Module - 4

Business Application Development Framework

Business Application Development

- Business Application Development involves
  - Developing, acquiring and maintaining the application systems that are critical to the Business Processes.
  - The effective management and control of the critical business systems and they often control critical information assets.

The SDLC, one of the methodologies, involves well defined phases in business application development.

Business Application Development

- The system development life cycle projects are initiated when a business organization visualizes one or more of the following situations:
  - To address a new or an existing business requirement
  - Adoption of a new available technology
  - To fix a problem relating to the existing business or technology

Business Application Development

- Need for Structured Systems Development Methodology

  - Software is a logical intangible product which makes all the difference in estimating, measuring and managing it without a methodology.
  - Software products depend heavily on the quality of people carrying out system development, therefore, clear procedures for the development are required to be communicated and followed by the concerned employees.
  - Without proper and adequate planning, design and testing at the appropriate times for the functionality and controls, the software may not satisfy the user's needs,
  - Without adequate control framework, it may lead to vulnerabilities for the business processes.

System Development Life Cycle

- The primary objectives of any SDLC are to deliver quality systems that:
  - Meet or exceed customer expectations
  - Within cost estimates,
  - Work effectively and efficiently within the current and planned information technology infrastructure
  - Are inexpensive to maintain and
  - Cost-effective to enhance

System Development Life Cycle

- SDLC establishes a logical order of events for conducting system development that is
  - Planned
  - Controlled
  - Complying with prescribed standards
  - Measured
  - Documented and
  - Ultimately improved

  (The SDLC serves as the mechanism to assure that systems under development meet the established requirements and support the organization's mission functions.)

System Development Life Cycle

- Risks associated with SDLC
  - The development team may find it cumbersome
  - The users may feel delays in delivery of end products
  - The rigidity of approach may prolong the duration of many projects
  - It may not be suitable for small and medium sized projects.

System Development Life Cycle

- Risks associated with SDLC
  - User requirements and objectives not being met by the application system
  - Inadequate stakeholder (including internal audit) involvement
  - Lack of management support
  - Inadequate project management
  - Inappropriate technology and architecture
  - Scope variations

System Development Life Cycle

- Risks associated with SDLC
  - Time over-runs
  - Cost over-runs
  - Inadequate quality of the application system
  - Insufficient attention to security and controls (including validations and audit trails) in the application system
  - Performance criteria not being met
  - Inappropriate resourcing/staffing model management

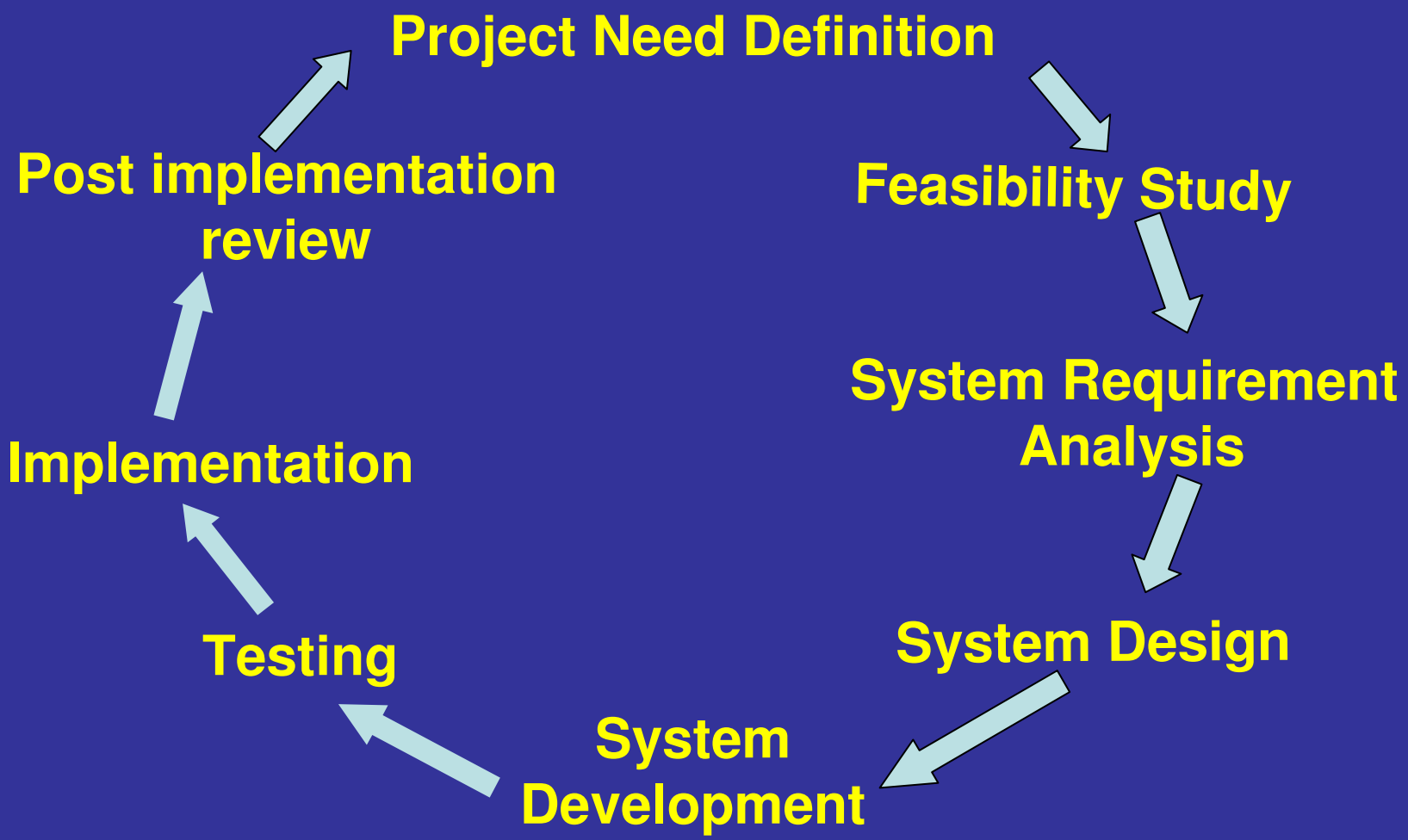
System Development Life Cycle

- Risks associated with SDLC
  - Inadequate staffing skills
  - Insufficient documentation
  - Inadequate contractual protection
  - Insufficient attention to interdependencies on other applications and processes
  - Inadequate configuration management
  - Insufficient planning for data conversion/migration and cutover
  - Post cut-over disruption to business

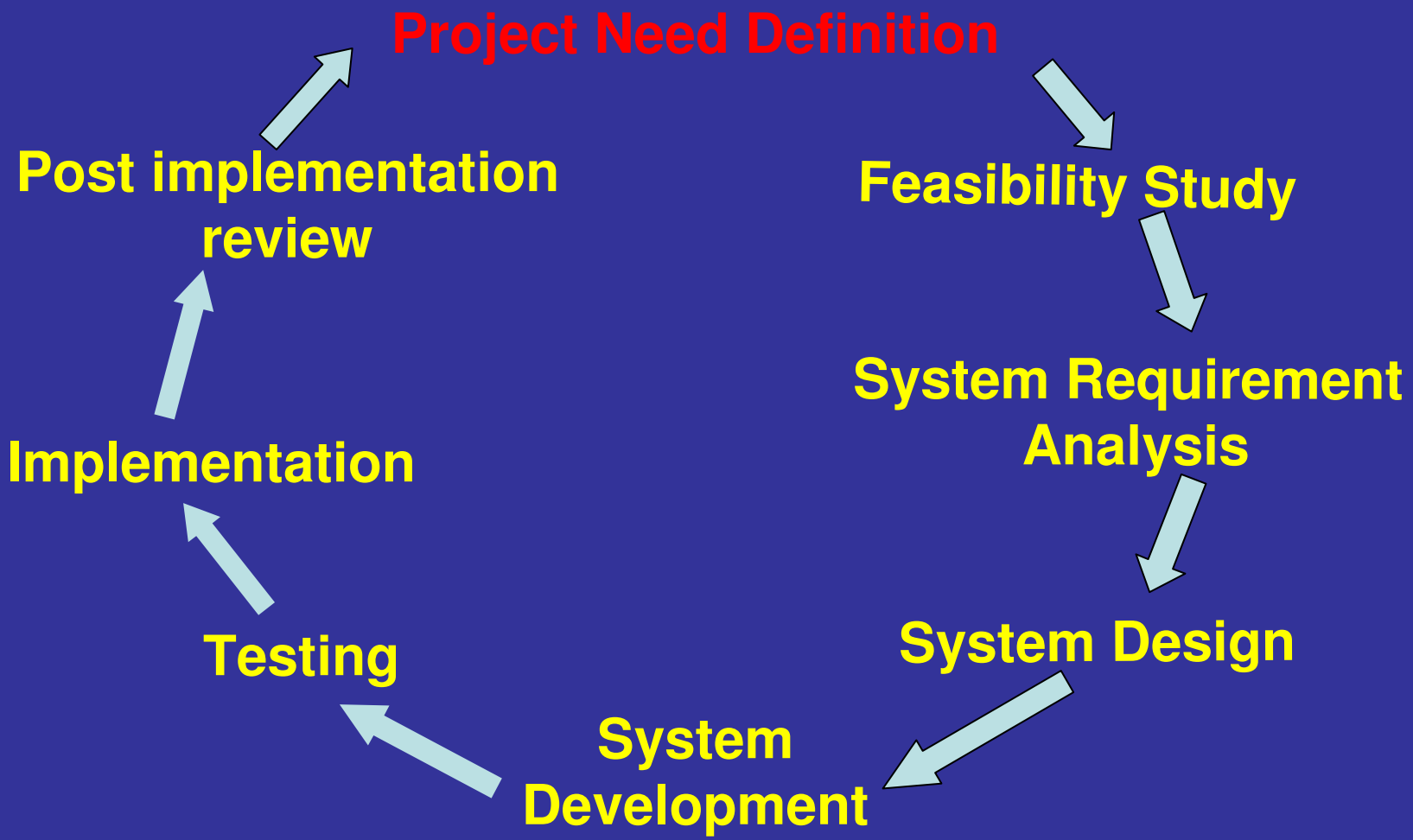
System Development Life Cycle

- From the perspective of the IS Audit, the following are the possible advantages. The IS Auditor:
  - can have clear understanding of the various phases of the SDLC on the basis of the detailed documentation created during each phase of SDLC.
  - on the basis of his examination, can state in his report about the compliance by the IS management of the procedures, if any, set by the management.
  - if has a technical knowledge and ability of the area of SDLC, can be a guide during the various phases of SDLC.
  - can provide an evaluation of the methods and techniques used through the various development phases of the SDLC.

System Development Life Cycle



System Development Life Cycle



System Development Life Cycle

- Project Need Definition: The objective of this step is to clearly define the need for the project. The Primary questions asked at this stage are
  - Is the project required?
  - if yes, what are the broad expectations?

  It involves the following:
    - Recognize the problem
    - Define the problem
    - Set system objectives
    - Identify system constraints

System Development Life Cycle

- Recognize the problem. There has to be an awareness that a problem(s) exists in the existing systems which should not be ignored.
- Define the problem. This involves determining the exact nature of the problem, its location and its likely causes. The objective is to understand the problem to enable finding an appropriate solution.
- Set system objectives. At this point the objectives are set in broad, general terms. They are made specific later.
- Identify system constraints. Some constraints are imposed by the environment and others may imposed by the organisation's management, such as the requirement to use existing hardware or to have the system up and running by a certain date.

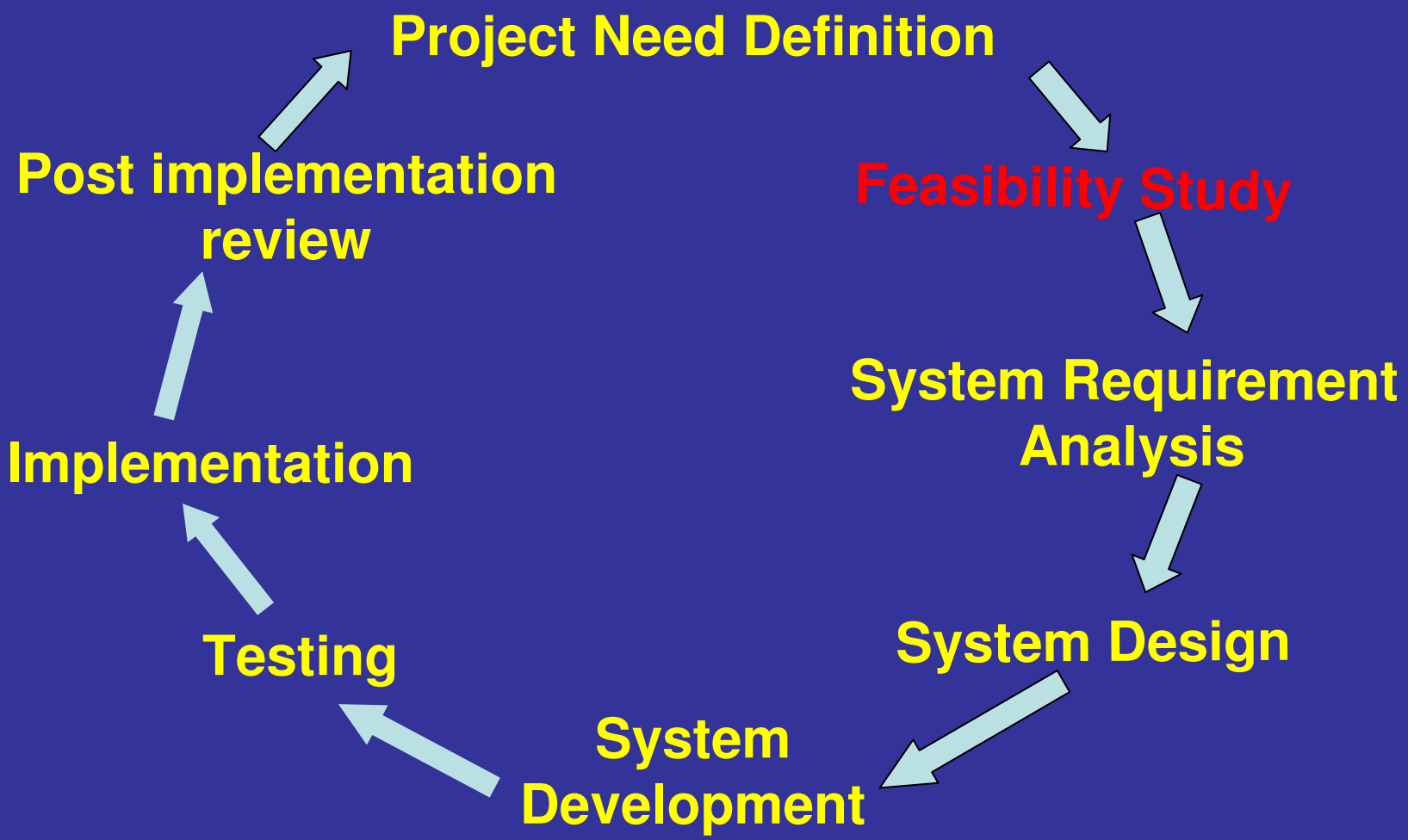
System Development Life Cycle

If it is finally decided that there appears to be a need for the project, a

feasibility study

is conducted

System Development Life Cycle



System Development Life Cycle

- Feasibility Study: This is the analysis of the possibility and the worthiness of undertaking the project. There are several dimensions of feasibility:
  - Technical
  - Economic
  - Financial
  - Non-financial
  - Operational
  - Schedule
  - Behavioral

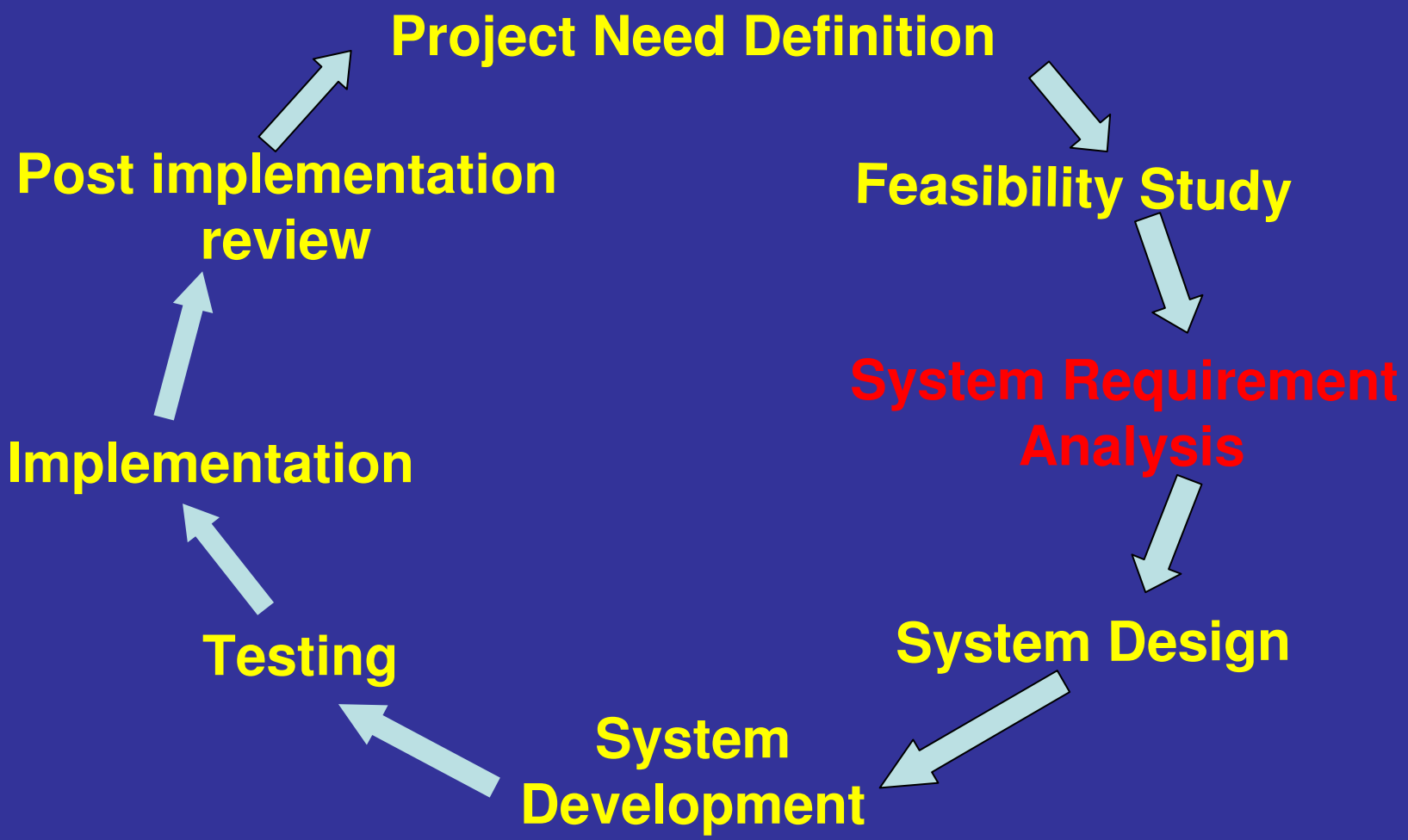
System Development Life Cycle

- Technical: Is there hardware and software available to perform the necessary processing?
- Economic: Can the proposed system be justified monetarily by comparing its benefits and costs?
- Financial: How the funds are to be managed.
- Non-financial: Can the proposed system be justified based on benefits that cannot be measured in monetary terms? Example, more comfortable working environment for computer staff is a non financial benefit.
- Operational: Is the system design such that it will receive the support of the people who must make it work?
- Schedule: Will it be possible to implement the system within the imposed time constraints?
- Behavioral: Will it / will it not adversely effect on the quality of life?
- Legal: Will / will not the system violate the local legal requirements?

System Development Life Cycle

The study is documented and a feasibility study report is prepared. The report would also recommend whether to undertake the project or not to do so. The report is submitted to top management for a decision to go ahead or not to. The decision to go ahead could in terms of develop or acquire. If the decision is to go ahead, then the next step is undertaken.

System Development Life Cycle



System Development Life Cycle

- System Requirement Analysis (SRA): The objective of this step is to thoroughly understand the existing system and determine the users' information and performance requirements
- The main issues considered by the project team at this stage are:
  - Define information needs
  - Define performance criteria

System Development Life Cycle

Define information needs:

- What information is needed?
- Who needs it?
- When is it needed?
- Where is it needed?
- In what form is it needed?

Define performance criteria: What are the acceptable performance standards? For example, a marketing manager who needs a monthly expense report may insist that the report must be available no later than three days after the end of the month.

System Development Life Cycle

- Users' information needs are captured by engaging in a variety of information-gathering activities:
  - personal interviews
  - observations
  - record searches
  - and surveys

System Development Life Cycle

- Entity Relationship Diagram: (A requirement analysis tool)
  - An ERD depicts a system's data and how these data interrelate.
  - An ERD can be used to obtain an understanding of the data a system needs to capture and manage (a logical data model).
  - An ERD can also be used later, as a design tool that helps document the actual database schema that will be implemented (a physical data model).

System Development Life Cycle

- Study of information flows
  - Data Flow Diagrams
  - Flow Charts
  - Decision trees
  - Decision Tables
  - Structured English

System Development Life Cycle

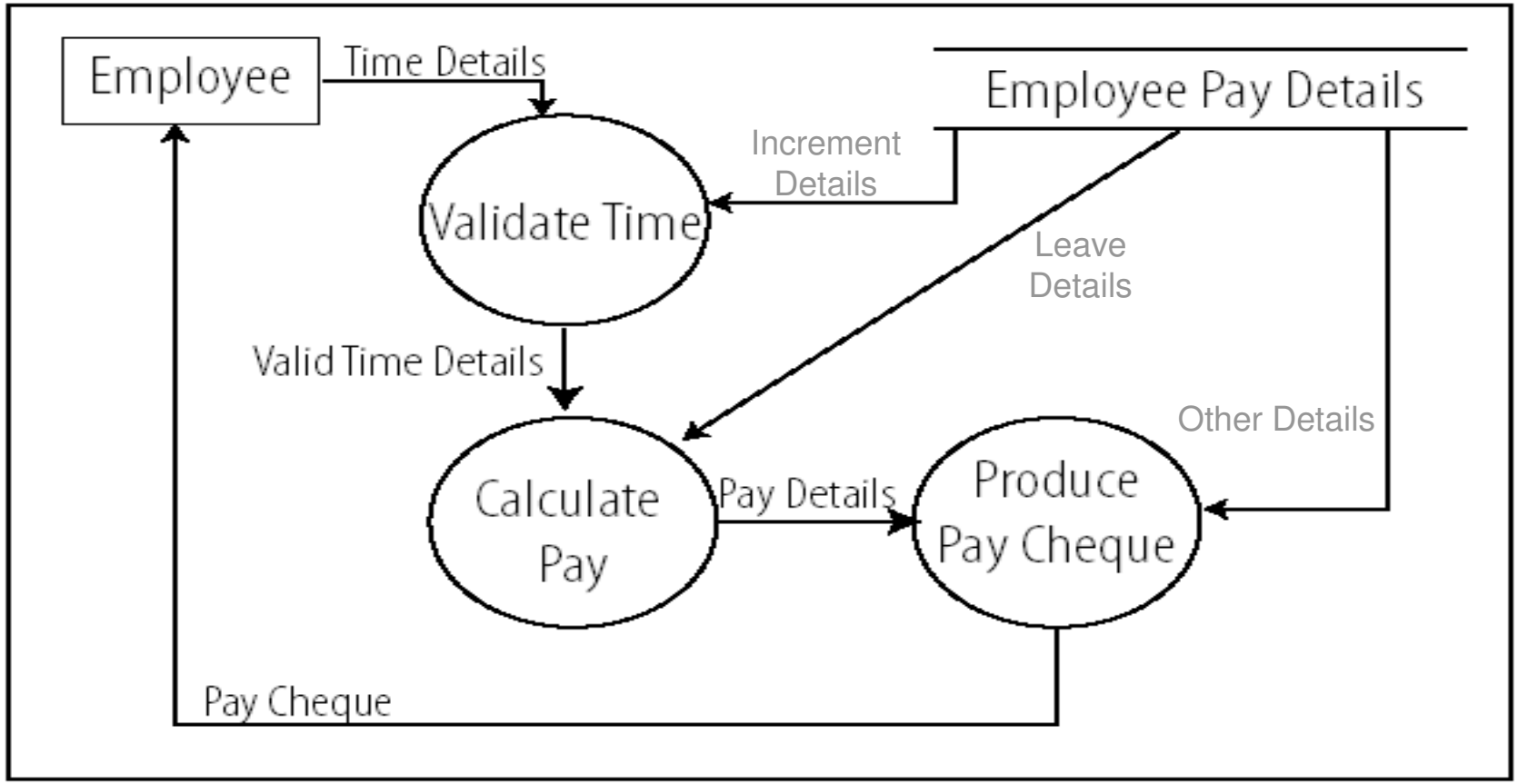
Data Flow Diagram (DFD) is a popular tool for analyzing and representing information flow:

A DFD has four elements:

- Data Flow: represented by arrow
- Process: represented by circle (bubble) or rectangle with rounded edges
- Data Stores: represented by parallel lines or open rectangle
- External Entity: represented by square

System Development Life Cycle

Data Flow Diagram



System Development Life Cycle

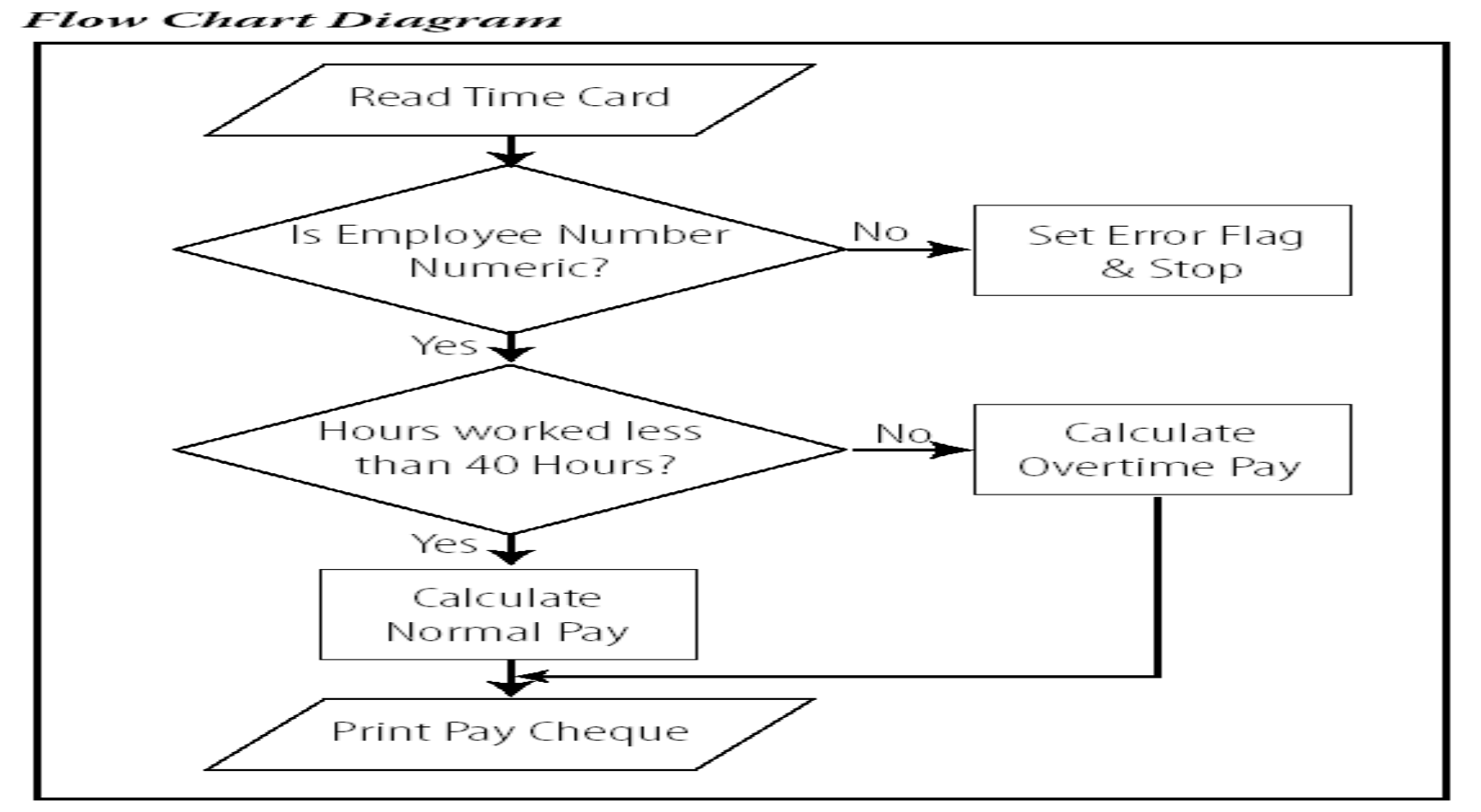
Rules of DFD

- All entities, processes and data stores should be numbered / named
- It is preferable to mention the data flowing on the arrows too.
- DFD is always read from top to bottom and left to right.
- No more than seven processes (five is ideal) should be represented in the same page.

System Development Life Cycle

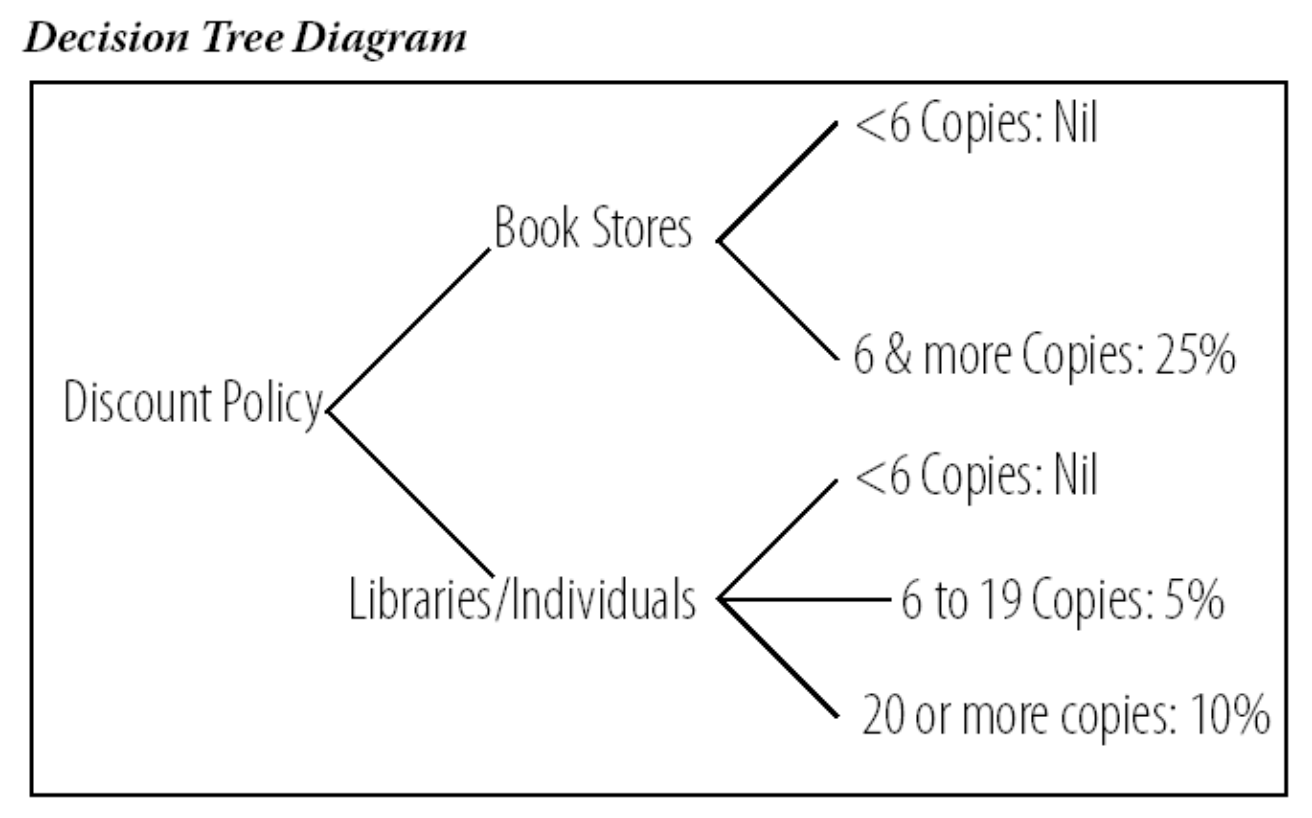
- Flow chart is the most widely-used method for representing logic. In a flow chart
  - A circle represents start and stop
  - A box represents a processing step
  - A rhombus represents a logical condition
  - An arrow represents the flow of information

System Development Life Cycle



System Development Life Cycle

- Decision tree: When multiple conditions operate, the decision tree is a better option.



System Development Life Cycle

- Decision table : When a set of complex conditions needs to be evaluated for selecting appropriate action, then the decision table is the preferred option. The decision table provides conditions and actions in a tabular form.

System Development Life Cycle

Decision table :

Type of customer	**W**	**W**	**R**	**R**
Type of Inventory	**A**	**B**	**A**	**B**
Discount	**10**	**15**	**5**	**10**

System Development Life Cycle

- **Structured English**

IF order is from wholesale customer
IF order is for Type-A inventory item
THEN item-discount is 10%
ELSE order is for Type-B inventory item
item-discount is 15%
ELSE order is from retail customer
IF order is for Type-A inventory item
THEN item-discount is 5%
ELSE order is for Type-B inventory item
item-discount is 10%

System Development Life Cycle

This stage, System Requirement Analysis, concludes with the preparation of a User Requirements Specification (URS) document. This has to be approved by the user representative and by top management

System Development Life Cycle

- Software Acquisition: Software acquisition is not a phase in standard SDLC. But if the business has decided to acquire the software rather than developing the same, software acquisition is the process which should occur after System Requirement Analysis Phase.

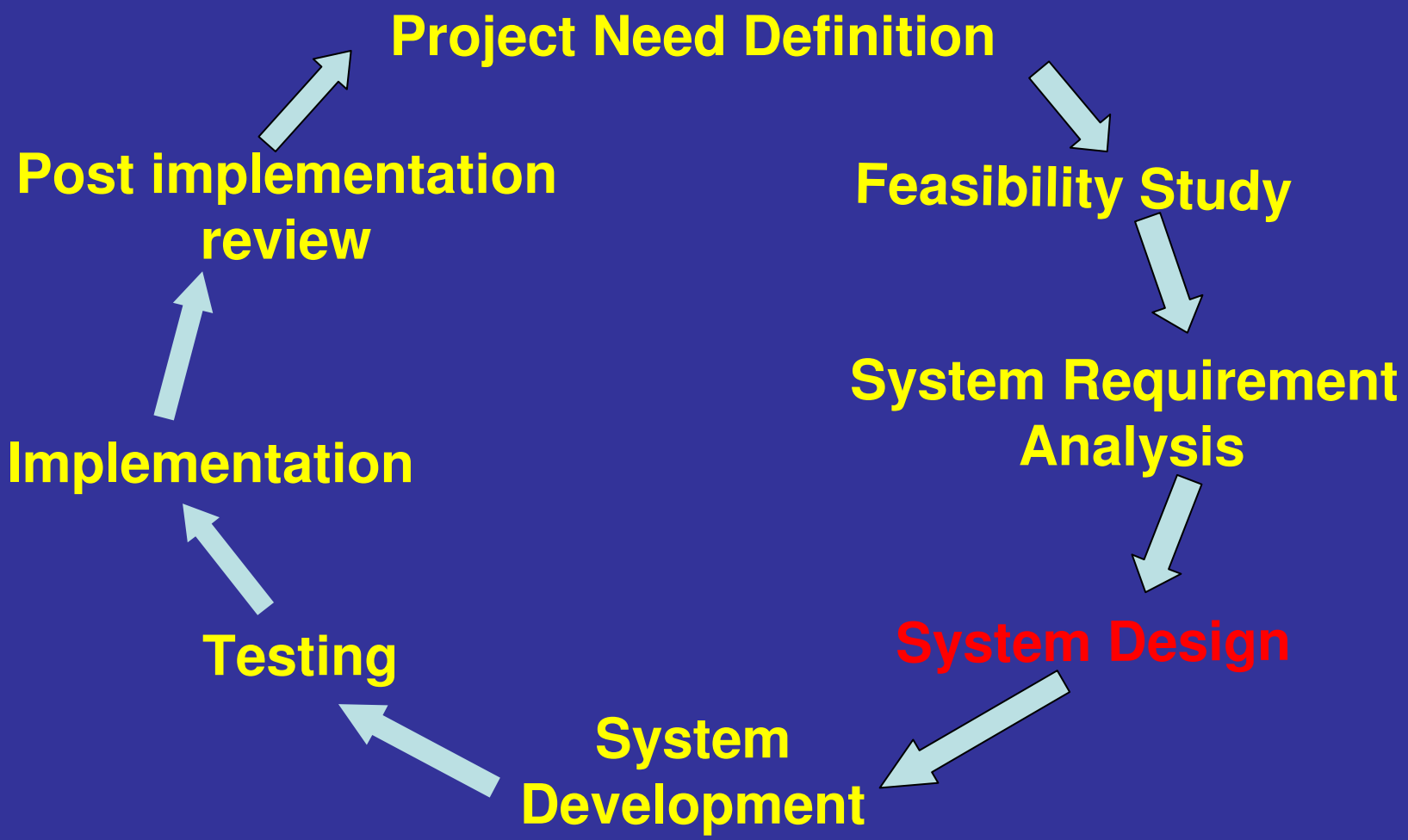
System Development Life Cycle

- The following steps shall be relevant in acquisition of the software.
  - The feasibility study documentation that supports the decision to acquire the software must be available.
  - The decision is based upon various factors like
    - Cost difference between development and acquisition
    - Availability of the required software readily in the market
    - Time saved between development and acquisition etc.
  - A project team comprising of technical supports staff and key users should be crated to prepare Request for Proposal (RFP).
  - Formation of comparative chart based on the examination of the various responses received from the Vendor to the RFP for evaluation
  - Short listing of the vendors on the basis of such evaluation

System Development Life Cycle

- RFP Contents
  - Product vs. Systems Requirements
  - Customer References
  - Vendor viability / financial stability
  - Availability of complete and reliable documentation
  - Vendor Support
  - Source code availability / Escrow arragement
  - Number of years experience in offering the product
  - A list of recent or planned enhacements to the product, with dates
  - Number of client sites using the product with a list of current users
  - Acceptance testing of the product

System Development Life Cycle



System Development Life Cycle

- System Design
  - The system design is meant to be a blueprint of the new IT system.
  - The project team considers and evaluates alternative designs and selects the one that is expected to meet the user requirements most satisfactorily within the given constraints.
  - The output of this stage is the system design document (SDD).

System Development Life Cycle

The SDD includes the following:

- Data flow in the information system
- Database structure
- Security plan
- Hardware and software configurations
- User interface: That is, how the users are expected to interact with the system
- Change control mechanism to prevent uncontrolled entry of new requirements.
- Physical facilities required

The user involvement may be minimal in this phase.

System Development Life Cycle

Key design phase activities include:

- Developing system flowcharts to illustrate how the information shall flow through the system
- Defining the applications through a series of data or process flow diagrams, showing various relationships from the top level down to the detail
- Describing inputs and outputs, such as screen design and reports
- Determining the processing steps and computation rules for the new solution
- Determining data file or database system file design
- Preparing the program specifications for the various types of requirements or information criteria defined

System Development Life Cycle

- Developing test plans for:
  - Unit (Program) Testing
  - Subsystem (Module) Testing
  - Integration (System) testing
- Developing
  - Interface with other systems
  - Loading and initializing files
- Developing Security, backup and recovery plan
- Developing data conversion plans to convert data and manual procedures from the old system to the new system.

System Development Life Cycle

The Design phase deals with the way the proposed system can be transformed into a working model. The sequence of steps involved in this phase are:

- Architectural design
- Design of data / Information flow
- Design of database
- Design of user interface
- Physical design
- Selection of appropriate hardware and software

System Development Life Cycle

Design phase

- Architectural design
- Design of data / Information flow
- Design of database
- Design of user interface
- Physical design
- Selection of appropriate hardware and software

System Development Life Cycle

- Architectural design
  - Architectural design deals with the organisation of applications in terms of hierarchy of modules and sub -modules. At this stage, we identify:
    - Major modules
    - Function and scope of each module
    - Interface features of each module
    - Modules that each module can call directly or indirectly
    - Data received from / sent to / modified in other modules

System Development Life Cycle

Design phase

- Architectural design
- Design of data / Information flow
- Design of database
- Design of user interface
- Physical design
- Selection of appropriate hardware and software

System Development Life Cycle

- Design of data / Information flow
  - In designing the data / information flow for the proposed system, the following inputs are required:
    - Existing data / information flows
    - Problems with the present system
    - Objective of the new system

    For representing the new design, the same tools that were used in analysis phase: Data Flow Diagrams (DFDs) or object-oriented notations are used. The frequency and timing of the flow are also identified in this phase.

System Development Life Cycle

Design phase

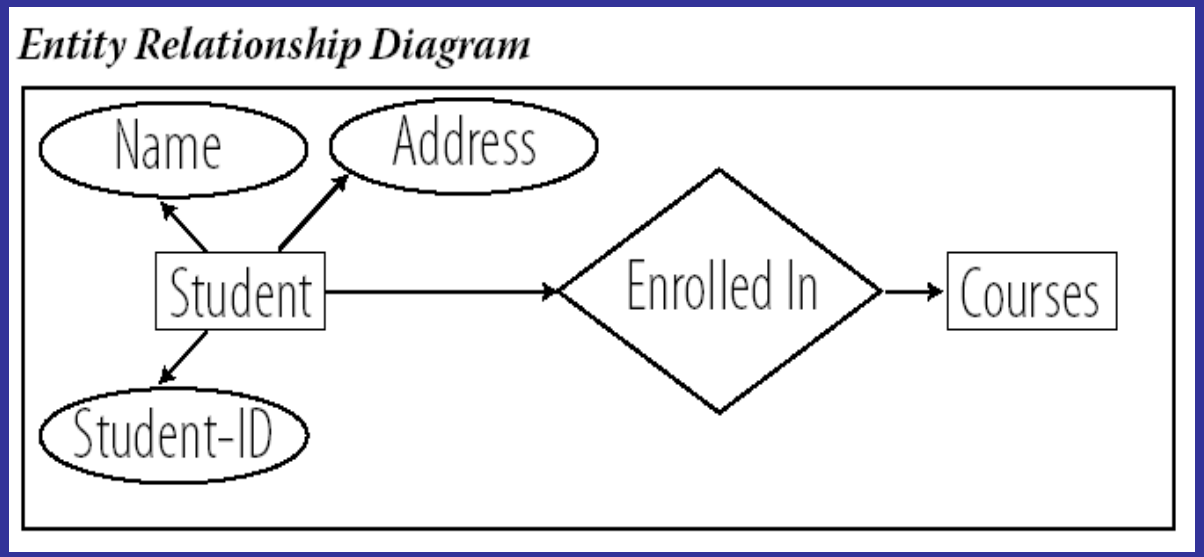
- Architectural design
- Design of data / Information flow
- Design of database
- Design of user interface
- Physical design
- Selection of appropriate hardware and software

System Development Life Cycle

- Design of database
  - At this stage, the scope and structure of the database, centralized or distributed, is decided.
  - The design of the database consists of four major activities:
    - Conceptual modeling
    - Data modeling
    - Storage structure design
    - Physical layout design

System Development Life Cycle

- Conceptual modeling
  - In this activity, the application is modeled in terms of entities / objects and their relationships.
  - Every object or event in the system is known as an entity.



System Development Life Cycle

- Data modelling
  - Conceptual models have to be translated into data models, so that they can be manipulated using programming languages.
  - Every entity identified in the conceptual model is converted into a table, and the attributes are converted into fields (columns in the table).

EMP NAME	EMP ID
Kamalakar	A1005
Prafulla	A1006
Vinod	A1007

System Development Life Cycle

- Storage structure design
  - During this activity, a decision is made on how to store the data structure on some device.
- Physical Layout Design
  - At this stage the detailed implementation to the level of storage device for storage of data is planned.

System Development Life Cycle

Design phase

- Architectural design
- Design of data / Information flow
- Design of database
- Design of user interface
- Physical design
- Selection of appropriate hardware and software

System Development Life Cycle

- Design of user interface
  - Some points that should be considered while designing the user interface design:
    - Level of user - educational background, computer exposure
    - Source document format
    - Output requirements
    - Enquiry screens
    - Graphic and color displays
    - Requirement for special input / output device

    Prototypes can be used to elicit user feedback and refine the user interface.

System Development Life Cycle

Design phase

- Architectural design
- Design of data / Information flow
- Design of database
- Design of user interface
- Physical design
- Selection of appropriate hardware and software

System Development Life Cycle

- Physical Design: At this stage, the logical design is converted into physical implementation units.

  Some of the issues addressed here are:

  - Type of hardware for client application and server application
  - Operating systems to be used
  - Type of networking
  - Processing type batch, online, real-time
  - Frequency of input, output
  - Month-end cycles / periodical processing

System Development Life Cycle

Design Principles: The following principles should be kept in mind, while designing the system

- More than one alternatives should be designed and the best one on pre-specified criteria should be chosen.
- The design should be based on the analysis.
- The software functions designed should be directly relevant to business activities.
- The design should follow standards laid down. For instance, the user interface should have consistent color scheme, menu structure, location of error message and the like.
- The design should be modular.

System Development Life Cycle

Modularity

- A module :
  - is a manageable unit performing a well- defined task.
  - Each module contains data and instructions for the task.
  - Modules have well defined interfaces.
  - Interaction among modules is based on interfaces,.
  - Modularity is measured by two parameters: Cohesion and Coupling
- Cohesion:
  - Refers to the manner in which elements within a module are linked.
  - A module that performs one discrete task is said to be more cohesive than a module that performs multiple tasks. Cohesion reflects the strength of bonding between elements of a module.
- Coupling:
  - Coupling is a measure of the interconnection between modules.
  - It refers to the number and complexity of connections between ‘calling’ and ‘called’ modules.

In a good modular design, cohesion will be high and coupling low.

System Development Life Cycle

- Roles Involved in SDLC
  - *Steering Committee*
  - *Project Manager*
  - *Project leader*
  - *Systems Analyst*
  - *Module leader/Team leader*
  - *Programmers*
  - *Database Administrator (DBA)*
  - *Quality Assurance*
  - *Domain Specialist*
  - *Technology Specialist*
  - *Documentation Specialist*
  - *IS Auditor*

System Development Life Cycle

- The functions of Steering Committee are:
  - To Provide overall direction and ensures appropriate representation of all concerned parties
  - To be responsible for all cost and timetables
  - To conduct a regular review of progress of the project in the meetings of steering committee which may involve co-ordination and advisory functions, taking corrective actions like rescheduling, re-staffing, change in the project objectives and need for redesigning.

System Development Life Cycle

- Project Manager
  - This is a middle management function, and he is responsible for delivery of the project within the time and budget
  - He periodically reviews the progress of the project with the project leader and his team
  - Project leaders report to him
  - He apprises the Steering Committee about the progress of the project

System Development Life Cycle

- Project leader
  - He is dedicated to a project and has to ensure its completion and fulfillment of objectives
  - The entire project team reports to him. He reviews the project position more frequently than a Project Manager
- Systems analyst
  - His main responsibility is to conduct interviews with users and understand their requirements
  - He is a link between the users and the programmers
  - He converts the users requirements in the system requirements
  - He plays a pivotal role in the Requirements analysis and Design phase

System Development Life Cycle

- Module leader / Team leader
  - A project is divided into several manageable modules, and the development responsibility for each module is assigned to module leaders
  - The module leaders will have programmers reporting to them
  - Module leaders are responsible for the delivery of tested modules within the stipulated time and cost

System Development Life Cycle

- Programmers
  - Programmers are the masons of the software industry
  - They convert design into programs by coding using programming language
  - They are also referred to as coders or developers
  - Apart from developing the application in a programming language, they are also responsible for first level testing of the programs they develop and debugging activity

System Development Life Cycle

- Database Administrator (DBA)
  - The DBA is responsible for maintaining database in a database environment so that the database supports the application program
  - He ensures the integrity and security of information stored in the database
  - He also helps the application development team in database performance issues
  - Inclusion of new data elements has to be done only with the approval of the database administrator

System Development Life Cycle

- Quality assurance
  - This team sets the standards for development, and checks compliance with these standards by project teams on a periodic basis
- Testers
  - Testers are junior level quality assurance personnel attached to a project
  - They test programs and subprograms as per the plan given by the module / project leaders and prepare test reports

System Development Life Cycle

- Domain specialist
  - Whenever a project team has to develop an application in a field that's new to them, they take the help of a domain specialist
  - A domain specialist need not have knowledge of software systems
- Technology specialist
  - They are experts in specific technology areas, such as Microsoft technology, Web-enablement and the like
  - If the application development team needs guidance in any of these areas, then it could associate with experts on a part-time / full-time basis

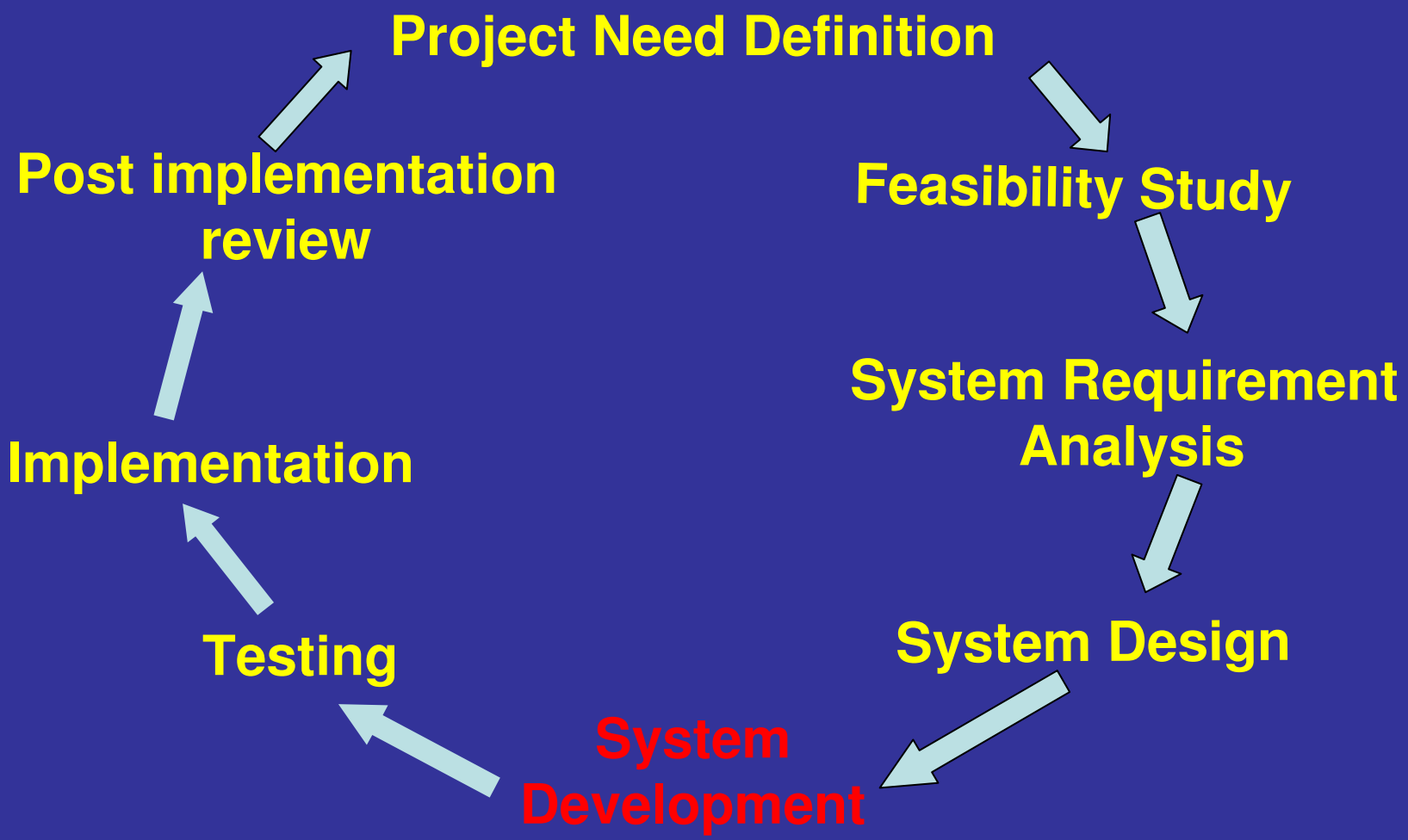
System Development Life Cycle

- Documentation specialist
  - These professionals are responsible for the creation of user manuals
  - They are expected to understand the functions of the software, and provide documentation that makes it easy for the user also to do so
  - They should also have a good command of the English language, and be able to simplify complex concepts

System Development Life Cycle

- IS auditor
  - Ideally an IS auditor should be part of the development process
  - As a member of the team, he can ensure that the application development also focuses on the control perspective
  - He should be involved at the Design Phase and the final Testing Phase to ensure the existence and the operations of the Controls in the new software

System Development Life Cycle



System Development Life Cycle

- Development phase:
  - The objective of the Development Phase is to convert the deliverables of the Design Phase into a complete information system
  - The activity addresses the computer programs that make up the system
  - Also puts in place the hardware, software, and communications environment for the system
  - At the end of this phase, the system will be ready for the activities of the Test Phase.

System Development Life Cycle

- Development Phase: The following are the key activities performed during this phase.
  - Coding and developing program and system level documents
  - Testing and debugging continuously for improvements in programs developed
  - Developing programs for conversion of the data in the legacy system to new system
  - Training the selected users on the new system
  - In case of vendor supplied software, documenting the modifications carried out to ensure that future updated versions of the vendor's code can be applied appropriately

System Development Life Cycle

Characteristics of a good program

- Accuracy: Should not do what it is not supposed to do
- Reliability: No hard coding
- Robustness: Exception handling capabilities
- Efficiency: Performance
- Usability: User friendly
- Readability: Provided with comments
- Maintainability: Easier to maintain
- Modularity: Separate module for separate task

System Development Life Cycle

Programming Methods & Techniques

- Adoption of the Program Coding Standards
  - Easier understanding of programs
  - Data declaration , input/output standards
  - Easier communication among developing teams
  - Helps avoiding setbacks due to programmer turnover
  - Efficient maintenance

System Development Life Cycle

- Structured programming
  - Single entry, single exit
  - No jumps
  - Top down / Bottom up approach
- Online Programming Facilities
  - Faster development, lower development cost
  - Faster maintenance
  - Risk of multiple versions of the programs
  - Risk of unauthorized access and updating of programs

System Development Life Cycle

- Programming Language
  - Programming languages have also evolved through generations from the machine languages to high level languages
  - High level (third generatioin) languages specify the sequence in which the computer will execute instructions to arrive at a result. In other words, these languages require the programmer to specify what he wants to achieve, and how he wants to achieve
  - These are called procedural languages
  - In fourth- generation languages like SQL, a programmer is required to specify only what he wants and not how. That is why these are called non-procedural languages

System Development Life Cycle

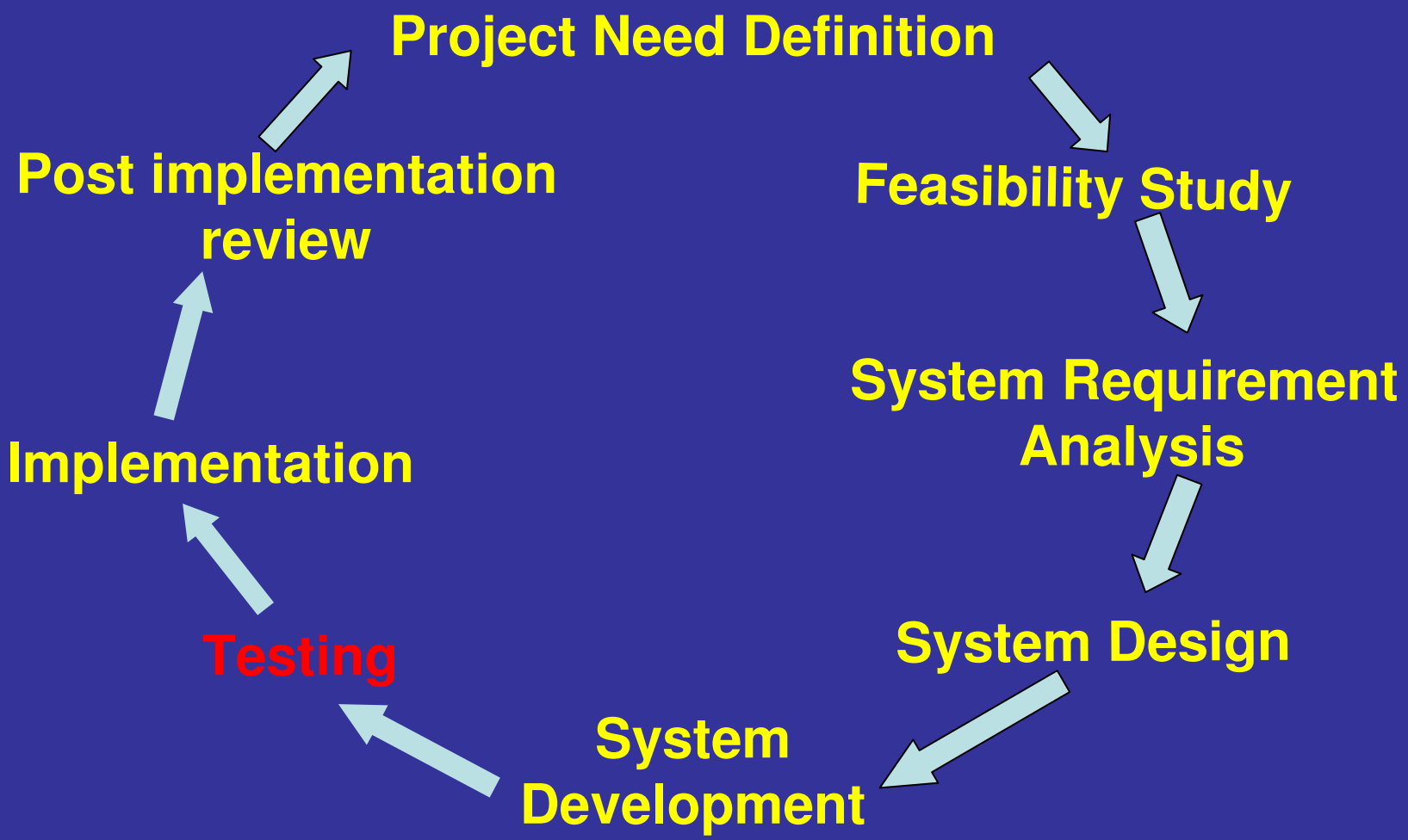
- Features of 4GLs
  - Their functions can be driven by an event like a mouse click
  - They are portable across hardware environments
  - They have highly user-friendly syntax: the syntax is as near English statements as possible
  - They have utilities for screen painting and report generation
  - They can produce graphical outputs
  - They have powerful querying capability
  - However where response time or batch- execution time is the criterion, they could be somewhat slower
  - They are not very good in handling a large number of transactions

System Development Life Cycle

Some issues relating to programming

- Choice of programming language
  - Application area
  - Algorithmic complexity
  - Environment in which software has to be executed
  - Performance consideration
  - Data structure complexity
  - Knowledge of software development staff
  - Capability of in-house staff for maintenance
- Coding style
  - Naming convention
  - Indentations
  - Comment lines
  - Single instruction per line
- Program Debugging
  - Logic path monitors
  - Trace
  - Memory Dumps

System Development Life Cycle



System Development Life Cycle

- Testing Phase:
  - Software testing is the process of testing software in a controlled manner to ensure it meets the requirements and find if there are any logical errors
  - A good test case is one that has high probability of finding an error
  - A successful test is one that uncovers an error

System Development Life Cycle

Levels of Testing

- Unit Testing
- Integration Testing
- System Testing

System Development Life Cycle

- Unit Testing
  - Functional tests: The test plan specifies operating conditions, input values, and expected results, then it checks by inputting the values to see the actual result and expected result match
  - Performance tests: To verify the response time, the execution time, the throughput, primary and secondary memory utilization and the traffic rates on data channels and communication
  - Stress tests: To verify the load bearing capacity of the system
  - Structural tests: These are concerned with examining the internal processing logic of a software system
  - Parallel tests: To compare the result of the manual and automated system with same set of test data

System Development Life Cycle

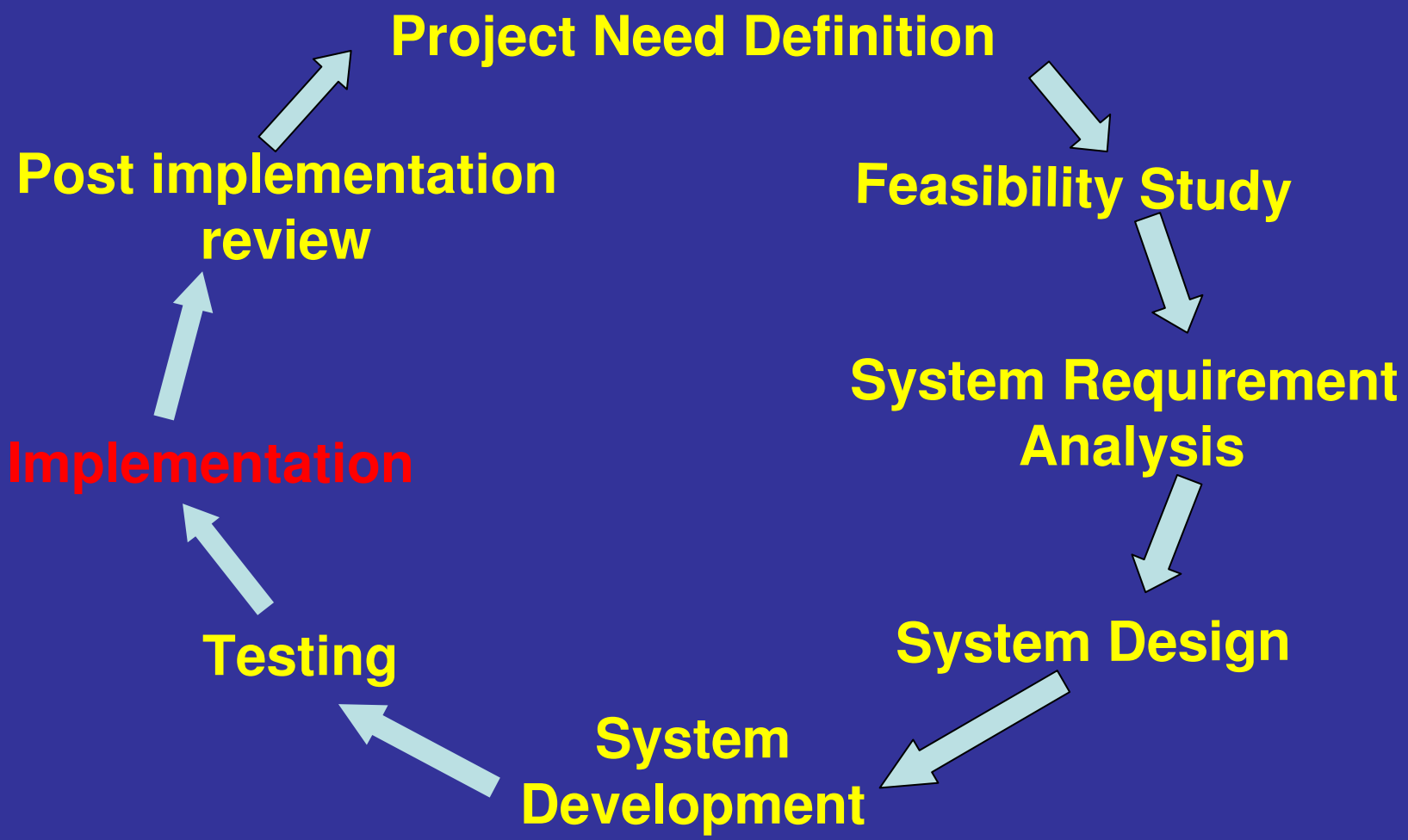
Types of unit tests

- Static analysis tests
  - Desk Check
  - Structured walk-through
  - Code inspection
- Dynamic analysis tests
  - Black Box Test
  - White Box Test

System Development Life Cycle

- System testing
  - Integration/interface testing
    - Bottom-up integration
    - Top-down integration
    - Regression testing
  - Acceptance testing
    - Involves the planning and execution of functional tests, performance tests and stress tests
    - Quality assurance testing
    - User acceptance testing

System Development Life Cycle



System Development Life Cycle

- Implementation

  There are four types of implementation strategies:

  - Direct implementation / Abrupt change-over
  - Parallel implementation
  - Phased implementation
  - Pilot implementation

System Development Life Cycle

Activities during Implementation Stage

- Installation of new hardware / software
- Data conversion
- User training

System Development Life Cycle

Installation of new hardware / software

- If the new system interfaces with the other systems or is distributed across multiple software platforms, some final commissioning tests of the production environment may be desirable to prove end to end connectivity.

System Development Life Cycle

Data conversion

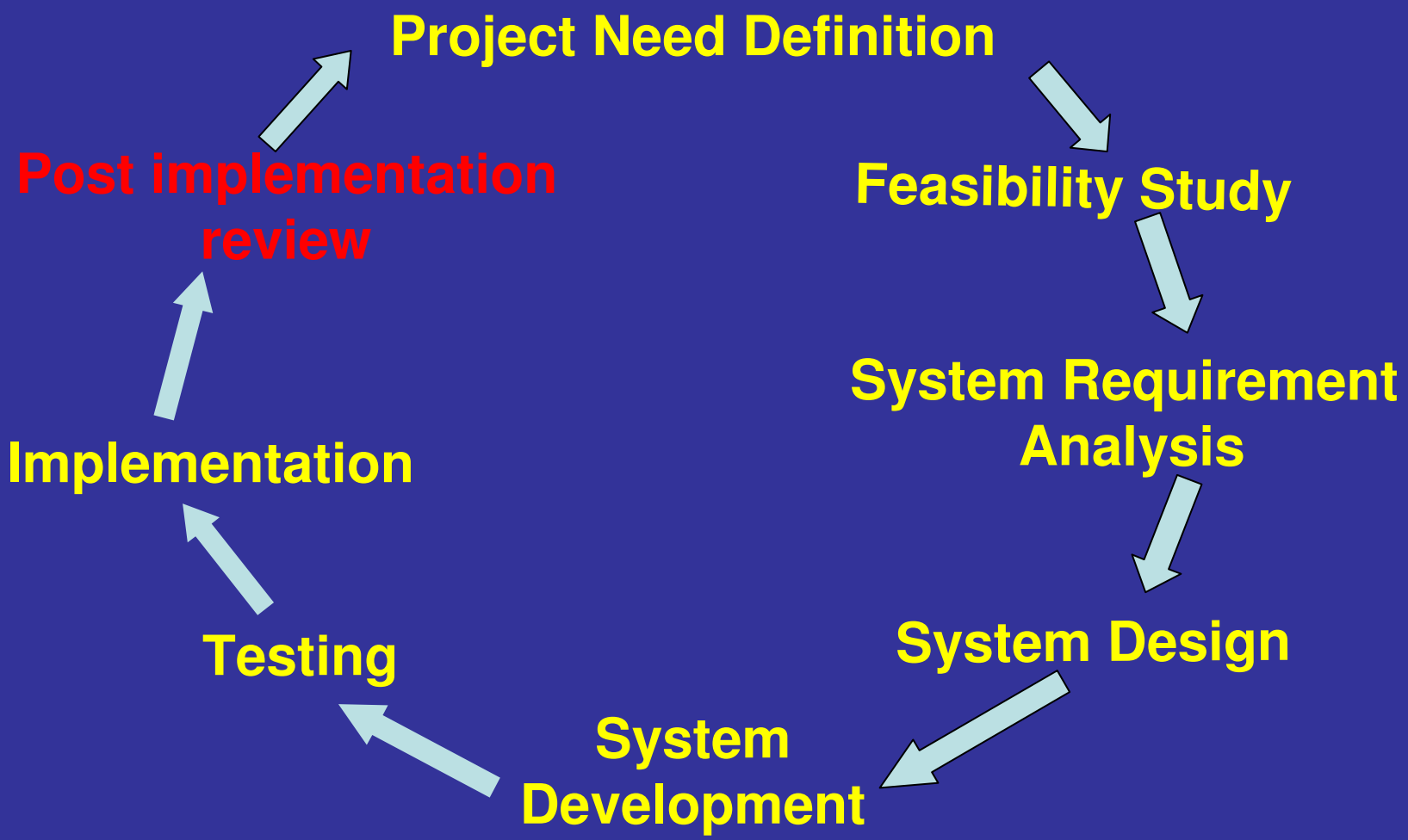
- Determining what data can be converted through software and what data manually.
- Performing data cleansing before data conversion
- Identifying the methods to access the accuracy of conversion like record counts and control totals
- Designing exception reports showing the data which could not be converted through software.
- Establishing responsibility for verifying and signing off and accepting overall conversion by the system owner
- Actual conversion

System Development Life Cycle

User training

- Manager's training on overview and MIS
- Operational user training on how to use the software, enter the data, generate the output
- EDP training on the technical aspects

System Development Life Cycle



System Development Life Cycle

- Post Implementation Review

  An independent group not associated with the development process, either internal or external, should carry out the audit, to meet the following objectives:

  - Assess the adequacy of the system:
    - Whether the system met management's objectives and user requirements
    - Whether the access controls have been adequately implemented and actually working

System Development Life Cycle

- Evaluation and comparison of the actual Cost Benefit or ROI as against the same projected in the feasibility study phase
- Recommend on the system's inadequacies and deficiencies
- Develop a plan for implementing the accepted recommendations
- Evaluate the system development project process

System Development Life Cycle

Maintenance is also part of the post implementation review. It can be categorized into four types:

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Preventive maintenance

System Development Life Cycle

- Umbrella activities

  In addition to the activities associated with each phase, there are many others which are undertaken throughout the life cycle. Some of these activities are:

  - Project tracking and control
    - Budgeting
    - Reporting
    - Review
    - Signoffs
  - Quality assurance
  - Configuration management
  - Documentation
  - Change management
  - Reusability management
  - Measurement
  - Risk management

Alternative Methodologies of Software Development

There have been basically 3 approaches in information system development area:

- process-oriented,
- data-oriented and
- object-oriented approaches.

Alternative Methodologies of Software Development

- Process Oriented
  - Focuses on processes rather than data
  - Processing is optimised
  - Does not focus directly on data
  - Traditional programming approach

Alternative Methodologies of Software Development

- Data Oriented

  Method for representing software requirements

  - focuses on data and its structure (not data flow)
  - considers data independently of the processing that transforms the data
  - optimize data usage
  - Examples: Management Information Systems (MIS) and Data Warehousing applications

Alternative Methodologies of Software Development

- Object-oriented
  - Combines data and processes (called methods) into single entities called objects
  - Objects usually correspond to the real things an information system deals with e.g. customers, suppliers, contracts, rental agreements etc.
  - Conceptually, an object is something that has a fixed or defined set of properties. For an employee as an object, the properties are employee name, employee ID etc.
  - It is easier to establish relationships between any two entities of a program, whether it is a database or an application program,

Alternative Methodologies of Software Development

- Object-oriented approach
  - It is easier to associate two similar objects
  - It allows inheritance of properties from one object to create another, which simplifies the process of creating newer objects with all attributes as the older one, and some additional attributes
  - Can represent complex relationships
  - Can represent data and processing with a consistent notation
  - System elements become more reusable

Alternative Methodologies of Software Development

Applications that use object – oriented technology are:

- Web applications
- E-business applications
- Computer-aided software engineering (CASE) for software development
- Office automation for e-mail and work orders
- Artificial intelligence
- Computer-aided manufacturing (CAM) for production and process control

Alternative Methodologies of Software Development

- Prototyping -
  - When a customer defines a set of general objectives for the software, but not detailed input, processing and output requirements, prototyping may be the best approach.
  - The following are the steps in the prototyping approach:
    - Requirements gathering: The developer gets the initial requirements from the users
    - Quick design: The emphasis is on visible aspects such as input screens and output reports

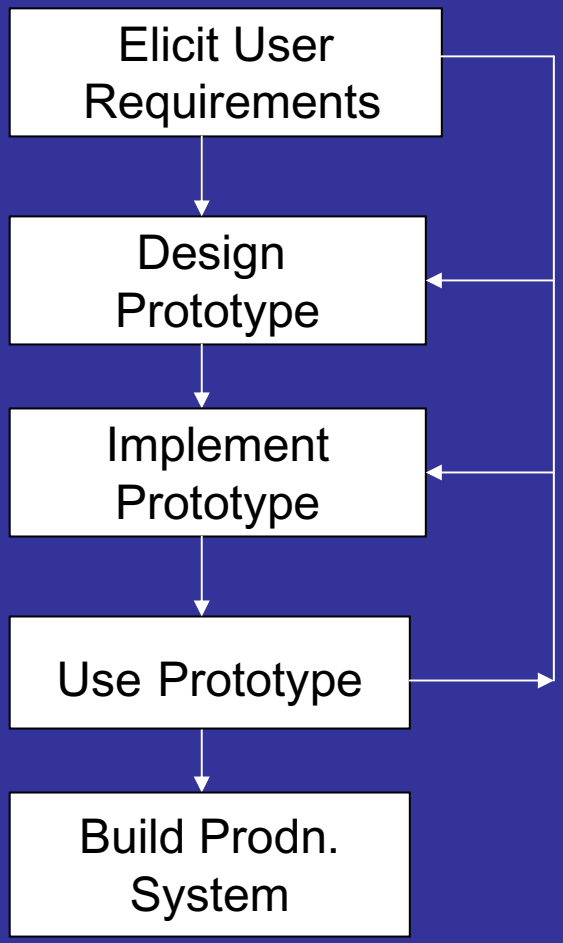
Alternative Methodologies of Software Development

- Prototyping
  - Construction of prototype: by the developer on the basis of inputs from the users
  - Users evaluation of prototype: The users accepts the screens and options as shown to them
  - Refinement of prototype: Prototype is refined by fine tuning the users requirements
  - The last two steps are iterated till the user is fully satisfied with the prototype

Alternative Methodologies of Software Development

- Prototyping
  - A process of creating a system
    - Through controlled trial & error.
    - To reduce the level of risks in developing.
    - It enables developer & customer to understand and react to risks at each evolutionary level.
    - It combines best features of SDLC by maintaining the systematic step-wise approach, but incorporates into it an iterative framework that more realistically reflects the real world

Alternative Methodologies of Software Development



Prototyping

Alternative Methodologies of Software Development

Prototyping – Disadvantage

- Often leads to functions or extras being added to system that are not included in the initial requirements document
- All major enhancements beyond original need to be reviewed to ensure that they meet the strategic needs of the organization and are cost effective, else the final system can end up functionally rich but inefficient
- Finished system may have poor controls, focussing mainly on user needs, controls like backup, recovery, security and audit trails may be missed out
- Change control often becomes much more complicated with prototype systems. Changes in design and requirements happen so quickly that they are seldom documented or approved and hence it can become unmaintainable

Alternative Methodologies of Software Development

- Rapid Application Development (RAD)

  Has a short development cycle. Uses the following techniques to reduce the development time
  - Multiple small trained development teams
  - Modular applications
  - Evolutionary prototype
  - Automated tools
  - Design workshops
  - Component- based development
  - Fourth generation languages
  - Rigid time frames

Alternative Methodologies of Software Development

The four stages in RAD are:

- Definition of scope
  - Defines the business functions & data subject areas that the system will support
- Functional design
  - Uses workshops to model the system's data & processes
  - Builds a working prototype of critical system components
- Development
  - Completes the construction of the physical database and application system
  - Builds the conversion system
  - Develops user aids and deployment of work plans
- Deployment
  - Includes final user testing and training
  - Data conversion and implementation of the application system

Alternative Methodologies of Software Development

The drawbacks of RAD are:

- For large projects, it requires a lot of human resources creating management problems
- It is a joint effort between the customer and developer. Weak commitment on either side may result in failure.
- Not suited for mission critical applications, where quality and reliability assume higher importance than time of development.

Alternative Methodologies of Software Development

- Reengineering

It is a process by which a new system is created / updated by extracting and reusing the design and program component of old sytem. Reengineering is a good alternative to the maintainance work

Alternative Methodologies of Software Development

Reasons for Reengineering:

Difficulty of maintenance of old applications because of:

- outdated technology
- lack of vendor support
- lack of knowledge of programming languages
- difficult to migrate old huge mission critical applications to new systems quickly

Alternative Methodologies of Software Development

Activities involved in reengineering

- Inventory analysis
- Document restructuring
- Reverse engineering
- Code restructuring
- Data restructuring
- Forward Engineering

Alternative Methodologies of Software Development

- **Inventory analysis:**
  - Every organisation should have an inventory of all applications that it uses
  - This should include details such as size, age, business criticality
  - Applications can be sorted on the basis of business criticality, age, current maintainability and other locally important criteria
  - Then applications for reengineering can be selected

Alternative Methodologies of Software Development

- **Document restructuring:**
  - In many legacy applications, documentation is sketchy, or may not exist at all
  - Static applications can be left with current level of documentation
  - While for dynamic applications, the documentation is updated, as and when the program is being modified
  - For critical applications, the documentation may be reworked completely

Alternative Methodologies of Software Development

- **Reverse engineering:**
  - This is the technique of drawing design specifications from the actual product by studying its source code
  - In software reverse engineering, the program is first analyzed and then design specifications are worked out
- **Code restructuring:**
  - After source code analysis, a list of programs is made, which are difficult to maintain because of non-standard coding
  - These programs are restructured, tested and updated

Alternative Methodologies of Software Development

- **Data restructuring:**
  - In this stage, the data files / database is reviewed and restructured
  - Since data architecture has a strong influence on program architecture, code level changes will follow data restructure
- **Forward Engineering:**
  - In this stage, the design decided upon in the reverse engineering stage is analyzed and perfected

Alternative Methodologies of Software Development

Recommended steps:

- Selection of programs likely to be used very soon
- Estimate the maintenance cost of each program
- Rearrange the program in order of importance and maintenance cost
- Calculate the cost of re-engineering the programs
- Compare the maintenace cost and reengineering cost
- Calculate the time required to show its economy
- Consider the advantages of re-engineered system.
- Approval of management to be got
- Subsequent plans for re-engineering of other programs

Alternative Methodologies of Software Development

- Structured Analysis

This approach provides a framework for representing the physical components (data & process) of an application using various graphical notations at different levels of abstraction , until it reaches the abstraction level that enables programmers to code the system

Alternative Methodologies of Software Development

- Strenghts
  - It is an efficient and easy to understand documentation and communication tool between users and developers
  - Automation is easy
  - CASE tool support is available
- Weaknesses
  - Time intervals between tasks are not represented
  - It does not offer inputs for designing concurrent tasks
  - Conditions and selection cannot be represented

Alternative Methodologies of Software Development

- Web based Application Development

Attributes of the Web based applications.

- Network Intensive
- Content Driven
- Continuous evolution

Alternative Methodologies of Software Development

Web based Application categories

- Informational: Read only content
- Download: User can download information
- Customization: The user customizes contents to specific needs
- Interactive: chat-room, bulletin boards, or instant messaging
- User Input: Forms based input is the primary mechanism for communicating need

Alternative Methodologies of Software Development

- Transaction oriented: The user makes a request (e.g. places an order) that is fulfilled by the web based application
- Service Oriented: The application provides a service to the user (e.g. assists the user in calculating the EMI of loan)
- Portal: The application channels the user to other web content or services outside the domain of the portal application
- Database Access: The user queries a large database and extracts information
- Data Warehousing: The user queries a collection of large databases and extracts information

Alternative Methodologies of Software Development

Step involved in development

- Identification of the goals and objectives of development
- Estimation of overall project cost
- Evaluation of risks associated
- Analyzing and establishing technical requirements for the application
- Defining a development schedule
- Identification of the content items
- Page generation using automated tools
- Testing to ensure navigation and uncover errors in function and content

Information Systems Maintenance Practices

- Systems are seldom static after they are moved into production enviornment. Change is the only thing constant in every system.
- To control the on-going maintainance a standard methodology for performing and recording changes is neccessary .
- System maintainance controls refer to the process of modifying application program, based on needs of organization, while maintaining the integrity of both the production source and executable code.

Information Systems Maintenance Practices

- Change Management
  - An appropriate level of management, an inidividual or committee called change control authority, should authorize changes of production programs even if system developement/maintenance staff initiates the change
  - Users should convey requests to system management through some formal correspondence viz. E-mail etc
  - User request should include name, dates, priority & description of change. They could also provide reason, cost justification and expected benefits of change. In addition, the request should provide evidence that it has been reviewed and authorized by user management.

Information Systems Maintenance Practices

- A unique control number to each request could allows to track the status of request
- The user request is then assessed by the relevant application developer who does the following tasks:
  - Evaluates the impact of the modifications on other programs
  - Estimates the number of days required to make all the necessary changes
  - Estimates time required for testing
  - Estimates time required for changing documentation.
  - Prepares a report on the basis of time and cost of change.

Information Systems Maintenance Practices

- The CCA then reviews the report and either authorizes or rejects the change
- If approved, the programmer then makes the approved changes, and the programs are tested
- If a program change warrants a change in the database, then it is first made in the in the test data base
- All related documents are then changed
- The CCA then reviews the changes made to programs, data and documents

Information Systems Maintenance Practices

- After it is approved, the changed version is moved into the production directory
- All users are informed of the change and the revised version number
- Maintenance records of all program changes should exist either manually or automatically (Library mgmt. software)
- Programmers should not have write, modify or delete access to production data, sometimes not even read-only(credit-card, social secy.)

Information Systems Maintenance Practices

- Continuous update of systems documentation

  To effectively maintain systems, documents should reflect the current version of the software. Documents that have to be kept updated are:

  - System manual
    - Input screens
    - Output reports
    - Program specifications
    - Flow charts, Decision tables, Decisions trees

Information Systems Maintenance Practices

- Narrative of program logic
- Data dictionary
- Entity relationship diagram
- Object diagrams
- Data flow diagrams

➢ User manual

- Data entry procedures
- Batch operation procedures

Information Systems Maintenance Practices

- Emergency Changes
  - They are Required to resolve system problems and enable critical processing to continue
  - It Involves the use of special logon-IDs that grant temporary access to production environment during emergency situation.
  - Emergency IDs have special privileges hence their usage should be logged & carefully monitored

Information Systems Maintenance Practices

- Testing program changes

While assessing the effectiveness of the change control procedure, the auditor should check if:

- All change requests are documented in a standard form containing
  - Adequate description of change desired
  - Explanation of reasons for change request
  - Signature of the requestor
  - Date of request
- Cost and time analysis of change has been conducted by the developer

Information Systems Maintenance Practices

- It is compliant with the approval process for change request
- Access rights and the rationale behind them have been clearly defined
- The process of checking in and communicating change to the users is being implemented

It is a good practice that the auditor takes a change request and trace all activities from the documentation to assure himself that all the processes are being followed

Information Systems Maintenance Practices

- Library Control Software
  - It Ensure that all program changes have been authorized
  - The primary purpose of Library Control software is to:
    - Prohibit programmers from accessing production source & object libraries.
    - Prohibit batch program updating
    - Require the control group or operator to release source code & place it in the programmer's library
    - Require the programmer to turn over changed source code to control group who updates the object library
    - Allow read-only access to source code
    - Require that program naming conventions have a unique identifier to distinguish test from production versions

Information Systems Maintenance Practices

- Executable and Source Code Integrity
  - Each production executable module has to have one corresponding source module
  - When a modified program is moved into the production source code library, a compiled executable version of that program should be moved to the production library
  - Ensures that the improper version of the program is not executed
  - Timestamp of the source's module should not be later than its corresponding executable
  - User or application programmer should have no access to production source code or libraries

Information Systems Maintenance Practices

- Program code comparison
  - Auditors use program code comparison to ensure that the source code provided to them for review is the same as the one that has been compiled to produce the current version of object code
  - Program code comparison can be of two types:
    - Source code comparison
    - Object code comparison

Information Systems Maintenance Practices

- Source Code Comparison
  - The source code comparison is an effective and easy-to-use method for tracing changes to programs
  - Allows IS auditor to examine source-program changes and controls without assistance or misleading information from IS personnel
  - Does not detect a source program changed and restored to its original form
  - IS auditor obtains a source listing of the module being audited and scans the program for changes

Information Systems Maintenance Practices

- Object code comparison

  Object code comparison is more difficult as object code is not readable. This is done through following steps:

  - After going through the source code comparison, the auditor should compile a copy of the source code in the production directory and generate an object code
  - The test plan for the program should be applied on the object code and the results must be documented
  - The same test data has to be applied on the object code available in the production directory
  - The results of both these tests should match

Information Systems Maintenance Practices

- Configuration Management
  - Configuration management involves various procedures to
    - identify
    - define and
    - baseline

    software items in the system thus providing a basis for
    - problem management,
    - Change management and
    - release management
  - Configuration management process involves identification of items like programs, documentation and data.

Information Systems Maintenance Practices

- Configuration Management
  - Configuration management involves various procedures to
    - identify
    - define and
    - baseline

    software items in the system thus providing a basis for
    - problem management,
    - Change management and
    - release management
  - Configuration management process involves identification of items like programs, documentation and data.

Information Systems Maintenance Practices

- Configuration management team can be formed to whom the programs, documentation and data as stated above is handed over for safekeeping.
- Each is assigned a reference number for a quick retrieval.
- Once handed over to the configuration management team, the item cannot be changed without a formal change control process which is approved by a change control group.
- The process of moving an item to the controlled environment is called checking in.
- When a change is required, the item will be checked out by the configuration manager.
- Once the change is made, it is checked in by a different version number.

Project Management Tools and Techniques

- Introduction
  - Activities in a project need to be
    - Planed
    - controlled and
    - monitored
  - There are many tools available for a Project Manager to do that
  - The most appropriate tool or technique should be chosen, depending on the size and complexity of the project

Project Management Tools and Techniques

These tools provide assistence in the following areas:

- Program Planning , Scheduling and performance Measurement:
  - They are effective in planning a project within stipulated time frame and also take care of the budget
- Budget and Schedules
- Software Cost Estimation
  - The project is divided into smaller phases and cost associated with each part is found
- Software Configuration Management
  - The information sytem used in the project can be configured easily using these automated tools

Project Management Tools and Techniques

- Documentation
  - The input of the user is used by the automated tools to generate program narrtives and Flowcharts.
  - They are also used for handling of production , validation and maintainance of sytem and program documentation.
- Project Management
  - These tools help in effective project managemnt.
  - They incorporate PERT techniques.
- Office Automation
  - Tools such as E-Mail system, Voice messagig, E-Calenders, E-reminders, automated libraries and retrieval system

Project Management Tools and Techniques

- Phases in Project Management
  - Planning
    - Setting the objectives
    - Control to be executed
    - Calculating the duration
    - Establishing the relationship between activities
  - Scheduling
    - Calculate the start & finish time of each activity
    - Calculate the critical path
    - Calculate the slack and float for path other than the critical path
  - Controlling
    - Preparation of the progression of the project
    - Reviewing the project
    - Determining the status of the project
    - Analysing the management decisions

Project Management Tools and Techniques

- Function Point Analysis
  - The function point analysis (FPA) technique is widely used for estimating complexity in developing large business applications.

Project Management Tools and Techniques

FPA Feature Points

- In most standard applications, lists of functions are identified and the corresponding effort is estimated
- E.g. in web-enabled applications, the development effort depends on
  - number of screens (forms)
  - number of images
  - type of images (static or animated)
  - features to be enabled
  - interfaces and
  - cross referencing that is required.
- This estimate is arrived at in terms of person-months required to develop the application

Project Management Tools and Techniques

- Now this gross person-month effort has to consider details, such as:
  - In which sequence will these activities be performed?
  - How will the total person-month effort be distributed over these activities, i.e. how much time will each activity take and how many persons of which type of skills will be required for each of these activities?
  - On which date will each activity start and finish?
  - What additional resources are required to complete the activity?
  - What will be the measure that assesses the completion of an activity?
  - What will be the points in which the management will review the project? (milestones.)

Project Management Tools and Techniques

- Software Cost Estimation

  To use the automated techniques of cost estimation, a system is usually divided into main components, and a set of cost drivers is established. Components include:

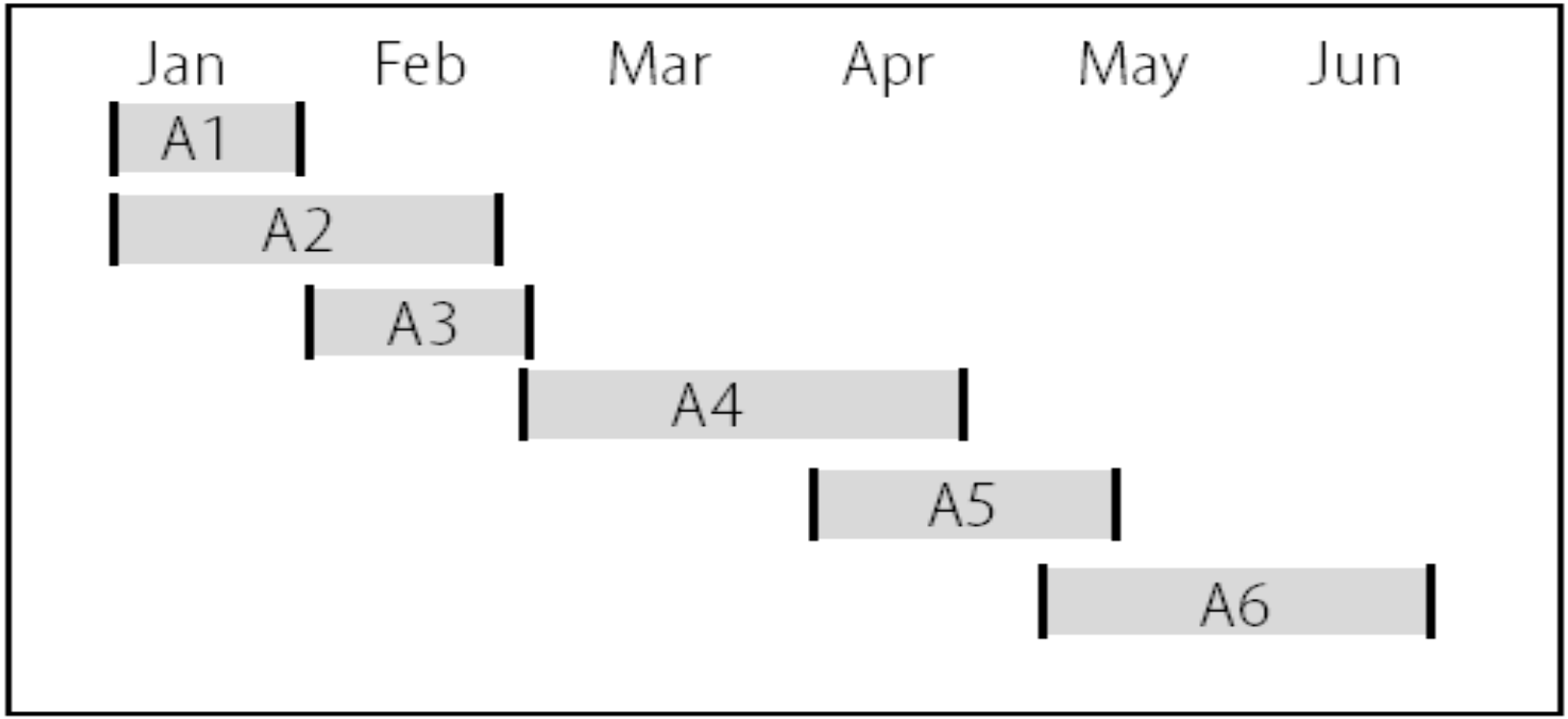
  - Source code language
  - Execution time constraints
  - Main storage constraints
  - Data storage constraints
  - Computer access
  - The target machine used for development

Project Management Tools and Techniques

- Gantt Charts
  - Gantt Charts are prepared to schedule the tasks involved in the software development
  - They show when the tasks should begin and end
  - what tasks can be undertaken concurrently, and what tasks must proceed serially
  - They help to identify the consequences of early and late completion of the tasks

Project Management Tools and Techniques

Gantt Chart



Project Management Tools and Techniques

- Program Evaluation Review Technique (PERT)
  - PERT indicates the sequential and parallel relationship between activities
  - PERT terminology
    - Activity: An activity is a portion of the project, which requires resources and time to complete. The activity is represented by an arrow in the following diagram.
    - Event: An event is the starting or end point of an activity. It does not consume resources or time. It is represented by the starting circle in the following diagram

Project Management Tools and Techniques

- Predecessor activity: Activities that must be completed before another activity can begin are called predecessor activities for that activity. In the following figure, Activity A is the Predecessor Activity for Activity E
- Successor activity: Activities that are carried out after an activity is completed are known as successor activities. In the following figure, Activity E is the successor activity of Activity A while Activity G is the successor activity for Activities F, C and D

Project Management Tools and Techniques

- Slack: Slack is the difference between earliest and latest completion time of an activity
- Dummy: Dummy activity is that activity which requires no resources
- Time estimate: PERT recognizes that the estimates cannot be precise, and hence allows a weighted average of different estimates such as pessimistic, optimistic and most likely. A heavier weightage is given to the most likely estimate and the calculation is as follows:
  - to - optimistic estimate
  - tp - pessimistic estimate
  - tm - most likely estimate
  - Expected time = (to+4tm+tp) /6

Project Management Tools and Techniques

PERT

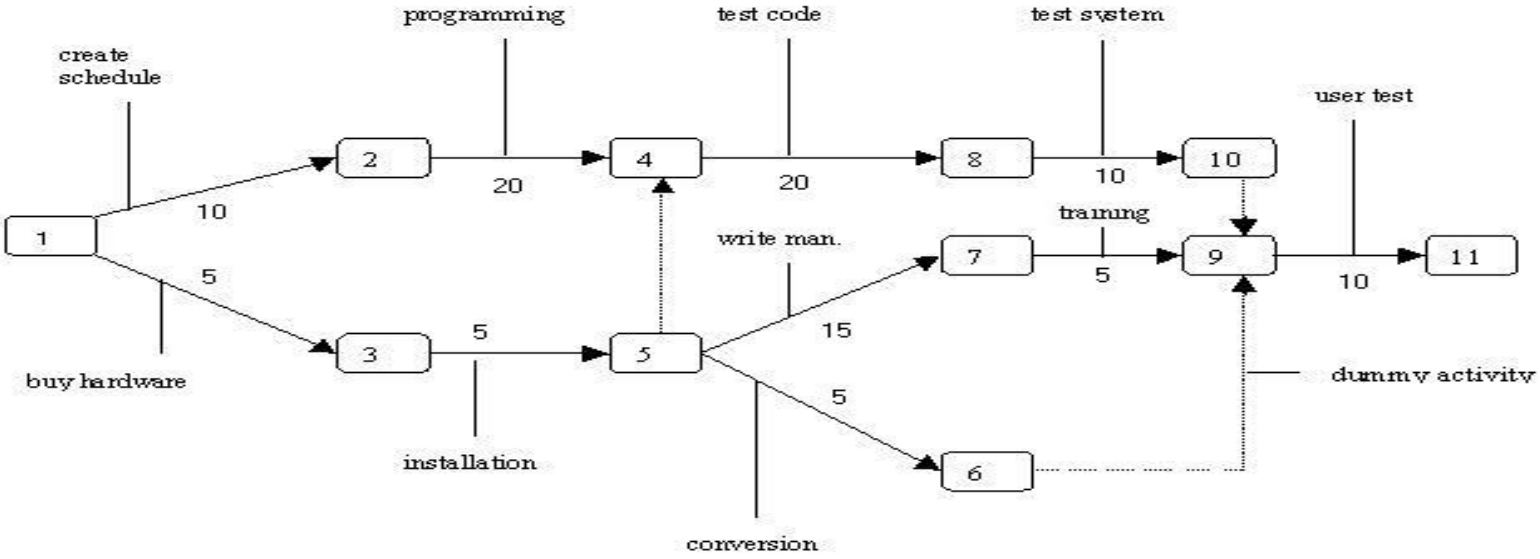
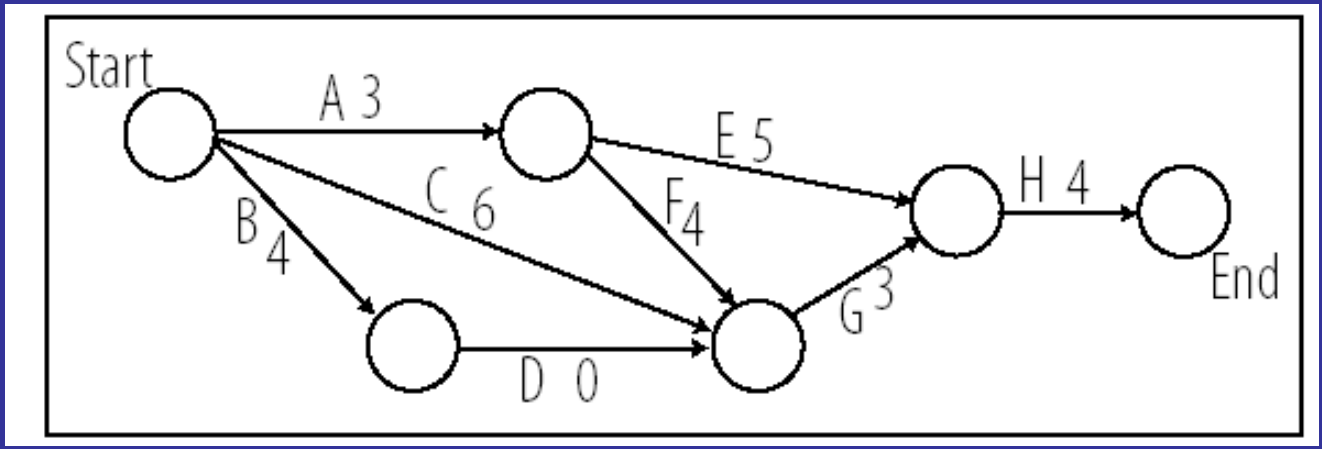


Fig. 1:
PERT Chart

* Numbered rectangles are nodes and represent events or milestones.
* Directional arrows represent dependent tasks that must be completed sequentially.
* Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
* Dotted lines indicate dependent tasks that do not require resources.

Project Management Tools and Techniques

- Critical Path Method (CPM)
  - Path 1: A - E - H = 12 days
  - Path 2: A – F – G – H = 14 days (Critical Path)
  - Path 3: C – G – H = 13 days
  - Path 4: B – D – G – H = 11 days



System Development Tools and Productivity Aids

- Code generators
  - Tools that generate program code on the basis of
    - Parameters defined by system analyst
    - Data flow diagrams created in design phase
  - Work as aid in increasing programmer's efficiency

System Development Tools and Productivity Aids

- Computer Aided Software Engineering (CASE)
  - CASE is an attempt to automate all activities associated with the software development life cycle.
  - Some of the CASE tools are:
    - Project planning tools
      - Estimation tools
      - Scheduling tools

System Development Tools and Productivity Aids

- Risk analysis tools
- Requirement tracing tools
- Analysis and design tools
- Process modelling tools
- Web application development tools
- Quality assurance tools
- Testing tools
- Dynamic analysis tools
- Documentation tools
- Metrics tools
- Configuration management tools
- Reengineering tools

System Development Tools and Productivity Aids

- Classification of CASE tools
  - Upper CASE: These tools are useful in the early stages of system life cycle. Tools that help in defining application requirements fall in this
  - Middle CASE: These tools address the needs in the middle levels of SDLC such as Design. Those that help in designing screen and report layouts, data and process design falls in this category.
  - Lower CASE: The later parts of the life cycle make use of these tools. These tools use design information to generate program codes.

System Development Tools and Productivity Aids

- Benefits of using CASE
  - Since CASE strictly follows SDLC, use of CASE enforces the discipline of following steps of SDLC
  - The standardization / uniformity of processes can be achieved
  - Since CASE tools generate inputs of each stage from the outputs of previous stage, consistency of application quality can be ensured
  - Tasks such as diagramming, which are monotonous, need not be done by the programmer, and can be left to the CASE tool
- Disadvantages of CASE
  - CASE tools are costly, particularly ones that address the early stages of the life cycle
  - Use of CASE tools requires extensive training

System Development Tools and Productivity Aids

- Auditing CASE environment
  - Since CASE introduces many disciplines and enforces standards and consistency, many of the tasks of auditor might become easier. However, the auditor should address the following issues:
    - In case different tools are used at different stages, then the interface between these tools should be checked to make transition easy and ensure consistency
    - The integrity of data between manual process and CASE tool and vice-versa should also be checked
    - The appropriateness of the tool to the application development methodology should also be checked

System Development Tools and Productivity Aids

- The auditor should check the manner in which changes are introduced at different stages of the development process. They must be the consistent throughout the stages
- CASE does not automatically guarantee application controls. The auditor should check this aspect, as he would do in a traditional application development environment
- The CASE data base security and access controls should also be checked

Specialized Systems

- Artificial Intelligence (AI)
  - Artificial Intelligence is an attempt to duplicate intelligent behaviour of the human beings in computer system on the basis of predetermined set of rules
  - A few such capabilities include:
    - Thinking and reasoning
    - Using reason to solve problems
    - Learning from experience
    - Exhibiting creativity and imagination
    - Handling ambiguous or incomplete information

Specialized Systems

- The applications of AI can be classified into three major categories:
  - Cognitive Science
  - Robotics and
  - Natural Languages

Specialized Systems

- Cognitive Science: This is an area based on research in disciplines such as biology, neurology, psychology, mathematics and allied disciplines. It focuses on how human brain works and how humans think and learn. Applications of AI in the cognitive science are:
  - Expert Systems: These are information systems with reasoning capability. The components of expert systems are:
    - User interface
    - Interface engine
    - Knowledge base

Specialized Systems

User Interface Programs

Inference Engine Program

Knowledge Base

User **Workstation**

THE EXPERT SYSTEM

Knowledge Acquisition Programs

Workstation EXPERT

EXPERT SYSTEM DEVELOPEMENT

Specialized Systems

- Advantages of expert systems:
  - The knowledge and experience of the expert is captured before he leaves the organisation
  - The codified knowledge in a central repository makes it easy to share it with the less experienced in the application area
  - This ensures consistent and quality decisions
  - It also enhances personnel productivity
- Limitations of expert systems:
  - Expert systems perform well in solving specific types of problems in a limited domain. When the problems involve multiple domains, expert systems become difficult to construct

Specialized Systems

- » They do not have the capacity to learn and from that point of view are static in their knowledge
- » Usage of specialized languages render maintenance of expert systems difficult
- » Development costs of expert systems are high

– Applications of expert systems:

- » Portfolio analysis
- » Insurance
- » Demographic forecasts
- » Help Desk operations
- » Medical diagnostics
- » Maintenance scheduling
- » Communication network planning
- » Material selection

Specialized Systems

- Applications of expert systems in IS Audit
  - » Risk Analysis: This expert system will evaluate materiality and various types of risks associated with the auditee.
  - » Evaluation of Internal Control: This expert system can identify the likely exposures in the given control area.
  - » Audit Program planning: This expert system will recommend a set of audit procedures to be conducted on the basis of auditee's characteristics and may help to evaluate the internal control system of the organization
  - » Technical Advice: This expert system assists in technical advice which are encountered during audit. E.g. Confirming that the financial statements confirm the statutory regulation.

Specialized Systems

- Learning Systems: These are the systems that can modify their behaviour based on information they acquire as they operate. Chess playing systems is one such popular application
- Fuzzy logic: These are systems that can process data that are ambiguous and incomplete. This permits them to solve unstructured problems. Many embedded systems such as in auto-focus cameras and energy efficient air conditioners use fuzzy logic.
- Neural networks: These are computing systems modeled after the human brain. It permits them to recognize patterns and refine them with more and more data input. For example credit risk determination could be a good application for neural network systems.

Specialized Systems

- Intelligent agents: Most of the modern word processing packages have a facility to observe user operations, correct his mistakes and provide useful hints depending on the context. This facility is an intelligent agent.

- Robotics: This technology produces robot machines with computer intelligence and human-like physical capabilities. Robotics find expensive application in computer aided manufacturing.

Specialized Systems

- Natural languages: Being able to ‘converse’ with computers in human languages is the goal of research in this area. Interactive voice response and natural programming languages, closer to human conversation, are some of the applications.

Specialized Systems

- Data Warehouse
  - A planned, integrated, managed store of relevant corporate data optimized for analysis, query and reporting purposes
  - The data is extracted from the operational database. A data warehouse contains historical data of a much longer period than operational database
  - This is the collection of enterprise data extracted from the business operations, often combined with derived or third party information

Specialized Systems

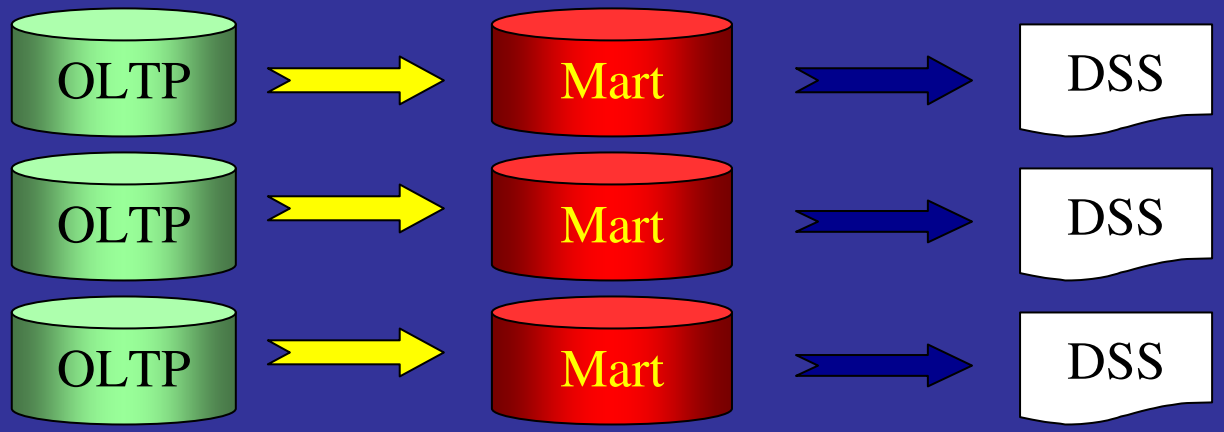
- The structure of the data warehouse could be:
  - Subject Oriented Data Warehouse
    - e.g. sales, marketing etc
  - Integrated Data Warehouse
    - Complete data for speedy access
  - Time Variant Data Warehouse
    - Sorted on time basis for comparison purposes

Specialized Systems

- The Data Warehouse is primarily composed of:
  - Data sources made up of existing databases, files and external information
  - Data extraction and transformation from data sources (i.e. transactions to the data warehouse) where the information is extracted, cleansed and restructured for inclusion in the data warehouse
  - Data mart, which is a data intended for a single function or a Department on selected subjects like marketing, product etc, is created by various user departments through the process of aggregation and summarization.

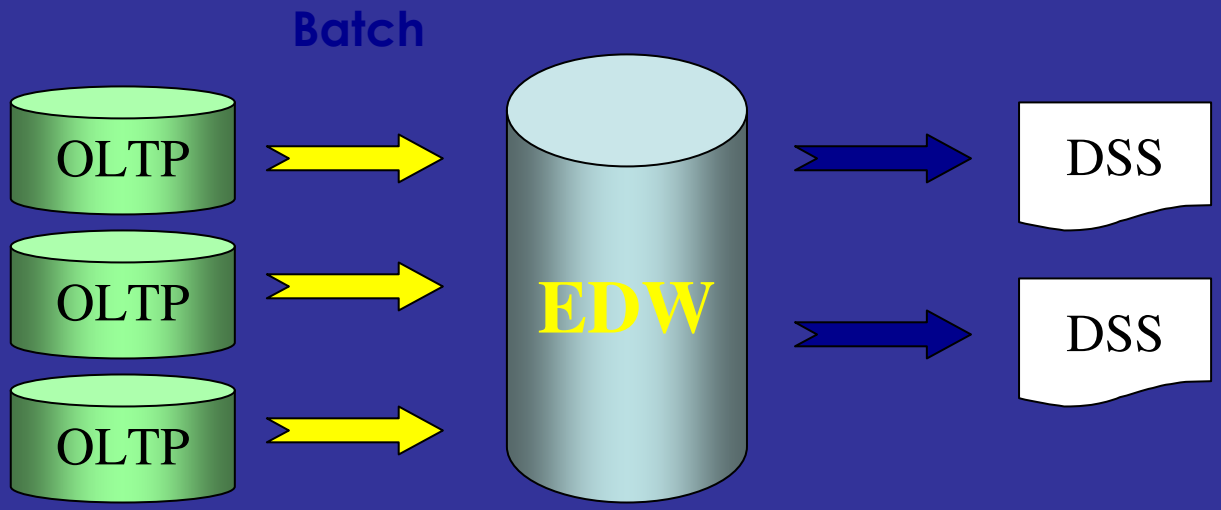
Specialized Systems

Independent Data Marts - Architecture

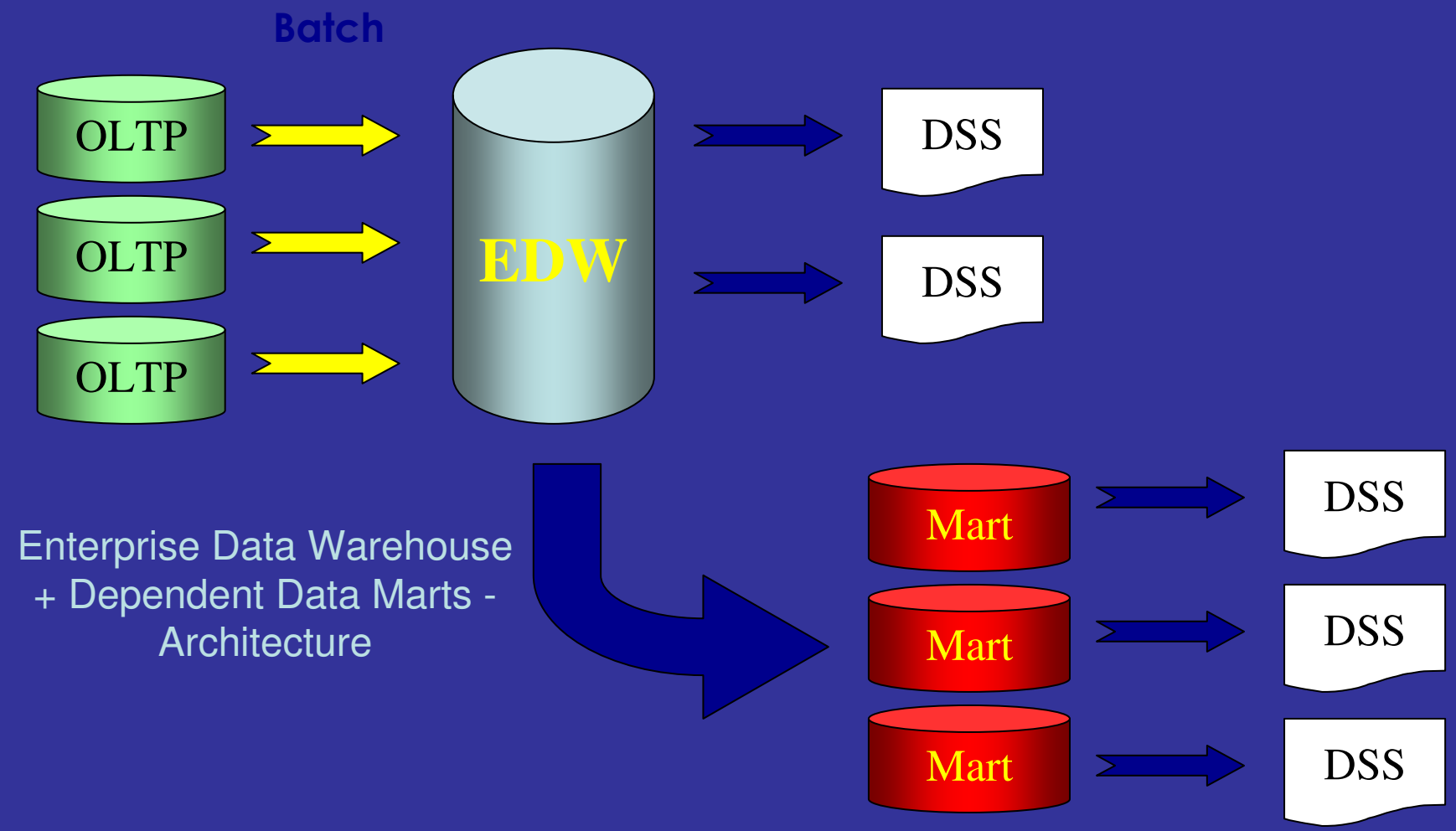


Specialized Systems

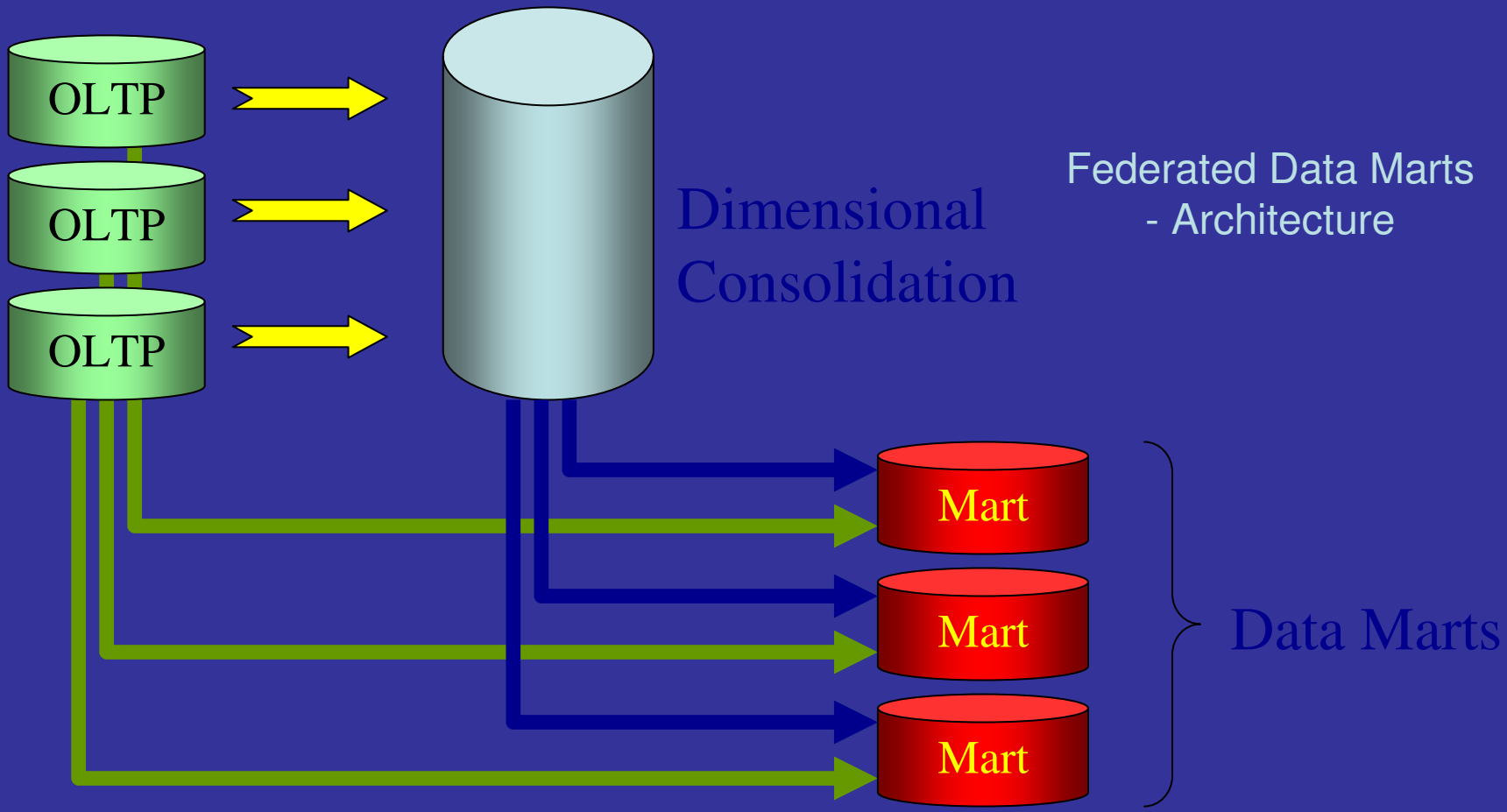
Enterprise Data Warehouse - Architecture



Specialized Systems



Specialized Systems



Specialized Systems

- Developers during development of the technology architecture to consider
  - Data delivery and access issues
  - Accessibility to source data
  - Complexity of source data structure
  - Distributed vs. Centralized
  - Network capacity
  - Excellent performance

Specialized Systems

- Data warehouses find application in areas such as market analysis
- The process of identifying patterns in data warehouse is called data mining
- When a data warehouse is not created for the entire enterprise, but only for a part of it for a specific function, it is called data mart

Specialized Systems

- Decision Support Systems (DSS)
  - Information systems that provide interactive information support to managers with the use of analytical models
  - DSS are designed to be adhoc systems, modeled for specific decisions of individual managers
  - Typical examples:
    - Comparative sales figures between two consecutive months for different products with the percentage variation to total sales
    - Revenue and Cost projections on the basis of certain product mix
    - Evaluation of different alternatives, leading to selection of the best

Specialized Systems

- DSS frameworks
  - DSS framework will depend upon
    - The extent to which the problem can be structured
    - The level of management that will use the model. This corresponds to the type of decision making involved –
      - operational
      - control or
      - strategic

Specialized Systems

- Decision Support Systems

Management Level Dimension	Decision structure
1) Operational Control	1) Structured
2) Management Control	2) Semi structured
3) Strategic Planning	3) Unstructured

While a structured problem is amenable to automation, a highly unstructured problem involves greater complexity in unearthing the requirements.

Specialized Systems

- DSS - Design and Development
  - Prototyping is the most popular approach to DSS design and development
  - The prototype is iteratively refined on user feedback
  - Therefore, this can be a cost effective and speedy method

Specialized Systems

- DSS - Assessment and evaluation
  - The true test of a DSS lies in whether it improves the manager's decision making, which is difficult to measure in tangible terms
  - DSSs is evolutionary in nature and they do not have a defined completion date
  - Therefore, DSS should be evaluated in the incremental steps for tangible results at each step

Specialized Systems

- Point of Sale Systems (POS)
  - A POS system is intended to capture data at the time and place of transaction
  - It is often attached to scanners to read bar codes and magnetic cards for credit card payment and electronic sales
  - They provide significant cost and time saving as compared to the manual methods
  - They also eliminate errors that are inherent in manual system
  - POS may involve batch processing or an online processing

Specialized Systems

- Automated Teller Machines (ATM)
  - An automated teller machine is a specialized form of the point of sale terminal
  - Designed for unattended use by a customer of a financial institution
  - ATMs transfer the financial information over communication lines
  - These systems must provide a high level of logical and physical security for both the customer and the ATM machine

Specialized Systems

- Electronic Data Interchange (EDI) Systems
  - Electronic Data Interchange is the oldest form of transmitting business transactions between the business partners with dissimilar computer systems
  - EDI is used to transmit and exchange business documents like
    - purchase orders,
    - request for proposals,
    - invoices and
    - shipping notices etc.

    in a standard machine readable format.

Specialized Systems

- The advantages of EDI are:
  - Reduction in paperwork
  - Improved flow of information
  - No necessity of repeated data entry
  - Less errors while transmitting / exchange of information
  - Speed in communication due to electronic transmission
  - Improvement in carrying out a business process.

Specialized Systems

- EDI system function
  - **Communication Software:**
    - moves the data from one point to another
    - marks the start and the end of the EDI transmission and
    - decides how the acknowledgements are transmitted and reconciled
  - **Translation Software:**
    - involves conversion of data from a business application translated into a standard format, to be transmitted over the communication network
    - convert this data back from the EDI format into the proprietary format of the receiver organization
  - **EDI standard:**
    - specifies the standards for the transmittal of the business documents like invoices, purchase orders etc.

Specialized Systems

Traditional EDI process generally involves three functions within each trading partner's computer system

- **Communication handler:**
  - Process for transmitting and receiving electronic documents between trading partners
  - VAN uses computerized message switching and storage capabilities
  - VAN may perform translation and verification services.

Specialized Systems

- **EDI Interface:**
  - Interface function manipulates and routes the data between the application system and the communications handler
  - May generate and send the functional acknowledgements
  - Verify the identity of the partners and check the validity of the transactions by checking the transmission information against the trading partner master file

Specialized Systems

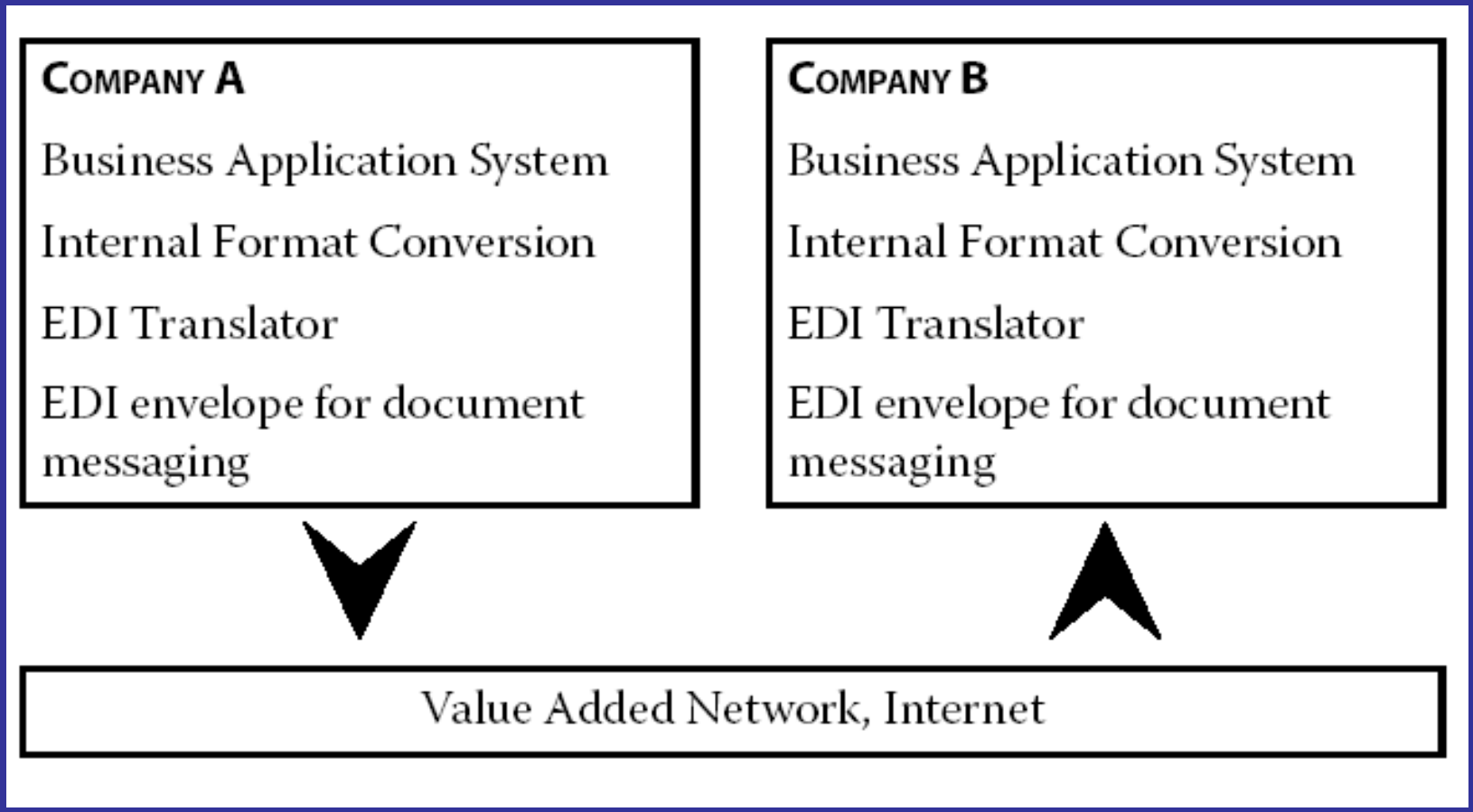
- The interface consists of two components.
  - EDI Translator – This device translates the data between the standard format and the trading partner's proprietary format
  - Applications Interface – This interface moves the electronic transactions to or from the applications systems and performs data mapping

➢ **Application System:**

- The programs that process the data sent to/ received from the trading partner. E.g. Purchase orders from purchasing system.

Specialized Systems

- EDI System



Specialized Systems

The following are the Layers for EDI transmission:

EDI Layers	Constituents	
EDI Semantic Layer	Application level services	
EDI Standard Layer	EDIFACT business form standards	
	ANSI X.12 business form standards	
EDI Transport Layer	Electronic Mail	X.435 MIME
	Point to Point	FTP, TELNET
	World Wide Web	HTTP
Physical Layer	Dial Up lines, Internet	

Specialized Systems

EDI Risks and Controls:

Risk of EDI	EDI Controls
Improper transaction authorization	Controls for inbound / outbound transactions
Integrity of EDI transactions	Encryption
Loss or duplication of EDI transactions	Use of sequence number for each transaction
Loss of confidentiality	Encryption
Non Repudiation	Digital signature

Specialized Systems

- Electronic Commerce (E Commerce)
  - Other than buying and selling goods on the Internet, the E Commerce involves:
    - Information Sharing
    - Payment
    - Fulfillment
    - Service and Support

Specialized Systems

- The Advantages of the E Commerce are:
  - Savings in Cost
  - Saving in transaction time
  - No limitations of the geographical boundaries
  - Larger availability of the customer base for the suppliers and larger choice to the customers
  - No restriction of timings
  - Storage or holding cost can be greatly reduced
  - Different roles for the intermediaries

Specialized Systems

- The E Commerce risks are those risks connected with the transfer of any information with Internet like:
  - Authentication
  - Confidentiality
  - Integrity
  - Non-repudiation
  - Availability

Specialized Systems

- Types of E Commerce Models
  - Business to Business (B to B) relationship
  - Business to consumer (B to C) relationship
  - Business to Employee (B to E) relationship
  - Business to Government (B to G) relationship
  - Consumers to Consumers (C to C) relationship
  - Citizen to Government (C to G) relationship
  - Exchange to Exchange (X to X) relationship

Specialized Systems

- Enterprise Resource Planning Systems (ERP Systems)
  - Enterprise Resource Planning (ERP) are fully integrated corporate solutions focusing on the business applications like
    - Finance
    - Production
    - Planning
    - Sales
    - Warehousing
    - logistics etc.

Specialized Systems

- Some ERP Software
  - SAP
  - Oracle Applications
  - BAAN
  - JD Edwards
  - PeopleSoft etc.

Questions

Which of the following types of testing would determine whether program changes have introduced errors that did not exist before

- Interface Testing
- Unit Testing
- System Testing
- Regressive Testing

Questions

Which of the following would MOST likely ensure that a system development project meets business objectives

- Maintenance of Program Change Logs
- Development of a Project Plan identifying all development activities
- Release of application changes at specific time of the year
- User Involvement and System Specifications and Acceptance

Questions

Which of the following procedures would an IS Auditor not perform during the design phase of a system project

- Assist in developing a functional design for embedded audit routines
- Assess the adequacy of the system
- Advise the analyst regarding control routine
- Review the design for adherence to corporate policies

Questions

With respect to various phases in the SDLC which of the following is least likely to vary ?

- Conduct of each phase
- Sequence in which phases are performed
- Presence of each phase
- Resources needed to perform each phase

Questions

Which of the following is unlikely to be a concern of designers during the design of the data / information flow for the information processing system ?

- The extent to which the data and information flows should be formalized
- The frequency and timing of information flows
- The medium that will be used to transport the data and information flows
- The locations where transformation are to be performed on data

Questions

In which of the following phases of the systems development process is the Auditor most likely to have greatest involvement ?

- Conversion
- Organizational and Job Design
- Elicitation of strategic requirements
- Information Processing System Design

Questions

Which of the following is NOT considered an advantage of package software ?

1. Reduced development cost
2. Reduce Risk of logical error
3. Increased Processing efficiencies
4. Increased Flexibility due to Optional features

Questions

During a detailed system design, the IS Auditor would be LEAST concerned with

1. Adequacy of audit trails
2. Handling of rejected transactions
3. Adequacy of hardware to handle the system
4. Procedures to ensure that all transactions are received

Questions

Which of the following statements regarding the function of a System Development Life Cycle Steering Committee is FALSE ?

1. Review projects progress regularly
2. Report only to senior management on project status
3. Serve as a Coordinator and Advisor to answer questions about system and program design
4. Take corrective actions regarding personal changes on the project team

Questions

Risk analysis is MOST useful when applied during which phase of the system development process?

A. Project initiation
B. System Construction
C. Acceptance Testing
D. Implementation Planning

Questions

In regard to moving an application program from the test environment to the production environment, the BEST control would be provided by having the :

1. Application Programmer copy the source program and compiled object modules to the production libraries
2. Application Programmer copy of the source program to the production libraries and then have the Production Control Group compile the program
3. Production Control Group copy the source program and compile the object module to the production libraries
4. Production Control Group copy the source program to the production libraries and then compile the program

Questions

The PRIMARY purpose of undertaking a parallel run of a new system is to:

A. verify that the system provides required business functionality.
B. validate the operation of the new system against its predecessor.
C. resolve any errors in the program and file interfaces.
D. verify that the system can process the production load.

Questions

Coding standards would provide which of the following?

A. Field naming conventions
B. Data flow diagrams
C. Access control tables
D. Program documentation

Questions

Which of the following would MOST likely ensure that a system development project meets business objectives?

A. Maintenance of program change logs
B. Development of a project plan identifying all development activities
C. Release of application changes at specific times of the year
D. User involvement in system specification and acceptance

Questions

Many IT projects experience problems because the development time and/or resource requirements are underestimated. Which of the following techniques would provide the GREATEST assistance in developing an estimate of project duration?

A. Function point analysis
B. PERT chart
C. Rapid application development
D. Object-oriented system development

Questions

The MOST appropriate person to chair the steering committee for a system development project with significant impact on a business area would be the:

A. business analyst
B. chief information officer
C. project manager
D. executive level manager

Questions

A project manager has asked that you advise him of the potential risk associated with the use of timebox development techniques in a system development project. Which of the following would NOT be good advice?

A. That the timebox technique should only be applied to projects that can be completed within a reasonable timeframe.

B. For the timebox approach to be effective, end-users and management should have agreed to core functionality to be developed in the timebox.

C. That delivery of all functionality within the timebox is more important than quality.

D. That the timebox approach will require the use of evolutionary prototype techniques

Questions

When auditing the requirements phase of a software acquisition, the IS Auditor should:

A. assess the feasibility of the project timetable
B. assess the vendor's proposed quality processes
C. ensure that the best software package is acquired
D. review the completeness of the specifications