementation. A post-implementation review measures the system's performance against predefined requirements. A post-implementation review determines, how well the system continues to meet performance specifications. It also provides information to determine whether major redesign is necessary.

How well system meets the stated requirements? Whether major redesign is necessary? Some modification (minor) is needed? Actual projects costs exceed initial estimates. Actual project benefits/initial estimates Major problems surfaced during conversion (review).

Activities for Post-Implementation review: Request for review plan: formal review plan Objectives of review Type of evaluation Time schedule

Administrative Plan: Area objectives, costs, performance, benefits Personnel requirements plan: review, performance objectives and training performance to data – personnel performance, training performance Hardware plan Documentation review Plan: Instructions for filling the forms, checking the forms, data entry/menu driven; Manuals- flow charts/charts, dos and don'ts; evaluate the accuracy and completeness of documentation compiled to data, documentation standards

Software Maintenance: Maintenance is important part of development. Programmers spend more time maintaining program than they do writing them. Software maintenance is required because:

Software is a product designed in an adhoc fashion with few standards poorly documented, difficult to maintain Maintenance is not as rewarding as developing system. It is assumed that maintenance requires no skill or experience. Users are not fully aware of maintenance problems or its high cost. Few tools/techniques are available. A good test plan is lacking Standards, procedures and guidelines are poorly defined and enforced Mostly delegated to junior programmer Mostly maintain without care for structure/documentation Programmers expect they will not be there when the implementation takes place.

Maintenance/Enhancement: Maintenance can be classified as **Corrective maintenance:** means repairing processing/ performance failures, making changes because of previously uncorrected problems or false assumptions. **Adaptive maintenance:** means changing the program functions and changes in hardware and software environment.

Perfective maintenance: means enhancing the performance/modifying the program to respond to user's additional or changing needs. Enhancement of these, more time and money is spent on perfective maintenance than corrective/adaptive. Maintenance covers a wide range of activities including: Correcting coding and design errors. Updating documentation/test data. Upgrading user support. Enhancement- adding, modifying or redeveloping the code to support changes.

It is necessary to keep up with changing user needs and operational environment. Labor-intensive nature. Reducing maintenance costs- maintenance management audit, software system audit, software modification. Proper Maintenance Plan: makes the software more reliable, improved response time in correcting errors, improved user satisfaction, higher morale among maintenance staff.

To put maintenance in its proper perspective requires considerable skill and experience, and in an important and ongoing aspect of system development. Maintenance demands more orientation and training than any other programming activities. The environment must recognize the needs of the maintenance programmer for tools, methods and training.

The systems come with hardware, software and support. Today, selecting a system is a serious and time-consuming business.

There are several factors to consider prior to system selection:

1. Define the system capabilities that make sense for business. Computers have proven valuable to business in the following areas:

- Cost reduction includes reduction of the inventory, savings on space and improved ability to predict business trends
- Cost avoidance includes early detection of problems and ability to expand operations without adding clerical help.

- Improved service emphasizes quick availability of information to customers, improved accuracy and fast turnaround
- Improved profit reflects the bottom line of the business and its ability to keep receivables within reason.

1. Specify the magnitude of the problem, that is, clarify whether selections consist of a few peripherals or major decision concerning the mainframes.
2. Assess the competence of the in-house staff. This involves determining the expertise needed in areas such as telecommunications and data base design. Acquiring a computer often results in securing temporary help for conversion. Planning for this help is extremely important.
3. Consider hardware and software as a package. This approach ensures compatibility. Infact, software should be considered first, because often the user secures the hardware and then wonders what software is available for it.
4. Develop a schedule for the selection process. Maintaining a schedule helps keeps the project under control.
5. Provide user indoctrination. This is crucial, especially for first-time users. Selling the system to the user staff, providing adequate training, and preparing an environment a conducive to implementation are pre- requisites for system acquisition.

Major phases in selection

The selection process should be viewed as a project, and a project team should be organized with management support. In larger projects, the team includes one or more user representatives, an analyst and EDP auditor, and a consultant. Several steps make up the selection process.

1. Requirements analysis
2. System specifications
3. Request for proposal (RFP)
4. Evaluation and validation
5. Vendor selection
6. Post-installation review

Requirements analysis

The first step in selection understands the user's requirements within the framework of the organization's objectives and the environment in which the system is being installed. Consideration is given to the user's resources as well as to finances.

In selecting software, the user must decide whether to develop it in house, hire a software company or contract programmer to create it, or simply acquire it from a software house. The choice is logically made after the user has clearly defined the requirements expected of the software. Therefore, requirements analysis sets the tone for software selection.

System Specifications

Failure to specify system requirements before the final selection almost always results in a faulty acquisition. The specifications should delineate the user's requirements and allow room for bids from various vendors. They must reflect the actual applications to be handled by the system and include system objectives, flowcharts, input-output requirements, file structure and cost. The specifications must also describe each aspect of the system clearly, consistently and completely.

Request for Proposal

After the requirements analysis and system specifications have been determined, a request for proposal is drafted and sent to selected vendors for bidding. Bids submitted are based on discussions with vendors. At a minimum, the RFP should include the following

1. Complete statement of the system specifications, programming language, price range, terms and time frame.
2. Request for vendor's responsibilities for conversion, training and maintenance
3. Warranties and terms of license or contractual limitations.
4. Request for financial statement of vendor
5. Size of staff available for system support

Evaluation and validation

The evaluation phase ranks vendor proposals and determines the best suited to the user's needs. It looks into items such as price, availability and technical support. System validation ensures that the vendor can match his/her claims, system performance. True validation is obtained verified by having each system demonstrated. An outside consultant can be employed for consulting purpose.

Vendor selection

This step determines the winner – the vendor with the best combination of reputation, reliability, service record, training, delivery time, lease finance terms and conversion schedule. Initially a decision is made which vendor to contact. The sources available to check on vendors include the following

1. Users
2. Software houses
3. Trade associations
4. Universities
5. Publications/Journals
6. Vendor software lists
7. Vendor referral directories
8. Published directories
9. Consultants
10. Industry contacts

Post- installation Review

Sometime after the package is installed, a system evaluation is made to determine how closely the new system conforms to plan. System specifications and user requirements are audited to pinpoint and correct any differences

Software selection:

Software selection is a critical aspect for system development. There are 2 ways of acquiring the software.

- Custom -made
- Packages

Criteria for Software selection:

Reliability – It is the probability that the software will executed in a specific period of time without any failures. It is important to the professional user. It brings up the concept of modularity, or the ease which a package can be modified.

Functionality – It is the definition of the facilities, performance and other factors that the user requires in the finished product.

Capacity – Capacity refers to the capability of the software package to handle the users requirements for size of files, number of data elements, and reports. All limitations should be checked.

Flexibility – It is a measure of effort required to modify an operational program. One feature of flexibility is adaptability.

Usability – This criteria refers to the effort required to operate, prepare the input, and interpret the output of a program. Additional points considered here are portability and understandability. Portability refers to the ability of the software to be used. Understandability is the purpose of the product.

Security – It is a measure of the likelihood that a system's user can accidentally or intentionally access or destroy unauthorized data.

Performance – It is a measure of the capacity of the software package to do what it is expected to do. This criteria focuses on throughput or how effectively a package performs under peak load.

Serviceability –This criteria focuses on documentation and vendor support.

Ownership – Who owns the software and to consider whether he has the right to access the software, or he can sell or modify the software.

Minimal costs – Cost is a major consideration in deciding between in-house and vendor software.

Evaluation Process:

There are three process for evaluating hardware and software.

1. Benchmark programs: It is a sample program for evaluating different computers and their software. It is necessary because computers often use the same instructions, words of memory or machine cycle to solve a problem. Benchmarking includes the following

- Determination of the minimum hardware.
- An acceptance test
- Testing in an ideal environment to determine the timings and in the normal environment to determine its influence on other programs.

2. Experience of other users: Benchmarking only validates vendors' claims. Experience of other users with the same system software is essential.

3. Product reference manuals: These evaluate a system's capability. These reports elaborate on computer products, services and prices.

Evaluation of proposals:

After all proposals are evaluated, the final vendor is selected using any of the 3 methods

1. adhoc refers to the user's inclination to favor one vendor over others.

2. Scoring. In this method the characteristics of each system are listed and score is given in relation to the maximum point rating. Then each proposal is rated according to its characteristics.

3. Cost value approach. In this method a dollar credit method is applied to the proposal that meets the user's desirable characteristics. This credit is subtracted from the vendor's quoted price. The proposal with the lowest price is selected.

Performance evaluation:

Hardware selection requires an analysis on the following criteria

1. System availability
2. Compatibility
3. Cost
4. Performance
5. Uptime
6. Support
7. Usability

For the software evaluation, the following are considered

1. The programming language and its suitability to the applications
2. Ease of installation and training
3. Extent of enhancements to be made prior to installation.

In addition to hardware and software evaluation, the quality of the vendor's should be examined.

Considerations to ensure vendor quality are as follows

1. Backup
2. Conversion
3. Maintenance
4. System development

Part A (ONE Mark)

Multiple Choice Questions

Online Examination

Part B (2 Marks)

1. What makes up a feasibility report? How would you change it?
2. Define and explain then procedure for cost/benefit determination.
3. What design specifications are considered in preparing a test plan?
4. What is syntax error? How does it differ from a logic error? Give an example.
5. What are the pros and cons of Payback method, cash-flow analysis and break-even analysis?

Part C (5 Marks)

1. How important is a project team in a feasibility analysis? Is it mandatory in every study? Where are the exceptions?
2. What cost elements are considered in cost/benefit analysis? Which element do you think is the most difficult to estimate.
3. Discuss various cost estimation techniques.
4. How do NPV and present value analyses differ? Illustrate.
5. Why do we test systems? How important is testing? Discuss.
6. Discuss the various types of testing in detail.
7. Outline the various activities that represent a test plan.
8. There are two ways of debugging program software: bottom-up and top-down. How do they differ?
9. Elaborate on the steps taken in system testing that lead to the user's acceptance of the system.
10. Explain the differences between White-box and Black-box testing.

	KARPAGAM ACADEMY OF HIGHER EDUCATION					
	Department of Management					
	Unit 4- System Analysis and Design -Multiple Choice Questions- Each Question Carry ONE Mark					
S. No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	Traditionally, the only phase of software development where a formal approach is used is	Programming	Design	Requirements	Planning	Programming
2	Which of the following statements is/are true?	There is no restriction on multiple associations between the same two classes	There can be multiple associations between the same two classes, but they should represent different roles	Multiple associations between the same two classes is not allowed	Multiple associations between the same two classes must be aggregated to one	There can be multiple associations between the same two classes, but they should represent different roles
3	Which of the following statements accurately describes an Information System?	An Information System is an arrangement of people, data processes, information representation and information technology	An Information System is a contemporary term that describes the combination of computer technolog	An Information System is an arrangement of information representation and information technology that interacts t	An Information System is an arrangement of data processes, information representation and information technology	An Information System is an arrangement of people, data processes, information representation and information technology
4	Which of the following is not a step in the linear system development cycle?	Testing design	Prototyping	Requirements definition	Development	Prototyping

5	Which of the following statement is true?	The problem definition phase produces a document written using technical terminology of the system analyst	The problem statement is a document that contains the problems faced by the organization	The problem definition phase produces a document that is a broad statement of user requirements	The problem definition phase does not specify the resources allocated to the project	The problem definition phase produces a document that is a broad statement of user requirements
6	Which of the following is not considered during the cost-benefit analysis of an information system development project?	Personnel costs	Computer usage	Training costs	Clients staff costs	Clients staff costs
7	Which of the following is not a fact finding method?	Site visits	Prototyping	Study of similar systems	Business analysis	Business analysis
8	An initial attempt at defining the pieces/parts of the system and their relationships, organizing these pieces/parts are called	Use-case analysis	Architectural analysis	Structural analysis	Dependency analysis	Architectural analysis
9	What does an actor represents I in use case model?	A role that a human, hardware	can perform several acts	regardless of its role	or a hardware device together	human, hardware device,
10	A subclass inherits is 'parents' is	Attributes, links	Attributes, operations	Attributes, operations, relationships	Relationships, operations, links	Attributes, operations, relationships
14	Polymorphism can be described as	Hiding many different implementations behind one interface	Inheritance	Aggregation and association	Generalization	Hiding many different implementations behind one interface
15	Which of the following phrase best represents a generalization relationship?	"Is a part of"	"Is a kind of"	"Is a replica of"	"Is composed of"	"Is a kind of"

16	Business object models describe	The structure of the business	elements are used to fulfill the	of the business and how those	that the organization	structure of the business and
17	Business use-cases and actors together describe	The static elements of the work in progress	The dynamic elements of the work in progress	The logical view of the work in progress	The business processes that the organization supports	The business processes that the organization supports
18	What is the other name for Encapsulation?	Information hiding	Interface management	Polymorphism	Aggregation	Information hiding
19	When identifying the major Use Cases, the information needed to identify the Use Cases is contained in the _____.	Internal activity	External entity	user case information	hardware	user case information
20 Systems are prone to errors	data entry	control	information	system	data entry
21	Elements of user training	experience	motivation	success	failer	experience
22	The initial training period was	key development	key functions	key systems	key duration	key functions
23	Training aids	five aids	six aids	seven aids	four aids	four aids
24	Forms and displays conversion is	old forms	new forms	several forms	zero	old forms
25	Conversion of physical facilities	systems design	systems test	Systems development life cycle method	all the above	systems design
26relationship specifies an optional behavior.	An inheritance	Fertilization	important	all the above	An inheritance
27	System testing makes a	logical assumption	test data	output data	input data	logical assumption
28	Testing is vital to the	failure	non success	success	none	success
29	The nature of the test data is an	user data	suspect	important	all the above	important
30	In the volumn test they create a hardware and software of	functions	records	files	(a)&(b)	records
31	Usability documentation and procedure are	user-friendly	user-request	user-data	(a)&(b)	user-friendly
32	Recovery and security	file integrity	documentation	procedure	none	file integrity

33	Online response is to	online development	online systems	online representative	all the above	online systems
34	Activity network for system testing are involved	eight points	four points	six points	one points	eight points
35	Prepare test data for program testing	test data	test system	test points	test program	test data
36	The purpose of system testing is to identify and correct errors in the	candidate system	systems test	system documentation	string test	candidate system
37	Types of systems tests are	Program modules should be loosely coupled.	qualifications	human factor	bundled	bundled
38	String testing is	total system	acceptance	functions	(a)&(b)	total system
39	User acceptance testing is	fertility	reliability	integrity	none	reliability
40	System documentation	reference	files check out	searching	(a)&(b)	reference
41 defines the objectives of project and reviews the overall activities	Quality goals	Quality Assurance	Quality development	(a)&(b)	Quality Assurance
42	Quality factors specifications involved in points	nine	twelve	six points	five	twelve
43	Levels of quality assurance are	three	four	six points	eight	three
44	Trends in testing are	development	growth	success	all the above	development
45	An important function of control system is	Audit trail	role of audit	processing audit	all the above	Audit trail
46	How data store is represented in a data flow diagram?	Circle	Full Open rectangle	Half open rectangle	Full open triangle	Half open rectangle
47	The is a construct that helps analysts to work with users to	Not use case	Use case	user case information	all the above	Use case
48relationship specifies an optional behavior	specification	A generalization	A fertilization	(a)&(b)	A generalization

49	When identifying the major Use Cases, the information needed to identify the Use Cases is contained in the _____.	specification	External entity	original form	Observation form	External entity
50	When developing use cases, the project team first identifies the _____.	Triggering event that causes the use case to occur	use case begins	where the use case occurs	Managers that supervise the use case department	Triggering event that causes the use case to occur
51	Outputs from a use case are described on the use case form along with their corresponding _____.	models	output	input	all the above	output
52	Each use case describes how the system reacts to a(n) _____ that occurs to trigger the system.	data entry	Data store	data process	data collection	Data store
53	The four elements commonly found on a system request are _____.	Economic, organizational, technical and operational feasibility	Project sponsor, business need, requirements and business value	Risk analysis, familiarity, project size and cost-benefit analysis	Training, software, installation and equipment	Economic, organizational, technical and operational feasibility
54	Conversion process are	displays	physical facilities	procedure	all the above	physical facilities
55	Combating resistance to change	power house	post customer	college graduate	none	power house
56	Hardware suppliers are	group	several	few	none	group
57	Which of the following information systems are aimed at improving the routine business activities on which all organizations depend?	Management information systems	Decision support systems	Transaction processing systems	Management support systems	Transaction processing systems
58	Which of the following strategies are adopted if information requirements are not well defined?	Rapid application development method	Structured analysis	Systems development life cycle method	Prototyping method	Prototyping method

59	Structured programming involves	Functional modularization	Localization of errors	Decentralized programming	Stress on analysis	Functional modularization
60	Which of the following is not a fact finding techniques?	Third party enquiry	Interview	Questionnaire	Observation	Third party enquiry
61	Which of the following questions are useful in evaluating data flow diagrams?	Are there any unnamed componenets in the data flow diagram?	Are they any rocesses that do not receive input?	Are there any data stores that are input but never referenced?	All of the above	All of the above
62	In system design and development field what does spaghetti code mean?	Programs written in nstructured languages	Well structured and well documented code	Program code that has mant GOTO statements	Both a and c above	Both a and c above
63	Which of the following statements in false with respect to a data dictionary?	It is a repository of the elements in a system	Data dictionary and data store both are same	It managers detail	It communications the common meanings for system elements and activities	Data dictionary and data store both are same

UNIT-V

SYLLABUS

System Security: Introduction - Definition and Threats to system – Security - Control measures – Recent trends in system security. Disaster/recovery planning: The plan - Ethics in system development - Ethical codes and standards of behavior.

Systems analysis and design

The **systems analysis and design (SAD)** is the process of developing information systems (IS) that effectively use hardware, software, data, processes, and people to support the company's businesses objectives. System analysis and design can be considered the meta-development activity, which serves to set the stage and bound the problem. SAD can be leveraged to set the correct balance among competing high-level requirements in the functional and non-functional analysis domains.

System analysis and design interacts strongly with distributed enterprise architecture, enterprise I.T. Architecture, and business architecture, and relies heavily on concepts such as partitioning, interfaces, personae and roles, and deployment/operational modeling to arrive at a high-level system description. This high level description is then further broken down into the components and modules which can be analyzed, designed, and constructed separately and integrated to accomplish the business goal. SDLC and SAD are cornerstones of full life cycle product and system planning.

System analysis

The goal of system analysis is to determine where the problem is in an attempt to fix the system. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

Design

In systems design, the design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems.

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts.

Design elements describe the desired system features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo-code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the system in sufficient detail, such that skilled developers and engineers may develop and deliver the system with minimal additional input design.

Security introduction

Security is the degree of resistance to, or protection from, harm. It applies to any vulnerable and valuable asset, such as a person, dwelling, community, nation, or organization.

As noted by the Institute for Security and Open Methodologies (ISECOM) in the OSSTMM 3, security provides "a form of protection where a separation is created between the assets and the threat." These separations are generically called "controls," and sometimes include changes to the asset or the threat.

Definition and threats to system-Security**Definition - What does *Threat* mean?**

A threat, in the context of computer security, refers to anything that has the potential to cause serious harm to a computer system. A threat is something that may or may not happen, but has the potential to cause serious damage. Threats can lead to attacks on computer systems, networks and more. Threats are potentials for vulnerabilities to turn into attacks on computer systems, networks, and more. They can put individuals' computer systems and business computers at risk, so vulnerabilities have to be fixed so that attackers cannot infiltrate the system and cause damage.

Threats can include everything from viruses, trojans, back doors to outright attacks from hackers. Often, the term blended threat is more accurate, as the majority of threats involve multiple exploits. For example, a hacker might use a phishing attack to gain information about a network and break into a network.

The 10 most common security threats explained

Malware: Malware is short for “malicious software.” Wikipedia describes malware as a term used to mean a “variety of forms of hostile, intrusive, or annoying software or program code.” Malware could be computer viruses, worms, Trojan horses, dishonest spyware, and malicious rootkits—all of which are defined below.

Computer virus: A computer virus is a small piece of software that can spread from one infected computer to another. The virus could corrupt, steal, or delete data on your computer—even erasing everything on your hard drive. A virus could also use other programs like your email program to spread itself to other computers.

Rogue security software: Have you ever seen a pop-up window that advertises a security update or alert? It appears legitimate and asks you to click on a link to install the “update” or “remove” unwanted malicious software that it has apparently detected. This could be rogue security software designed to lure people into clicking and downloading malicious software. Microsoft has a useful webpage that describes rogue security software and how you can protect yourself.

Trojan horse: Users can infect their computers with Trojan horse software simply by downloading an application they thought was legitimate but was in fact malicious. Once inside your computer, a Trojan horse can do anything from record your passwords by logging keystrokes (known as a keystroke logger) to hijacking your webcam to watch and record your every move.

In February 2010, a Guardian Analytics and Ponemon Institute study of 500 small businesses in the U.S. found that 55 percent of respondents experienced a fraud attack in the last 12 months. The study reports that “well-funded cyber criminals executed a full-scale assault on authentication, leveraging widespread infection of end-user computers with banking Trojans to sneak into online banking accounts completely undetected.”

Malicious spyware: Malicious spyware is used to describe the Trojan application that was created by cybercriminals to spy on their victims. An example would be keylogger software that records a victim’s every keystroke on his or her keyboard. The recorded information is periodically sent back to the originating cybercriminal over the Internet. Keylogging software is widely available and is marketed to parents or businesses that want to monitor their kids’ or employees’ Internet usage.

Computer worm: A computer worm is a software program that can copy itself from one computer to another, without human interaction. Worms can replicate in great volume and with great speed. For example, a worm can send copies of itself to every contact in your email address book and then send itself to all the contacts in your contacts’ address books.

Because of their speed of infection, worms often gain notoriety overnight infecting computers across the globe as quickly as victims around the world switch them on and open their email. This happened with the Conficker worm (also known as Downadup), which, in just four days, had more than tripled the number of computers it infected to 8.9 million.

Botnet: A botnet is a group of computers connected to the Internet that have been compromised by a hacker using a computer virus or Trojan horse. An individual computer in the group is known as a “zombie” computer.

The botnet is under the command of a “bot herder” or a “bot master,” usually to perform nefarious activities. This could include distributing spam to the email contact addresses on each zombie computer, for example. If the botnet is sufficiently big in number, it could be used to access a targeted website simultaneously in what’s known as a denial-of-service (DoS) attack. The goal of a DoS attack is to bring down a web server by overloading it with access requests. Popular websites such as Google and Twitter have been victims of DoS attacks.

Spam: Spam in the security context is primarily used to describe email spam —unwanted messages in your email inbox. Spam, or electronic junk mail, is a nuisance as it can clutter your mailbox as well as potentially take up space on your mail server. Unwanted junk mail advertising items you don’t care for is harmless, relatively speaking. However, spam messages can contain links that when clicked on could go to a website that installs malicious software onto your computer.

Phishing: Phishing scams are fraudulent attempts by cybercriminals to obtain private information. Phishing scams often appear in the guise of email messages designed to appear as though they are from legitimate sources. For example, the message would try to lure you into giving your personal information by pretending that your bank or email service provider is updating its website and that you must click on the link in the email to verify your account information and password details.

Rootkit: According to TechTarget, a rootkit is a collection of tools that are used to obtain administrator-level access to a computer or a network of computers. A rootkit could be installed on your computer by a cybercriminal exploiting a vulnerability or security hole in a legitimate application on your PC and may contain spyware that monitors and records keystrokes.

Rootkits gained notoriety when, in 2005, a security blogger discovered that a copy-protection tool inside music CDs from Sony BMG Music Entertainment was secretly installing a rootkit when users copied the CD onto their computers. At the time, security expert Bruce Schneier warned that the rootkit could allow a hacker to “gain and maintain access to your system and you wouldn’t know it.”

These are perhaps the most common security terms you'll come across to describe the different methods cybercriminals use. You can find more useful information about security terms and examples of security threats in the Cisco 3Q10 Global Threat Report.

Security Threats

Computer systems face a number of security threats. One of the basic threats is data loss, which means that parts of a database can no longer be retrieved. This could be the result of physical damage to the storage medium (like fire or water damage), human error or hardware failures.

Another security threat is unauthorized access. Many computer systems contain sensitive information, and it could be very harmful if it were to fall in the wrong hands. Imagine someone getting a hold of your social security number, date of birth, address and bank information. Getting unauthorized access to computer systems is known as hacking. Computer hackers have developed sophisticated methods to obtain data from databases, which they may use for personal gain or to harm others.

A third category of security threats consists of viruses and other harmful programs. A computer virus is a computer program that can cause damage to a computer's software, hardware or data. It is referred to as a virus because it has the capability to replicate itself and hide inside other computer files.

System Security

The objective of system security is the protection of information and property from theft, corruption and other types of damage, while allowing the information and property to remain accessible and productive. System security includes the development and implementation of security countermeasures. There are a number of different approaches to computer system security, including the use of a firewall, data encryption, passwords and biometrics.

Firewall

One widely used strategy to improve system security is to use a firewall. A **firewall** consists of software and hardware set up between an internal computer network and the Internet.

A computer network manager sets up the rules for the firewall to filter out unwanted intrusions. These rules are set up in such a way that unauthorized access is much more difficult.

A system administrator can decide, for example, that only users within the firewall can access particular files, or that those outside the firewall have limited capabilities to modify the files. You can also set up a firewall for your own computer, and on many computer systems, this is built into the operating system.

Encryption

One way to keep files and data safe is to use encryption. This is often used when data is transferred over the Internet, where it could potentially be seen by others. **Encryption** is the process of encoding messages so that it can only be viewed by authorized individuals. An encryption key is used to make the message unreadable, and a secret decryption key is used to decipher the message.

Encryption is widely used in systems like e-commerce and Internet banking, where the databases contain very sensitive information. If you have made purchases online using a credit card, it is very likely that you've used encryption to do this.

Passwords

The most widely used method to prevent unauthorized access is to use passwords. A **password** is a string of characters used to authenticate a user to access a system. The password needs to be kept secret and is only intended for the specific user. In computer systems, each password is associated with a specific username since many individuals may be accessing the same system.

Good passwords are essential to keeping computer systems secure. Unfortunately, many computer users don't use very secure passwords, such as the name of a family member or important dates - things that would be relatively easy to guess by a hacker. One of the most widely used passwords - you guessed it - 'password.' Definitely not a good password to use.

So what makes for a strong password?

- Longer is better - A long password is much harder to break. The minimum length should be 8 characters, but many security experts have started recommending 12 characters or more.
- Avoid the obvious - A string like '0123456789' is too easy for a hacker, and so is 'LaDyGaGa'. You should also avoid all words from the dictionary.
- Mix it up - Use a combination of upper and lowercase and add special characters to make a password much stronger. A password like 'hybq4' is not very strong, but 'Hy%Bq&4\$' is very strong.

Remembering strong passwords can be challenging. One tip from security experts is to come up with a sentence that is easy to remember and to turn that into a password by using abbreviations and substitutions. For example, 'My favorite hobby is to play tennis' could become something like Mf#Hi\$2Pt%.

Regular users of computer systems have numerous user accounts. Just consider how many accounts you use on a regular basis: email, social networking sites, financial institutions, online shopping sites and so on. A regular user of various computer systems and web sites will have dozens of different accounts, each with a username and password. To make things a little bit easier on computer users, a number of different approaches have been developed.

Control Measures

ADVANCED SYSTEM SECURITY MEASURES

The *Advanced System Security Measures* define the Security Measures that must be applied to *high criticality* systems. The requirements are:

I. Audit and Accountability

1. **Enable process auditing or accounting:** Enable process auditing or accounting, which generates logs information about the creation of new processes and their system activity.
2. **Audit privilege escalation or change in privilege:** Generate a log message whenever a user changes their level of privilege.
3. **Audit firewall denial:** Generate a log message when the host-based firewall denies a network connection.
4. **Audit all significant application events:** Log all significant application events.

5. **Write audit events to a separate system:** System logs must be written to a remote system in such a way that they cannot be altered by any user on the system being logged.

II. Configuration and Maintenance

1. **Follow advanced vendor security recommendations:** This document cannot be comprehensive for all systems and applications available. Conform to best practices and recommendations outlined in vendor security whitepapers and documentation.
2. **Host-based and network-based firewalls:** Systems must be protected by both a host-based and a network-based firewall that allows only those incoming connections necessary to fulfill the business need of that system.
3. **Configuration management process:** Configuration changes must be regulated by a documented configuration and change management process.
4. **Partitioning:** Systems may share hardware and resources only with other systems that have similar security requirements, regardless of their *criticality* classification. Systems which share similar security requirements have user communities of similar size and character, similar firewall profiles, and similar technical requirements. For example:
 1. Multiple systems of the same *criticality* may be aggregated together to share hardware and resources provided they have similar security requirements.
 2. *High criticality* systems may share hardware and resources with *medium* and *low criticality* systems provided that all systems meet the *advanced systems Security Measures*, and share similar security requirements.

III. Additional Requirements

1. **Physical access:** The system must reside in a secured, managed data-center.

Recent trends in system security

For a long time enterprises have been primarily concerned with securing themselves against external cyber threats such as viruses and hackers. Yet some of the biggest threats of 2017 have actually been as a result of insider mistakes or misuse. The recent Equifax and Anthem breaches are proof positive that employees and contractors can be just as big a security liability as outsiders.

At root is the fact that many businesses still have insufficient visibility into what changes their users are making within their IT environment. Our own 2017 IT Risks Survey of more than 600 IT Pros confirms that 66% of organizations perceive employees to be the biggest threat to system availability and security. And the fears appear to be justified. Research from Egress Software Technologies has revealed that around one quarter (24%) of UK employees admit to intentionally sharing confidential business information outside their organization, typically to competitors or new and previous employers.

Verizon's 2017 Data Breach Investigation Report goes further, saying that employee data theft can take months or years to discover. From May 2018, GDPR will usher in a new era when data breaches will need to be reported within just 72 hours. It means organizations will be highly incentivized to adopt new strategies that make speedy discovery of cyber security breaches a top priority.

Looking ahead, a number of emerging IT security advances will arm organizations with the right information at the right time to help spot and mitigate potential breaches before they can occur. Here, in no particular order, are five security trends that are set to make a big impression on enterprise in 2018.

1. Security compliance will get serious

Regulators on both sides of the Atlantic are clamping down on security practices that put customer data at risk. In the U.S. NIST Special Publication 800-171, which comes into force December 31, 2017, will regulate the protection of controlled unclassified information (CUI) in non-federal information systems and organizations. Over in Europe, the much anticipated General Data Protection Regulation (GDPR) will ensure organizations worldwide that handle information relating to European citizens fully understand what data they have, where it is stored and who is responsible for it. These, along with stricter penalties for non-compliance, will require businesses to upgrade their data privacy controls.

2. Advanced analytics will improve data security

Organizations currently use a combination of security products from antivirus software and data loss prevention (DLP) tools to full-blown security information and event management (SIEM) software in an attempt to reduce data breach risk. SIEM in particular generates large volumes of data making it hard to spot information requiring immediate attention. Advanced data analytics tools will help organizations see the wood from the trees much more clearly. The growing adoption of technologies like user and entity behaviour analytics (UEBA) will enable organizations to establish stricter control over their IT infrastructures and better understand their weak points, so they can fix security holes before a data breach occurs.

3. Tailor-made security

The global cyber security market is evolving. Security vendors are rapidly expanding their range of solutions to allow them to solve similar pain points differently according to the customer's infrastructure. With strong data protection practices in high demand, security vendors will start to offer a more personalized approach, taking into account factors like IT infrastructure size and complexity, industry and budget. A more customized approach to IT security will provide organizations with solutions that are uniquely tailored to their requirements. Smaller, more specialist software providers will win business against larger, less flexible vendors by providing offerings that are ideally suited to meet specific business needs.

4. Gartner's CARTA approach will improve decision-making

In 2017, Gartner proposed a new approach to security based on a continuous process of regular review, re-assessment and adjustment. Known as CARTA (Continuous Risk and Trust Assessment), the new approach is intended to replace the old fit-it-and-forget-it mantra. We can expect this approach to become more central in 2018 as organizations take a fresh look at how the mitigate cyber risks. Real-time assessment of risk and trust in the IT environment enables companies to make better decisions regarding their security posture. A good example is to grant extended access rights to users only once previous patterns of behaviour on the network have been carefully studied to show they present minimal risk of privilege abuse.

5. Block chain principles to be applied to data security

An emerging approach to mitigate the increasing number and sophistication of cyber threats is to harness block chain principles to strengthen security. With block chain technology data is stored in a decentralized and distributed manner. Instead of residing in a single location, data is stored in an open source ledger. It renders mass data hacking or data tampering much more difficult because all participants in the block chain network would immediately see that the ledger had altered in some way. Block chain has the potential to be a major leap forward for securing sensitive information, especially in highly regulated industries like finance, government, health and law.

In summary, insider mistakes and privilege misuse have repeatedly been the source of security breaches and are as much a vulnerability to organizations as outsider threats. In response 2018 will see the introduction of a number of regulatory initiatives aimed at clamping down on inadequate security practices.

Even though every organization has its own individual security risks requiring different defense methods for mitigating insider and outsider attacks, some common technology trends are emerging. Businesses will need to adopt more continuous protection strategies, while vendors will take advantage of the latest technology advances to create more customized and better targeted solutions.

As a consequence we should see organizations becoming more proactive about securing confidential information, especially where consumer data is concerned. Malicious insiders and hackers alike will find their work more difficult. Stealing corporate data will take much more time and effort than it did in the past while the overall chances of being caught will also be higher.



Disaster recovery and business continuity auditing:

Disaster recovery (DR) and business continuity refers to an organization's ability to recover from a disaster and/or unexpected event and resume operations. Organizations often have a plan in place (usually referred to as a "Disaster Recovery", or "Business Continuity Plan") that outlines how a recovery will be accomplished. The key to successful disaster recovery is to have a plan (emergency plan, disaster recovery plan, and continuity plan) well before disaster ever strikes.

Given ever-changing business objectives, one common need in disaster recovery is to perform an audit of the disaster recovery capacity of an organization. The purpose of such audit is to discover how closely an organization's disaster recovery readiness aligns to actual organizational objectives. When conducting an audit of a disaster recovery plan, factors such as alternate site designation, training of personnel, and insurance issues are considered. In conducting a disaster recovery audit, the individual or team performing the audit uses a number of procedures and processes to achieve the objectives of the audit. Successful disaster recovery audits clear state their objectives in an audit plan.

Documentation:

To maximize their effectiveness, disaster recovery plans are documented in written form and in a manner that is easily understood by those who will need to use it. In addition, the plan must also be readily available as well, since digging for a hard-to-find or misplaced disaster recovery plan at a time of a disaster can complicate the effect of the disaster. Furthermore, because of the constant changes that occur in the modern business environment, disaster plans are most effective when updated frequently. This way, the plans will also cover new and existing threats as such threats develop. Adequate records need to be retained by the organization. The auditor examines records, billings, and contracts to verify that records are being kept. One such record is a current list of the organization's hardware and software vendors.

Such list is made and periodically updated to reflect changing business practice. Copies of it are stored on and off site and are made available or accessible to those who require them. An auditor tests the procedures used to meet this objective and determine their effectiveness.

Strategies

Site designation:

A hot/cold site is a location that an organization can move to after a disaster if the current facility is unusable. The difference between the two is that a hot site is fully equipped to resume operations while a cold site does not have that capability. There is also what is referred to as a warm site which has the capability to resume some, but not all operations. The decision a company makes when determining what type of site to establish often hinges on the results of a cost benefit analysis as well as the needs of the organization.

A disaster recovery plan spells out how relocation to a new facility is to be conducted. Companies perform occasional tests and conduct trials to verify the viability and effectiveness of the plan and to determine if any deficiencies exist and how they can be dealt with. An audit of a company's Disaster Recovery Plan primarily looks into the probability that operations of the organization can be sustained at the level that is assumed in the plan, as well as the ability of the entity to actually establish operations at the site. A review of the disaster recovery plan generally involves examining and testing the procedures included, conducting outside research relating to Disaster recovery, determining reasonable standards relating to implementation, and touring, examining, and researching the outside facility.

The auditor can verify this through paper and paperless documentation and actual physical observation. Testing of the backups and procedures is also performed to confirm data integrity and effective processes. The security of the storage site is also confirmed.

Data backup:

Data backups are central to any disaster recovery plan. An audit of backup processes determines if (a) they are effective, and (b) if they are actually being implemented by the involved personnel. Some techniques that are used to accomplish this include direct observation of the processes in question, analyzing and researching the backup equipment used, conducting computer-assisted audit techniques and tests, examining of paper and paperless records.

The continual backing up of data and systems can help minimize the impact of threats. Even so, the disaster recovery plan also includes information on how best to recover any data that has not been copied. Controls and protections are put in place to ensure that data is not damaged, altered, or destroyed during this process. Information technology experts and procedures need to be identified that can accomplish this endeavor. Vendor manuals can also assist in determining how best to proceed.

Drills:

Practice drills conducted periodically to determine how effective the plan is and to determine what changes may be necessary. The auditor's primary concern here is verifying that these drills are being conducted properly and that problems uncovered during these drills are addressed and procedures designed to deal with these potential deficiencies are implemented and tested to determine their effectiveness.

Backup of key personnel:

A disaster recovery plan includes clearly written policies and specific communication with employees to ensure that both regular and replacement personnel is selected, documented, and informed should a disaster occur. There must also be confirmation that the replacement personnel can actually do the duties assigned to them in an event of an emergency. Periodic training and cross training is often used to accomplish this. This training includes updates to existing job positions and testing to confirm proficiency. Some of the issues related to this activity verify that (1) policies are being enforced, (2) testing is effective, and (3) training is adequate.

Other considerations

Insurance issues:

The auditor determines the adequacy of the company's insurance coverage (particularly property and casualty insurance) through a review of the company's insurance policies and other research.

Among the items that the auditor needs to verify are: the scope of the policy (including any stated exclusions), that the amount of coverage is sufficient to cover the organization's needs, and that the policy is current and in force. The auditor also ascertains, through a review of the ratings assigned by independent rating agencies, that the insurance company or companies providing the coverage have the financial viability to cover the losses in the event of a disaster.

Effective DR Plans take into account the extent of a company's responsibilities to other entities and its ability to fulfill those commitments despite a major disaster. A good DR audit will include a review of existing MOA and contracts to ensure that the organization's legal liability for lack of performance in the event of disaster or any other unusual circumstance is minimized. Agreements pertaining to establishing support and assisting with recovery for the entity are also being outlined. Techniques used for evaluating this area include an examination of the reasonableness of the plan, a determination of whether or not the plan takes all factors into account and a verification of the contracts and agreements reasonableness through documentation and outside research.

Communication issues:

Good disaster recovery planning ensures that both management and the recovery team have disaster recovery procedures which allow for effective communication. This can be accomplished by ensuring contact information is easily accessible and that drills conducted test for communication abilities. A good disaster recovery plan includes not only internal communication considerations but external issues as well. Such external communications considers issues related to communication between the organization and outside individuals and organizations, such as business partners.

Procedures to test this communication capability generally mirror those of the organization itself. The disaster recovery evaluates these procedures and assumptions to determine if they are reasonable and likely to be effective. Some techniques used by a DR auditor in evaluating readiness include testing of procedures, interviewing employees, making comparison against the DR plans of other company and against industry standards, and examining company manuals and other written procedures. The auditor can verify through direct observation that emergency telephone numbers are listed and easily accessible in the event of a disaster.

Emergency procedures:

Procedures to sustain staff during a round-the clock disaster recovery effort are included in any good disaster recovery plan. Procedures for the stocking of food and water, capabilities of administering CPR/first aid, and dealing with family emergencies are clearly written and tested. This can generally be accomplished by the company through good training programs and a clear definition of job responsibilities. A review of the readiness capacity of a plan often includes tasks such as inquires of personnel, direct physical observation, and examination of training records and any certifications.

Environmental issues:

Disaster recovery plans may also involve procedures that take into account the possibility of power failures or other situations that are of a non-IT nature. Such plan indicates what procedures to be used in this situation and also includes information on storage of flashlights and candles, as well as additional safety procedures in case of gas leaks, fires or other such phenomena. Trial runs are conducted to test the procedures' effectiveness and viability. The readiness of an organization in this regard can be assessed by examining and testing procedures for reasonableness, making inquiries on personnel, and conducting outside research.

Ethical codes are adopted by organizations to assist members in understanding the difference between 'right' and 'wrong' and in applying that understanding to their decisions. An ethical code generally implies documents at three levels: codes of business ethics, codes of *conduct* for employees, and codes of professional practice.

Code of ethics or a code of conduct:

Many companies use the phrases 'ethical code' and 'code of conduct' interchangeably but it may be useful to make a distinction. A code of ethics will start by setting out the values that underpin the code and will describe a company's obligation to its stakeholders. The code is publicly available and addressed to anyone with an interest in the company's activities and the way it does business.

It will include details of how the company plans to implement its values and vision, as well as guidance to staff on ethical standards and how to achieve them. However, a code of conduct is generally addressed to and intended for employees alone. It usually sets out restrictions on behavior, and will be far more compliance or rules focused than value or principle focused.

Code of practice (professional ethics):

A code of practice is adopted by a profession or by a governmental or non-governmental organization to regulate that profession. A code of practice may be styled as a code of professional responsibility, which will discuss difficult issues, difficult decisions that will often need to be made, and provide a clear account of what behavior is considered "ethical" or "correct" or "right" in the circumstances. In a membership context, failure to comply with a code of practice can result in expulsion from the professional organization. In its 2007 International Good Practice Guidance, *Defining and Developing an Effective Code of Conduct for Organizations*, the International Federation of Accountants provided the following working definition: "Principles, values, standards, or rules of behavior that guide the decisions, procedures and systems of an organization in a way that (a) contributes to the welfare of its key stakeholders, and (b) respects the rights of all constituents affected by its operations."

General notes:

Ethical codes are often adopted by management, not to promote a particular moral theory, but rather because they are seen as pragmatic necessities for running an organization in a complex society in which moral concepts play an important part.

Systems development:

The **systems development life cycle (SDLC)**, also referred to as the **application development life-cycle**, is a term used in systems engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system.

The systems development life-cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both.

Systems development life cycle:

A systems development life cycle is composed of a number of clearly defined and distinct work phases which are used by systems engineers and systems developers to plan for, design, build, test, and deliver information systems. Like anything that is manufactured on an assembly line, an SDLC aims to produce high quality systems that meet or exceed customer expectations, based on customer requirements, by delivering systems which move through each clearly defined phase, within scheduled time-frames and cost estimates.

Computer systems are complex and often (especially with the recent rise of service oriented architecture) link multiple traditional systems potentially supplied by different software vendors. To manage this level of complexity, a number of SDLC models or methodologies have been created, such as "waterfall"; "spiral"; "Agile software"; "rapid prototyping"; "incremental"; and "synchronize and stabilize".

SDLC can be described along a spectrum of agile to iterative to sequential. Agile methodologies, such as XP and Scrum, focus on lightweight processes which allow for rapid changes (without necessarily following the pattern of SDLC approach) along the development cycle. Iterative methodologies, such as Rational Unified Process and dynamic systems development method, focus on limited project scope and expanding or improving products by multiple iterations.

Sequential or big-design-up-front (BDUF) models, such as waterfall, focus on complete and correct planning to guide large projects and risks to successful and predictable results. Other models, such as anamorphic development, tend to focus on a form of development that is guided by project scope and adaptive iterations of feature development.

In project management a project can be defined both with a project lifecycle (PLC) and an SDLC, during which slightly different activities occur. According to Taylor (2004) "the project life cycle encompasses all the activities of the project, while the systems development life cycle focuses on realizing the product requirements".

SDLC is used during the development of an IT project, it describes the different stages involved in the project from the drawing board, through the completion of the project. The product lifecycle describes the process for building information systems in a very deliberate, structured and methodical way, reiterating each stage of the product's life.

The systems development life cycle, according to Elliott & Strachan & Radford (2004), "originated in the 1960s, to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and number crunching routines". Several systems development frameworks have been partly based on SDLC, such as the structured systems analysis and design method (SSADM) produced for the UK government Office of Government Commerce in the 1980s. Ever since, according to Elliott (2004), "the traditional life cycle approaches to systems development have been increasingly replaced with alternative approaches and frameworks, which attempted to overcome some of the inherent deficiencies of the traditional SDLC".

The system development life cycle framework provides a sequence of activities for system designers and developers to follow. It consists of a set of steps or phases in which each phase of the SDLC uses the results of the previous one.

The SDLC adheres to important phases that are essential for developers, such as planning, analysis, design, and implementation, and are explained in the section below. It includes evaluation of present system, information gathering, and feasibility study and request approval. A number of SDLC models have been created: waterfall, fountain, and spiral build and fix, rapid prototyping, incremental, and synchronize and stabilize. The oldest of these, and the best known, is the waterfall model: a sequence of stages in which the output of each stage becomes the input for the next. These stages can be characterized and divided up in different ways, including the following:

Preliminary analysis: The objective of phase 1 is to conduct a preliminary analysis, propose alternative solutions, describe costs and benefits and submit a preliminary plan with recommendations.

Conduct the preliminary analysis: in this step, you need to find out the organization's objectives and the nature and scope of the problem under study. Even if a problem refers only to a small segment of the organization itself then you need to find out what the objectives of the organization itself are. Then you need to see how the problem being studied fits in with them.

Propose alternative solutions: In digging into the organization's objectives and specific problems, you may have already covered some solutions. Alternate proposals may come from interviewing employees, clients, suppliers, and/or consultants. You can also study what competitors are doing. With this data, you will have three choices: leave the system as is, improve it, or develop a new system.

Describe the costs and benefits.

Systems analysis, requirements definition: Defines project goals into defined functions and operation of the intended application. Analyzes end-user information needs.

Systems design: Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.

Development: The real code is written here.

Integration and testing: Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.

Acceptance, installation, deployment: The final stage of initial development, where the software is put into production and runs actual business.

Maintenance: During the maintenance stage of the SDLC, the system is assessed to ensure it does not become obsolete. This is also where changes are made to initial software. It involves continuous evaluation of the system in terms of its performance.

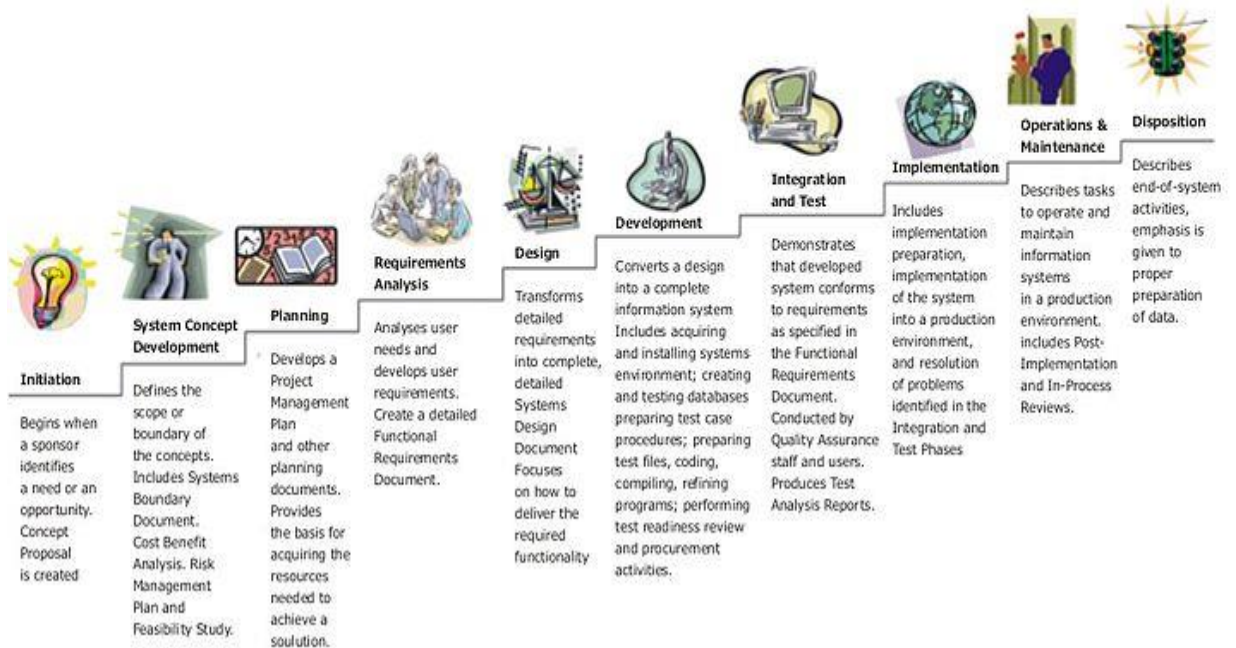
Evaluation: Some companies do not view this as an official stage of the SDLC, but is it an important part of the life cycle. Evaluation step is an extension of the Maintenance stage, and may be referred to in some circles as Post-implementation Review. This is where the system that was developed, as well as the entire process, is evaluated. Some of the questions that need to be answered include: does the newly implemented system meet the initial business requirements and objectives? Is the system reliable and fault-tolerant? Does the system function according to the approved functional requirements.

In addition to evaluating the software that was released, it is important to assess the effectiveness of the development process. If there are any aspects of the entire process, or certain stages, that management is not satisfied with, this is the time to improve. Evaluation and assessment is a difficult issue. However, the company must reflect on the process and address weaknesses.

Disposal: In this phase, plans are developed for discarding system information, hardware and software in making the transition to a new system. The purpose here is to properly move, archive, discard or destroy information, hardware and software that is being replaced, in a manner that prevents any possibility of unauthorized disclosure of sensitive data. The disposal activities ensure proper migration to a new system. Particular emphasis is given to proper preservation and archival of data processed by the previous system. All of this should be done in accordance with the organization's security requirements.

In the following example (see picture) these stages of the systems development life cycle are divided in ten steps from definition to creation and modification of IT work products: The tenth phase occurs when the system is disposed of and the task performed is either eliminated or transferred to other systems. The tasks and work products for each phase are described in subsequent chapters. Not every project will require that the phases be sequentially executed. However, the phases are interdependent. Depending upon the size and complexity of the project, phases may be combined or may overlap.

Systems Development Life Cycle (SDLC) Life-Cycle Phases



System investigation

The system investigation stage addresses the needs or opportunities that can be achieved by a sponsor or IT proposal. During this step, we must consider all current priorities that would be affected and how they should be handled. Before any system planning is done, a feasibility study should be conducted to determine if creating a new or improved system is a viable solution. This will help to determine the costs, benefits, resource requirements, and specific user needs required for completion. The development process can only continue once management approves of the recommendations from the feasibility study. Following are different components of the feasibility study:

- Operational feasibility

- Economic feasibility
- Technical feasibility
- Human factors feasibility
- Legal/Political feasibility

System analysis

The goal of system analysis is to determine where the problem is in an attempt to fix the system. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

Design

In systems design, the design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems.

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts.

Design elements describe the desired system features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo-code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the system in sufficient detail, such that skilled developers and engineers may develop and deliver the system with minimal additional input design.

Environments

Environments are controlled areas where systems developers can build, distribute, install, configure, test, and execute systems that move through the SDLC. Each environment is aligned with different areas of the SDLC and is intended to have specific purposes. Examples of such environments include the:

- *Development environment*, where developers can work independently of each other before trying to merge their work with the work of others,
- *Common build environment*, where merged work can be built, together, as a combined system,
- *Systems integration testing environment*, where basic testing of a system's integration points to other upstream or downstream systems can be tested,
- *User acceptance testing environment*, where business stakeholders can test against their original business requirements,
- *Production environment*, where systems finally get deployed to, for final use by their intended end users.

The planning for, provisioning, and operating of such environments is known as practice of IT environment management.

Testing

It's tested at various levels in software testing. Unit, system and user acceptance testing are often performed. This is a grey area as many different opinions exist as to what the stages of testing are and how much, if any iteration occurs. Iteration is not generally part of the waterfall model, but usually some occur at this stage. In the testing the whole system is tested one by one

Following are the types of testing:

- *Defect testing* the failed scenarios, including defect tracking
- Path testing
- Data set testing
- Unit testing

- System testing
- Integration testing
- Black-box testing
- White-box testing
- Regression testing
- Automation testing
- User acceptance testing
- Software performance testing

Training and transition

Once a system has been stabilized through adequate testing, the SDLC ensures that proper training on the system is performed or documented before transitioning the system to its support staff and end users.

Training usually covers operational training for those people who will be responsible for supporting the system as well as training for those end users who will be using the system after its delivery to a production operating environment.

After training has been successfully completed, systems engineers and developers transition the system to its final production environment, where it is intended to be used by its end users and supported by its support and operations staff.

Operations and maintenance

The deployment of the system includes changes and enhancements before the decommissioning or sunset of the system. Maintaining the system is an important aspect of SDLC. As key personnel change positions in the organization, new changes will be implemented. There are two approaches to system development; there is the traditional approach (structured) and object oriented.

Information Engineering includes the traditional system approach, which is also called the structured analysis and design technique. The object oriented approach views the information system as a collection of objects that are integrated with each other to make a full and complete information system.

Evaluation

The final phase of the SDLC is to measure the effectiveness of the application and evaluate potential enhancements.

Systems analysis and design

The **systems analysis and design (SAD)** is the process of developing information systems (IS) that effectively use hardware, software, data, processes, and people to support the company's businesses objectives. System analysis and design can be considered the meta-development activity, which serves to set the stage and bound the problem. SAD can be leveraged to set the correct balance among competing high-level requirements in the functional and non-functional analysis domains.

System analysis and design interacts strongly with distributed enterprise architecture, enterprise I.T. Architecture, and business architecture, and relies heavily on concepts such as partitioning, interfaces, personae and roles, and deployment/operational modeling to arrive at a high-level system description. This high level description is then further broken down into the components and modules which can be analyzed, designed, and constructed separately and integrated to accomplish the business goal. SDLC and SAD are cornerstones of full life cycle product and system planning.

Object-oriented analysis

Object-oriented analysis (OOA) is the process of analyzing a task (also known as a problem domain), to develop a conceptual model that can then be used to complete the task. A typical OOA model would describe computer software that could be used to satisfy a set of customer-defined requirements. During the analysis phase of problem-solving, a programmer might consider a written requirements statement, a formal vision document, or interviews with stakeholders or other interested parties.

The task to be addressed might be divided into several subtasks (or domains), each representing a different business, technological, or other areas of interest. Each subtask would be analyzed separately. Implementation constraints, (e.g., concurrency, distribution, persistence, or how the system is to be built) are not considered during the analysis phase; rather, they are addressed during object-oriented design (OOD).

The conceptual model that results from OOA will typically consist of a set of use cases, one or more UML class diagrams, and a number of interaction diagrams. It may also include some kind of user interface mock-up.

The input for object-oriented design is provided by the output of object-oriented analysis. Realize that an output artifact does not need to be completely developed to serve as input of object-oriented design; analysis and design may occur in parallel, and in practice the results of one activity can feed the other in a short feedback cycle through an iterative process. Both analysis and design can be performed incrementally, and the artifacts can be continuously grown instead of completely developed in one shot.

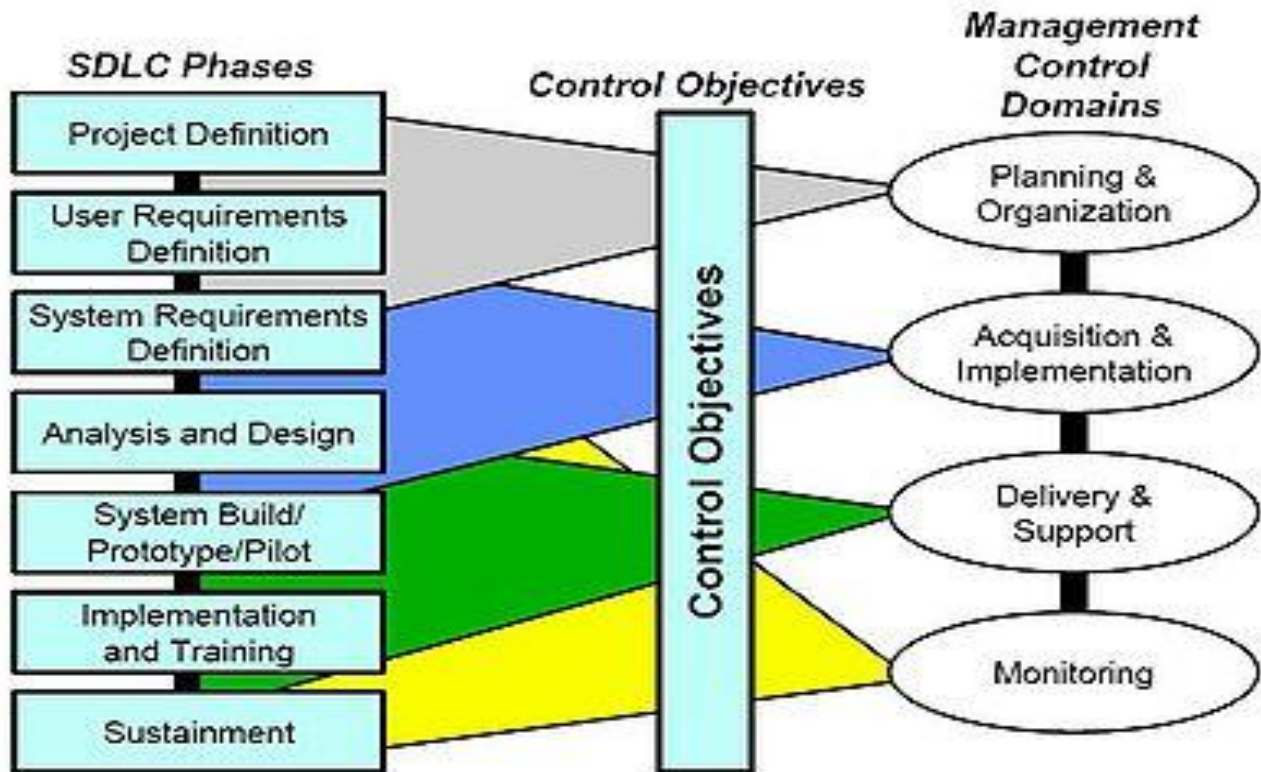
Some typical input artifacts for object-oriented:

- Conceptual model: Conceptual model is the result of object-oriented analysis; it captures concepts in the problem domain. The conceptual model is explicitly chosen to be independent of implementation details, such as concurrency or data storage.

- Use case: Use case is a description of sequences of events that, taken together, lead to a system doing something useful. Each use case provides one or more scenarios that convey how the system should interact with the users called actors to achieve a specific business goal or function. Use case actors may be end users or other systems. In many circumstances use cases are further elaborated into use case diagrams. Use case diagrams are used to identify the actor (users or other systems) and the processes they perform.
- System Sequence Diagram: System Sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events.
- User interface documentations (if applicable): Document that shows and describes the look and feel of the end product's user interface. It is not mandatory to have this, but it helps to visualize the end-product and therefore helps the designer.
- Relational data model (if applicable): A data model is an abstract model that describes how data is represented and used. If an object database is not used, the relational data model should usually be created before the design, since the strategy chosen for object-relational mapping is an output of the OOA design process. However, it is possible to develop the relational data model and the object-oriented design artifacts in parallel and the growth of an artifact can stimulate the refinement of other artifacts.

Life cycle

Management and control



SPIU phases related to management controls

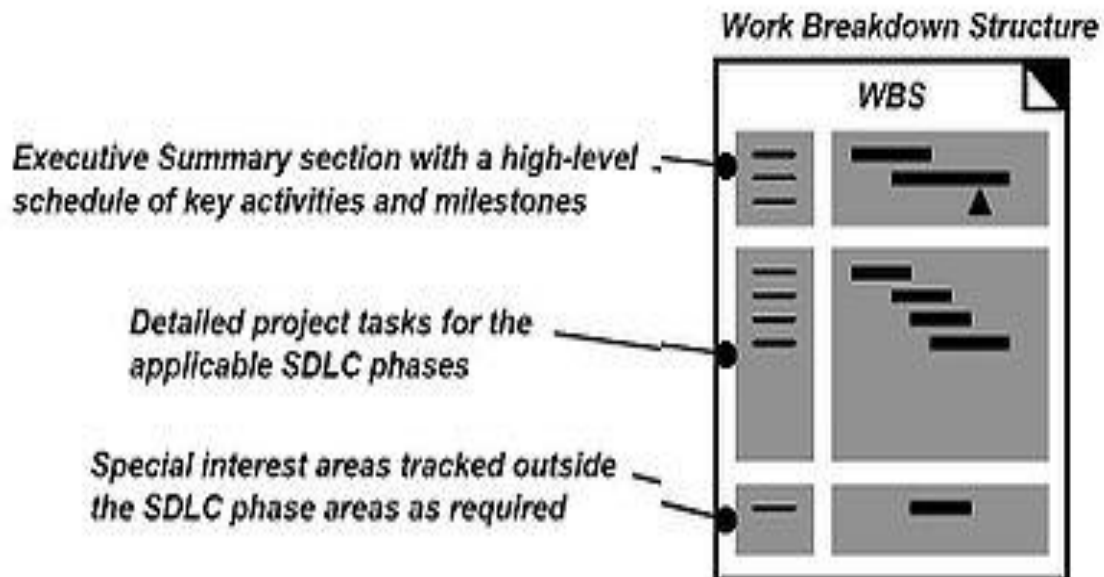
The SDLC phases serve as a programmatic guide to project activity and provide a flexible but consistent way to conduct projects to a depth matching the scope of the project. Each of the SDLC phase objectives are described in this section with key deliverables, a description of recommended tasks, and a summary of related control objectives for effective management. It is critical for the project manager to establish and monitor control objectives during each SDLC phase while executing projects. Control objectives help to provide a clear statement of the desired result or purpose and should be used throughout the entire SDLC process. Control objectives can be grouped into major categories (domains), and relate to the SDLC phases as shown in the figure.

To manage and control any SDLC initiative, each project will be required to establish some degree of a work breakdown structure (WBS) to capture and schedule the work necessary to complete the project.

The WBS and all programmatic material should be kept in the "project description" section of the project notebook. The WBS format is mostly left to the project manager to establish in a way that best describes the project work.

There are some key areas that must be defined in the WBS as part of the SDLC policy. The following diagram describes three key areas that will be addressed in the WBS in a manner established by the project manager.

Work breakdown structured organization



Work breakdown structure

The upper section of the work breakdown structure (WBS) should identify the major phases and milestones of the project in a summary fashion. In addition, the upper section should provide an overview of the full scope and timeline of the project and will be part of the initial project description effort leading to project approval. The middle section of the WBS is based on the seven systems development life cycle phases as a guide for WBS task development.

The WBS elements should consist of milestones and "tasks" as opposed to "activities" and have a definitive period (usually two weeks or more). Each task must have a measurable output (e.x. document, decision, or analysis).

A WBS task may rely on one or more activities (e.g. software engineering, systems engineering) and may require close coordination with other tasks, either internal or external to the project. Any part of the project needing support from contractors should have a statement of work (SOW) written to include the appropriate tasks from the SDLC phases. The development of a SOW does not occur during a specific phase of SDLC but is developed to include the work from the SDLC process that may be conducted by external resources such as contractors.

Baselines

Baselines are an important part of the systems development life cycle. These baselines are established after four of the five phases of the SDLC and are critical to the iterative nature of the model. Each baseline is considered as a milestone in the SDLC.

- functional baseline: established after the conceptual design phase.
- allocated baseline: established after the preliminary design phase.
- product baseline: established after the detail design and development phase.
- updated product baseline: established after the production construction phase.

Code of Ethics

Public responsibility

- a. rights of others
- b. long-range consequences of present actions
- c. interrelatedness of decisions
- d. *Provide adequate, timely, clear & accurate information*

Potential Conflict of Interests

The county council is considering an ordinance that would drastically increase the water and sewage fees for rental units. The county's housing planner has analyzed the proposal and feels that the proposed fees are excessive because the amount of water consumed by apartment units is far less than that of single-family houses. The planner also feels the rate hikes will exacerbate the county's existing rental housing shortage by encouraging the conversion of rental units to condominiums.

Here you are, ready to live by your very own ethical principles and lead a satisfying and meaningful life. One can draw ideas and applications from one's religion, spiritual beliefs, higher order teachings, from a mentor or simply use your own gut instincts to implement these changes. The only problem is, you're not sure what those principles are. Ethics is about relationships. It is about working to develop a well informed conscience and it is about being true to the idea of who we are and what we stand for. It is about having the courage to explore difficult questions and it is about being accountable. You will need values, morals, concepts, to understand right from wrong, having knowledge, need wisdom, having intelligence. Here are a few ideas for getting started on coming up with your own code of ethics.

ETHICS

Ethics, sometimes known as **philosophical ethics**, **ethical theory**, **moral theory**, and **moral philosophy**, is a branch of philosophy that involves systematizing, defending and recommending concepts of right and wrong conduct, often addressing disputes of moral diversity. The term comes from the Greek word *ethikos* from *ethos*, which means "custom, habit". The super field within philosophy known as axiology includes both ethics and aesthetics and is unified by each sub-branch's concern with value. Philosophical ethics investigates what is the best way for humans to live, and what kinds of actions are right or wrong in particular circumstances.

Ethics may be divided into three major areas of study:

Meta-ethics, about the theoretical meaning and reference of moral propositions and how their truth values (if any) may be determined

- Normative ethics, about the practical means of determining a moral course of action
- Applied ethics draws upon ethical theory in order to ask what a person is obligated to do in some very specific situation, or within some particular domain of action (such as business)

Related fields are moral psychology, descriptive ethics, and value theory. Ethics seeks to resolve questions dealing with human morality—concepts such as good and evil, right and wrong, virtue and vice, justice and crime.

Part A (ONE Mark)

Multiple Choice Questions

Online Examination

Part B (2 Marks)

1. How is stress testing different from volume testing?
2. What types of test data are used in system testing?
3. List and briefly describe the factors that affect the quality of a system.
4. What is implementation? How does it differ from conversion? Elaborate.
5. What is involved in converting files? Explain briefly.
6. What is the role of audit control trial in conversion? Who performs it? Explain.
7. What levels of quality assurance must a system meet? Explain.

Part C (5 Marks)

1. Define the term Quality Assurance. Discuss its importance in system design.
2. Discuss the role of the Data Processing Auditor in system testing.
3. Explain the various activities in conversion. Which activity is the most important? Why?
4. Distinguish between System maintenance and enhancement.
5. How System modification is different from software system audit.
6. Review the primary activities of a maintenance procedure.
7. Discuss the various training aids used for training users on a new system.
8. Briefly explain the procedure and makeup of the post-implementation review. Can one perform maintenance on a system without a post implementation review? Why?

	KARPAGAM ACADEMY OF HIGHER EDUCATION					
	Department of Management					
	Unit 5- System Analysis and Design -Multiple Choice Questions- Each Question Carry ONE Mark					
S. No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	Which of the following relationship are used in a use case diagram?	a) Communication	b) Extends	c) Uses	d) All of the above	d) All of the above
2	Some object oriented systems permit a class to inherit its state and behaviours from more than one super class. This is called	a) Multiple inheritance	b) Inheritance	c) Hybrid inheritance	d) Specialization	a) Multiple inheritance
3	Which of the following are UML interaction diagrams?	a) Activity diagram	b) Sequence diagram	c) Collaboration diagram	d) Both b and c above	d) Both b and c above
4	The vertical dimensions of a sequences diagram represents	a) Time	b) Objects	c) Lines	d) Messages	a) Time
5	Which of the following component is used to clarity which actor erforms which activity in an activity diagram?	a) Forks	b) Joins	c) Swimlanes	d) State	c) Swimlanes
6	Which of the folowing statements is not true regarding activity diagram?	a) The solid filled circle represents the final state	b) The final state is shown using so called bull's eye symbol	c) Transitions can branch and merge alternative computation threads	d) Activity diagram without concurrent rocesses resembles a conventional flowchart	a) The solid filled circle represents the final state
7	_____ can be defined as data that has been processes into a aform that is meaningful to the recipient and is of real or perceive value in current or prospective decisions	a) System	b) Information	c) Technology	d) Service	b) Information
8	Which of the following diagrams model the physical components of the system?	a) Use case diagram	b) Collaboration diagram	c) Component diagram	d) Deloyment diagram	d) Deloyment diagram

9	Which of the following the built in extensibility mechanism of the UML?	a) Associations	b) Relationships	c) Stereotypes	d) Comments	c) Stereotypes
10	Noun phrase approach and CRC approach are used to identify	a) Classes	b) Use cases	c) Objects	d) Collaborators	a) Classes
11	The largest percentage of total life cycle cost of software is	Conceptual	b) Design costs	c) Maintenance costs	d) Coding costs	c) Maintenance costs
12	Which of the following technique detects transposition errors?	a) Check digit	b) Automatic correction	c) Existence test	d) Duplicate processing	a) Check digit
13	The structure chart derived by studying the flow through the system supports the activity of	a) File design	b) Programs design	c) Database design	d) Internal control design	d) Internal control design
14	A _____ system is no more than an idea	Conceptual	Logical	Physical	None of the above	Conceptual
15	Class is analogous to	a) Object	b) Blueprint	c) Instance	d) Record	b) Blueprint
16	Which of the following represents the condition of an object at a specific moment in time?	a) Behaviour	b) Properties	c) Instance	d) State	d) State
17	Identify the following who presented the object modelling technique (OMT)	a) Booch	b) James Rumbaugh ET AL	c) Both a and b above	d) None of the above	b) James Rumbaugh ET AL
18	What category of information system determines the sale of an item and a withdrawal from an ATM?	a) Management information systems	b) Executive information system	c) Communication support system	d) Transaction processing system	d) Transaction processing system
19	Which of the following is not true regarding the waterfall method?	a) Fairly rigid approach	b) Can easily go back to previous phases	c) Good for traditional type of projects	d) Not as good for many of the new types of interactive and highly complex applications	b) Can easily go back to previous phases

20	Which feasibility determines the availability of team and support staff?	a) Economic feasibility	b) Cultural feasibility	c) Schedule feasibility	d) Resource feasibility	c) Schedule feasibility
21	Which among the following is an intangible benefits	a) Maintaining constant staff	b) Decreasing operating expenses	c) Survival	d) Reducing error rates	c) Survival
22	Which chart is represented by vertical bars?	a) PERT	b) ROI	c) Gantt	d) NPV	c) Gantt
23	Which of the model is used for system components?	a) PERT chart	b) Gantt chart	c) CPM	d) DFD	d) DFD
24	Which is not used in context level diagram?	a) Source	b) Destination	c) Data flow	d) Data store	d) Data store
25	Which of the following is a not resource for setting JAD sessions?	a) Overhead projector	b) Black or white board	c) Flip chart	d) All of the above	d) All of the above
26	What will help the system analyst to work with users to determine system usage?	a) Use case	b) Actor	c) Class	d) Component	a) Use case
27	A data store is represented in data flow diagram is	a) Rectangle	b) Square	c) Open rectangle	d) Open square	c) Open rectangle
28	The symbol represents	a) Aggregation	b) Generalization	c) Dependency	d) Composition	b) Generalization
29	Which cohesion operates on the same input or output data?	a) Communicational	b) Temporal	c) Functional	d) Procedural	a) Communicational
30	_____ is an important factor of management information system	a) System	b) Data	c) Process	d) All of the above	a) System
31	Which are the following is/ are the level of documentation?	a) Documentation for management	b) Documentation for user	c) Documentation for data processing department	d) All of the above	d) All of the above

32	_____ level supply information to supply information to strategic tier for the use of top management	a) Operational	b) Environmental	c) Competitive	d) Tactical	d) Tactical
33	In a DFD external entities are represented by a	a) Rectangle	b) Ellipse	c) Diamond shaped box	d) Circle	a) Rectangle
34	Use the new system as the same time as old system to compare the results this is known as _____	a) Procedure writing	b) Simultaneous processing	c) Parallel operation	d) File conversion	c) Parallel operation
35	Decision making model was proposed by _____	a) Harry goode	b) Herbert A simon	c) Recon michal	d) None of the above	b) Herbert A simon
36	A data flow can	a) Only emanate from an external entity	b) Only terminate in an external entity	c) May emanate and terminate in an external entity	d) May either emanate or terminate in an external entity but not both	c) May emanate and terminate in an external entity
37	_____ can be defined as most recent and perhaps the most comprehensive technique for solving computer problems	a) System analysis	b) System data	c) System procedure	d) System record	a) System analysis
38	SDLC stands for	a) System development life cycle	b) Structure design life cycle	c) System design life cycle	d) Structure development life cycle	a) System development life cycle
39	The data dictionary in SDLC contains descriptions of	a) DFD elements	b) E-R diagram	c) Use case	d) Class diagram	a) DFD elements
40	A _____ is a tabular form of presentation that specifies a set of conditions and their corresponding actions	a) Decision table	b) Decision	c) Structured english	d) Data flow diagram	d) Data flow diagram
41	The requirement definition document is intended for	a) System end users	b) Client engineers	c) Software developers	d) System architecture	d) System architecture
42	_____ is a sort of blueprint of the system development effort	MDP	DMP	MPD	DPM	MDP

43	Data store in a DFD represents _____	A sequential file	A disk store	A repository of data	A random access memory	A repository of data
44	_____ system consists of programs data files and documentation	Conceptual	Logical	Physical	None of the above	Physical
45	_____ is a good example of deterministic system	Life cycle	Computer program	Software program	None of the above	Computer program
46	The main ingrediant of the report documenting the _____ is the cost benefit analysis	System analysis	Feasibility study	System analyst	System design	Feasibility study
47	A data cannot flow between a store and i) a store ii) a process iii) an external entity	I and iii	I and ii	ii and iii	ii	I and iii
48	System development process is also called as _____	System Development Life Cycle	System Life Cycle	Both A and B	System process Cycle	System Development Life Cycle
49	DDS stands for _____	Data Data Systems	Data Digital Systems	Data Dictionary Systems	Digital Data Service	Data Dictionary Systems
50	In _____ system the interaction between various subsystems cannot be defined with certainly	Open system	Closed system	Deterministic system	Probabilistic system	Probabilistic system
51	_____ relationship specifies an optional behaviour	A generalization	An inheritance	An include	An aggregation	A generalization
52	The _____ is a construct that helps analysts to work with users to determine system usage	Use case	Actor	Class	Component	Use case
53	Outputs from a use case are described on the use case from along with their corresponding	Data models	Destination	Inputs	Source	Destination

54	Each use case describes how the system reacts to an _____ that occurs to trigger the system	Data flow	Process	Data store	Event	Event
55	The functionality of the sytem or what the information system will do is called the _____ of the system	Business need	Intangibles	Requirements	Sponsors	Requirements
56	RAD stands for _____	Rapid application development tool	Raid application development tool	Research application development	Reasearch aid design tool	Rapid application development tool
57	The process of converting a new or revised system design into an operational one is known as _____.	Testing	Implementati on	Quality Assurance	Design	Implementat ion
58	Which one the following is not a form of Decision Table?	Limited-Entry	Extended-Entry	Mixed-Entry	Double-Entry	Double-Entry

Reg. No.....
(17MBAPS303A)
KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act, 1956)
COIMBATORE-641021
(For the candidates admitted from 2017 onwards)

II MBA – Continuous Internal Assessment – I

SYSTEM ANALYSIS AND DESIGN

Time: 2hours

Maximum: 50marks

PART – A (15 X 1 = 15 Marks)
ANSWER ALL THE QUESTIONS

- 1) User training was
a) Poor b) best c) fast d) medium
- 2) Centres on the existing computer system hardware\ software, etc?
a) Production b) Organization c) Sales d) Objective
- 3) The structure chart derived by studying the flow through the system supports the activity of
a) File design b) Programs design
c) Database design d) internal control design
- 4) Which of the following technique detects transposition errors?
a) Check digit b) Automatic correction
c) Existence test d) duplicate processing

- 5)refers to the holism of systems
a) Integration b) Interaction c) Interdependence d) Information
- 6) The focus on the tools listed earlier in data dictionary
a) Structured graphical b) structured analysis
c) Structured tools d) structured systems
- 7) The objective is to build a new document, called
a) System planning b) system control
c) System development d) system specifications
- 8) In structured analysis, focus on functions rather than the physical are
a) Data dictionary b) decision trees
c) Decision tables d) Data flows
- 9) Is a detailed study of the various operations performed by a system?
a) Feasibility b) strategy c) analysis d) design
- 10) An HIPO diagram derived fromdiagram?
a) Hierarchy b) structured c) information structure d) Vertical
- 11) In specific recommendation regarding the candidate system, including assignments
a) Formal b) personnel c) interpersonal d) semi-personnel
- 12) Relate to the actual purchase or lease of the computer and peripherals?
a) Hardware costs b) software costs
c) personnel costs d) system costs
- 13)..... Include EDP staff salaries and benefits in vacation time
a)personnel costs b)hardware costs
c)software costs d)system costs

14)..... Are expenses incurred in the preparation of the physical site?

- a)operating costs b)personnel costs
- c)facility costs d)benefit cost

15) The largest Percentage of total life cycle cost of software is

- a) Conceptual b) Design costs
- c) Maintenance costs d) Coding costs

PART – B (3 X 8= 24 Marks)
ANSWER ALL THE QUESTIONS

16. a) Explain various stages of SDLC.

(Or)

b) Write the process and steps of system design.

17. a) Detail the Waterfall Model with diagram.

(Or)

b) Describe the Spiral Model and list out the advantages and disadvantages?

18. a) Write about the information gathering techniques.

(OR)

b) What is cost benefit analysis and its Categories.

PART – C (1 X 11= 11 Marks)
CASE STUDY (COMPULSORY)

19. E-Commerce site to enable members to buy and sell Portals/Domains:

- (i) Write the requirements for the domains
- (ii) Explain the analysis and design to the portals
- (iii) Give recommendation and conclusion to the corporation.