Information Systems Audit
Module VII

Mathew A Thomas
Grad. CWA, FCA, MCA, CDM, ISA, CISSP.

Introduction

- ✓ About the Course
- ✓ About the Module
  - ► Module 7 (22%) 44 Questions
- ✓ About the Material
  - ► Source, Organisation
  - ► Module 7: Chapter 28-33

About this Presentation

- ✓ Chapter 28:
  - ► Business Application Development F& P 4
- ✓ Chapter 29:
  - ► System Development Lifecycle (SDLC) 15
- ✓ Chapter 30:
  - ► Alternative Development Methodologies 60
- ✓ Chapter 31:
  - ► Project Management Tools & Techniques 84
- ✓ Chapter 32:
  - ► Specialized Systems 97
- ✓ Chapter 33:
  - ► IS Auditor's Role 121

Business Application Development Framework & Prerequisites

Chapter 28

Topics

- ✓ Need for Structured Methodology
- ✓ Phases of Structured methodology
- ✓ Roles and Responsibilities
  - People involved in SDLC

Need for Structured Methodology

- 1980's Software crisis
  - Realization that substantial portion of the application development cost is going to be in software.
- Problems:
  - Application development was far from methodical
  - No scientific method for estimating the time
  - Software projects also suffered heavy cost overrun
  - No guarantee that the software would perform the required functionality and there were many instances of software 'bombing' on site.
  - What cannot be measured cannot be managed.

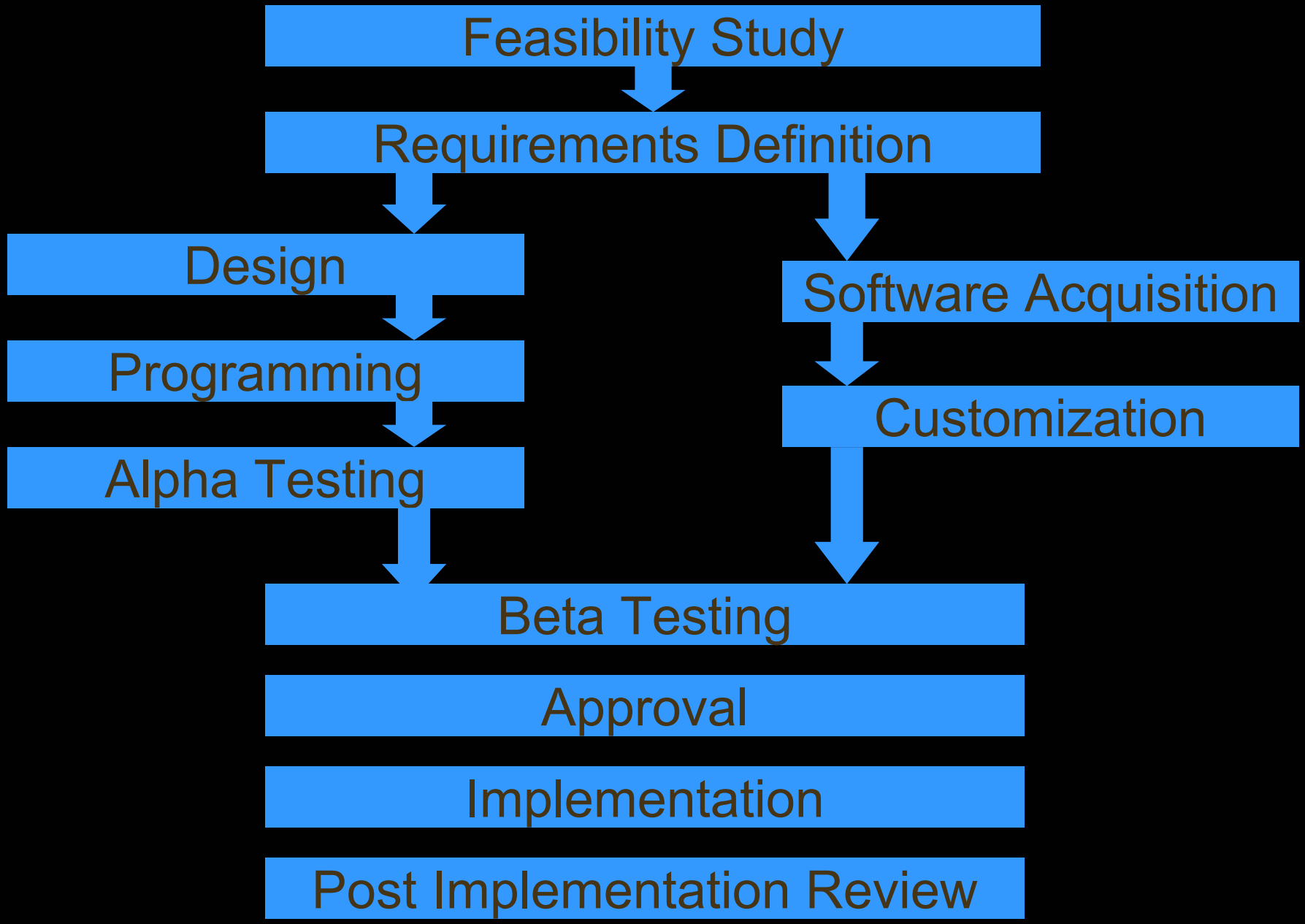
Estimation, Measurement and Management

- Software is a logical product and not a physical product
  - A product being intangible makes all the difference in estimating and measuring and hence in managing
- Software product depends on the quality of people employed.
- Software product undergoes continuous - sometimes drastic - changes after delivery.
- Every application is built as a new application and no code reuse.

Structured Methodology

- Steps
  - Problem Analysis
  - Design of a solution
  - Construction of a solution
  - Verification of the appropriateness of the solution
  - Ensuring sustainability of the solution - Maintenance
  - Management of technical aspects
- Answers the following questions
  - What is the problem to be solved?
  - What is the solution proposed?
  - Which characteristic of the solution will address which portion of the problem?
  - Now that the solution has been outlines, how can it be realized?
  - What approach will be used to uncover errors in design and construction?
  - How will the product be supported over a long time?

Structured Methodology



Roles and Responsibilities I

- ✓ Senior Management
  - ► Commits to the project
  - ► Approves the necessary resources to complete the project.
- ✓ User Management
  - ► Assumes ownership of the project and resulting system,
  - ► review and approve system deliverables
- ✓ Project Steering Committee
  - ► Provides overall direction
  - ► Responsible for all costs and timetables
- ✓ Project Sponsor
  - ► Provides funding for the project.
  - ► Typically the senior manger in charge of primary business function the application will support.

Roles and Responsibilities II

- ✓ Systems Development Management
  - ► Provides technical support for hardware/software/tools
  - ► Operating support and maintenance after installation.
- ✓ Project Manger
  - ► Provides overall direction,
  - ► Ensures project adheres to local standards
  - ► Resolves inter-departmental conflicts
  - ► Monitors and controls costs and project timetable.
- ✓ Systems Development Project Team
  - ► Does assigned tasks
  - ► communicates with users

Roles and Responsibilities III

- ✓ User Project Team
  - ► Communicates with the systems developers
- ✓ Security Officer
  - ► Ensures that system design adheres to security policies
  - ► Tests system security prior to implementation,
- ✓ Quality Assurance
  - ► Review results and deliverables within each phase
  - ► Confirm compliance with requirements

Roles and Responsibilities IV

- ✓ Systems Analyst
  - ► Also called as Business Analyst.
  - ► Conduct interviews with users and elicit user requirements.
- ✓ Testers
- ✓ Domain Specialist
- ✓ Technology Specialist
- ✓ IS Auditor

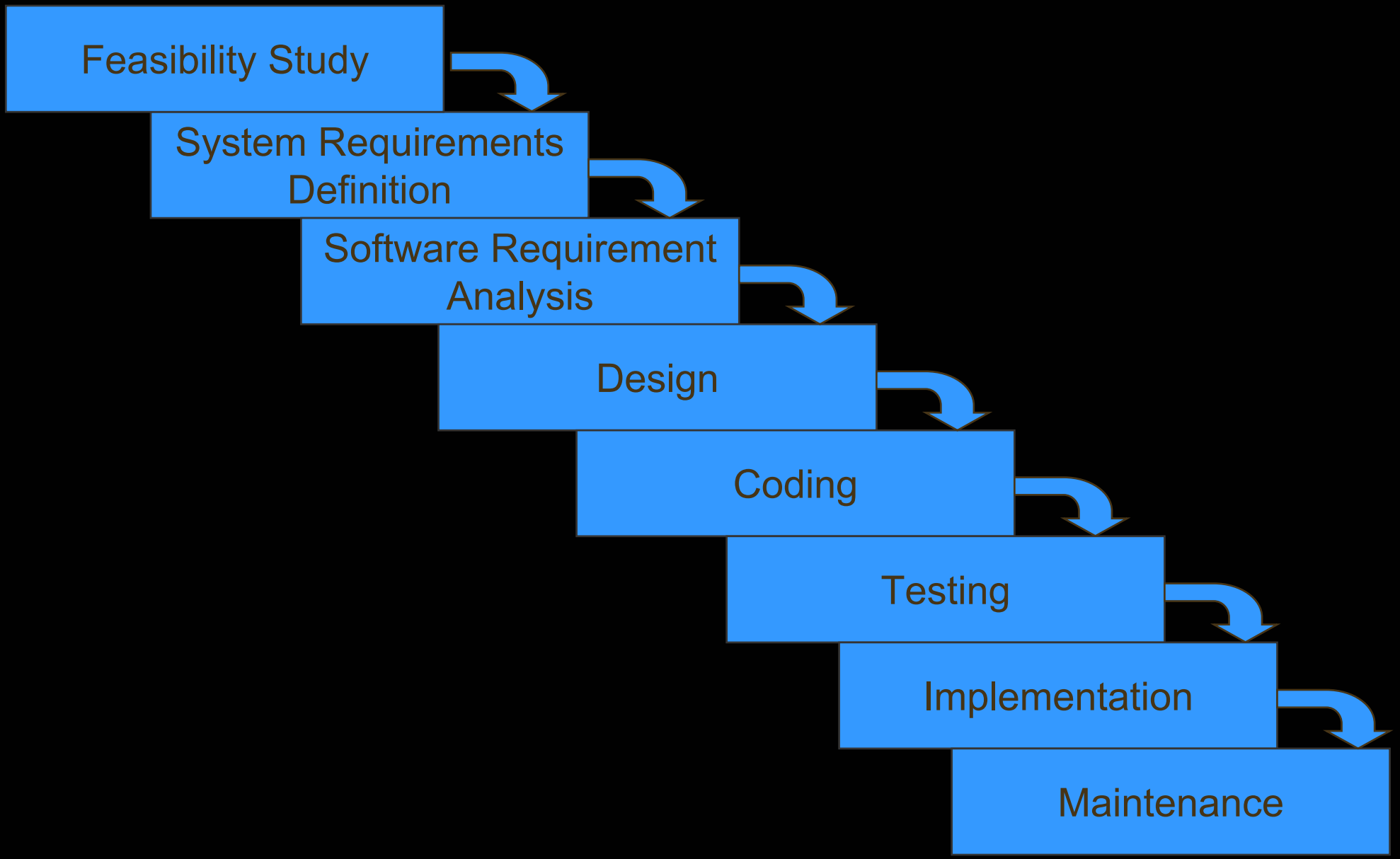
Risk Associated with Poor SDLC

- System does not meet the users' business needs, user requirements and expectations
- Requirements are still unfulfilled
- System is under-utilized and not maintained
- Exceed the limits of the financial resources
- The IS Auditor should
  - be aware that merely following an adequately designed SDLC methodology does not ensure the successful completion of a development project.
  - review other aspects of systems development, such as user involvement and senior management support, in addition to compliance with the organization's SDLC.

SDLC

Chapter 29

Traditional SDLC (Waterfall Model)



Traditional SDLC

- ✓ Feasibility Study
- ✓ System Requirement Analysis
- ✓ Design
- ✓ Program Construction
- ✓ Testing
- ✓ Implementation
- ✓ Maintenance

Feasibility Study

- ✓ Identification of problem
- ✓ Identification of objective
- ✓ Delineation of scope
  - ► Functionality Requirements
  - ► Control Requirements
  - ► Performance Requirements
  - ► Reliability Requirements
  - ► Constraints
  - ► Data and Interfaces etc.
- ✓ Feasibility Study
  - ► Technical Feasibility
  - ► Economic/Financial Feasibility
  - ► Behavioral, Legal, etc. Feasibility

System Requirement Analysis

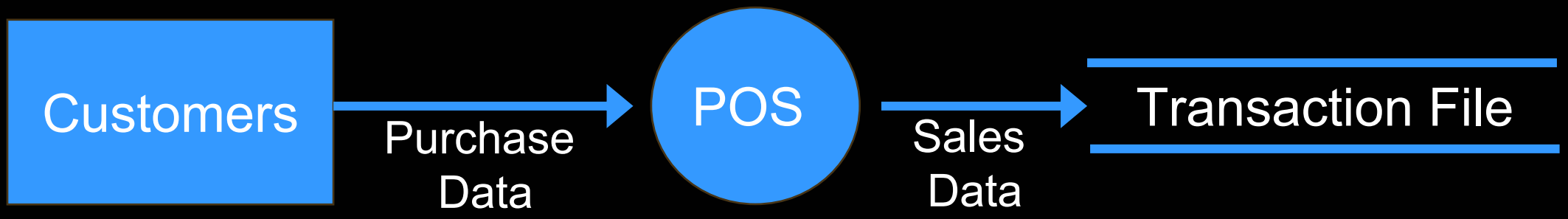
- ✓ Study the existing organizational history, structure and culture
- ✓ Study the existing information flows
  - ► Data Flow Diagram
  - ► Flow Charts
  - ► Decision Trees
  - ► Decision Tables
  - ► Structured English
- Object-oriented Analysis

Data Flow Diagrams

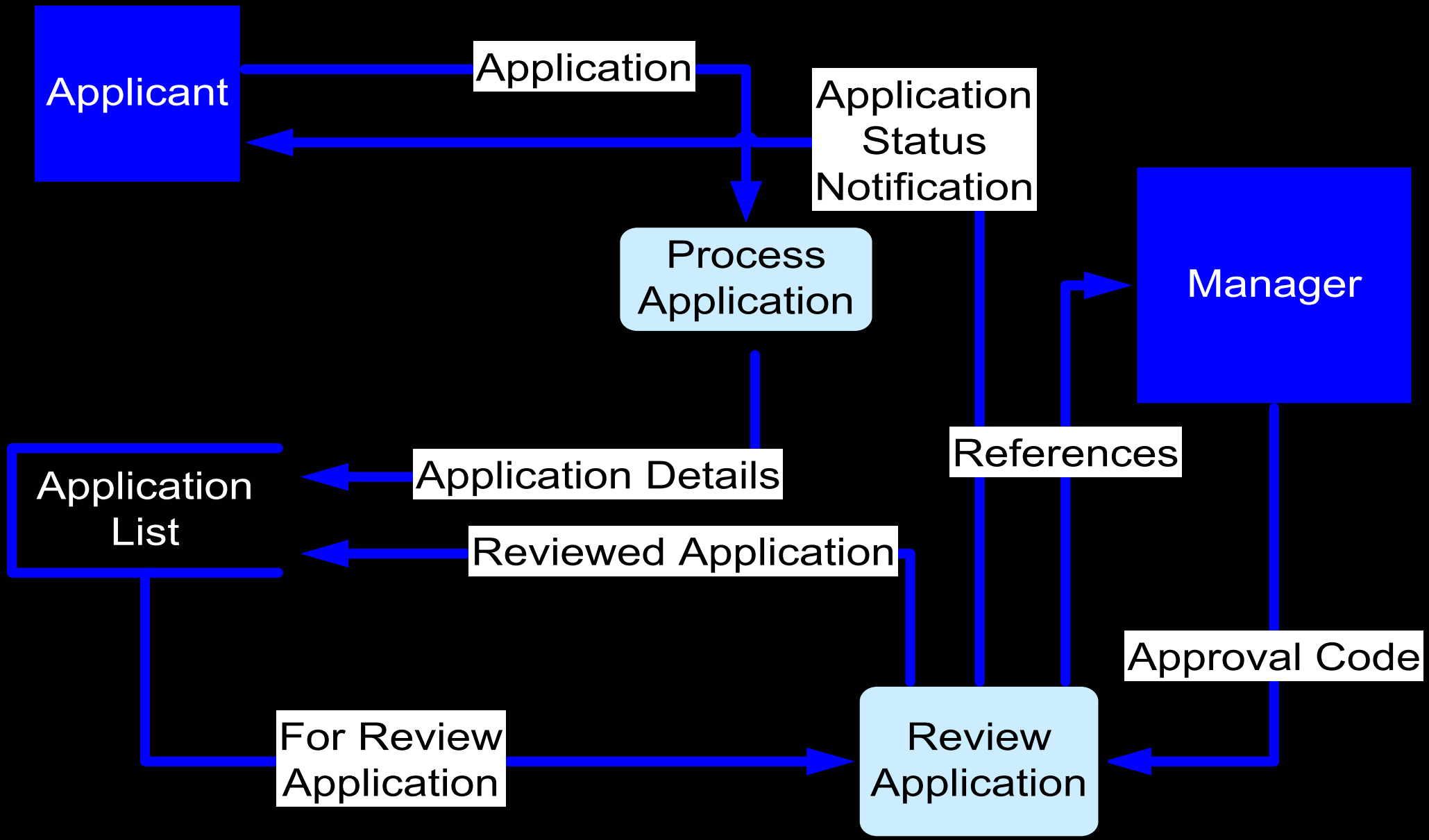
- Data Flow Diagram has four elements:

Element	*Represented By*
Data Flow	Arrow
Process	Circle (bubble), or Rounded edge rectangle
Data Stores	Parallel lines, or Open rectangle
External Entity	Square

Data Flow Diagram



Data Flow Diagram



Data Flow Diagram

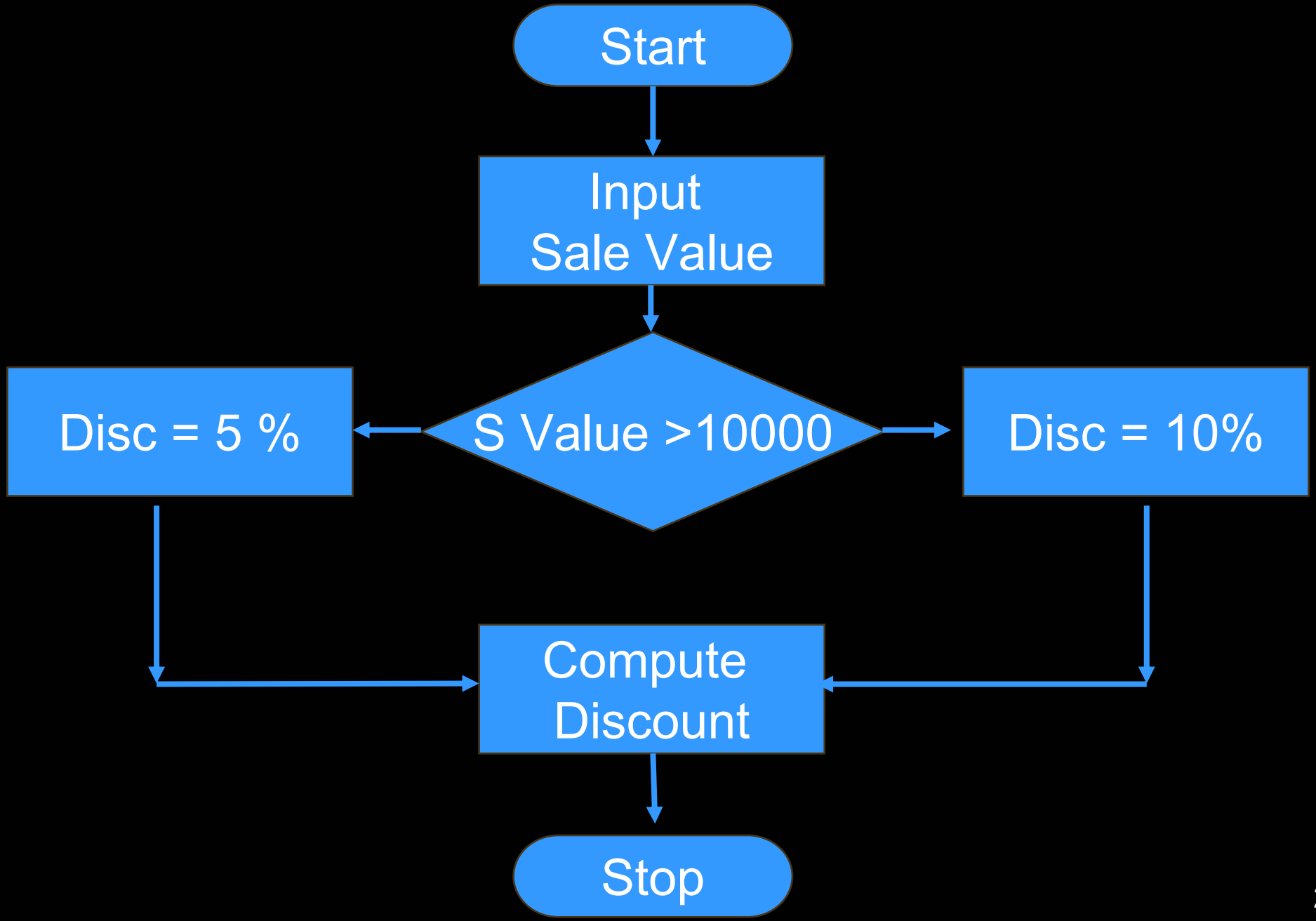
- Rules of DFD
  - All entities, processes and data stores should be numbered and named.
  - It is preferable to mention the information flowing on the arrows too
  - DFD is always read from top to bottom and left to right
  - No more than seven processes (five is ideal) should be represented in the same page.
- Limitations of DFD
  - It is not possible to represent conditions and decisions (as in flow charts) in DFD
  - DFDs do not show the time intervals or delays between processes.

Flow Charts

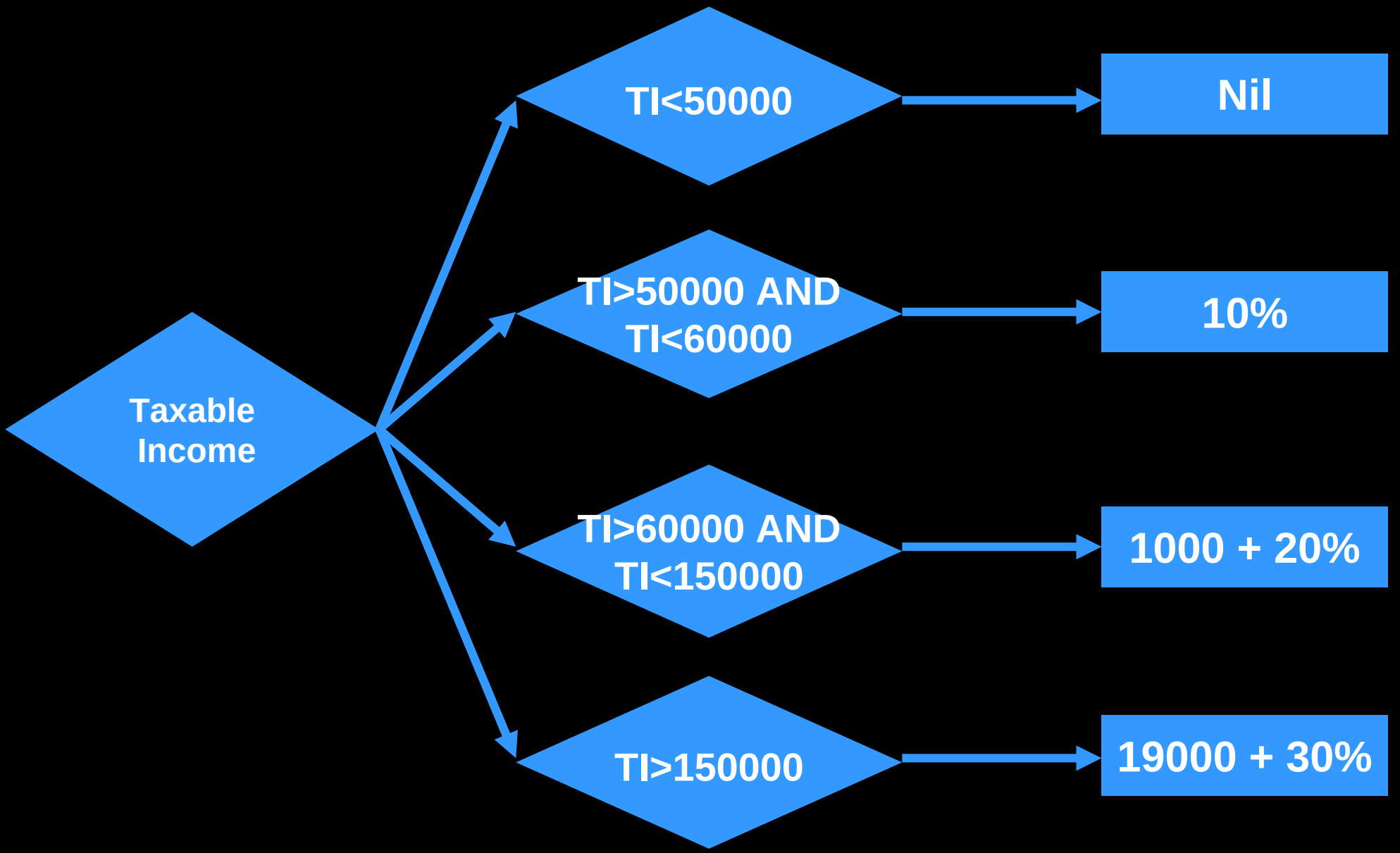
- In a flow chart:

Element	*Represented By*
Start and stop	Circle
Processing step	Rectangle
Condition/Decision	Diamond/Rhombus
Flow	Arrow

Flow Charts


Start
Input
Sale Value
S Value >10000
Disc = 5 %
Disc = 10%
Compute
Discount
Stop

Decision Trees



Decision Tables

	1	2	3	4
Domestic Consumer	T	T	F	F
Industrial Consumer	F	F	T	T
Consumption < 100 KWH	T	F	T	F
Consumption >= 100 KWH	F	T	F	T
Billing Rate Rs 1 per unit	X			
Billing Rate Rs 2 per unit		X	X	
Billing Rate Rs 3 per unit				X

Structured English

- ✓ IF (Income>60000 AND Income=<150000)
  THEN Tax =1000 + (Income-60000)* 0.1

Systems Design

- Steps
  - Architectural Design
  - Design of Data / Information Flow
  - Design of Data Base
  - Design of User Interface
  - Physical Design
  - Selection of Hardware and Software

Architectural Design

- Identification of
  - 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

Architectural Design

- Functional Decomposition is a tool to represent hierarchies
- Three issues
  - Modularity
  - Cohesion
  - Coupling

Modularity

- Module
  - Module is a manageable unit performing a well defined task.
  - Modules contain data and instructions for a task.
  - The modules interacting with a module need not know the process within a module and need to know only the interfaces.
  - Such a modular design results in good maintainability and this is why modular design is recommended.

Cohesion and Coupling

- Cohesion
  - Cohesion relates to how well the elements within a module fit together.
  - Performs one discrete task
  - Strength of bonding between the elements of a module
- Coupling
  - Coupling is a measure of strength of interconnection between modules.
  - Number and complexity of connections between the modules.
  - In a good modular design, cohesion will be high and coupling low.

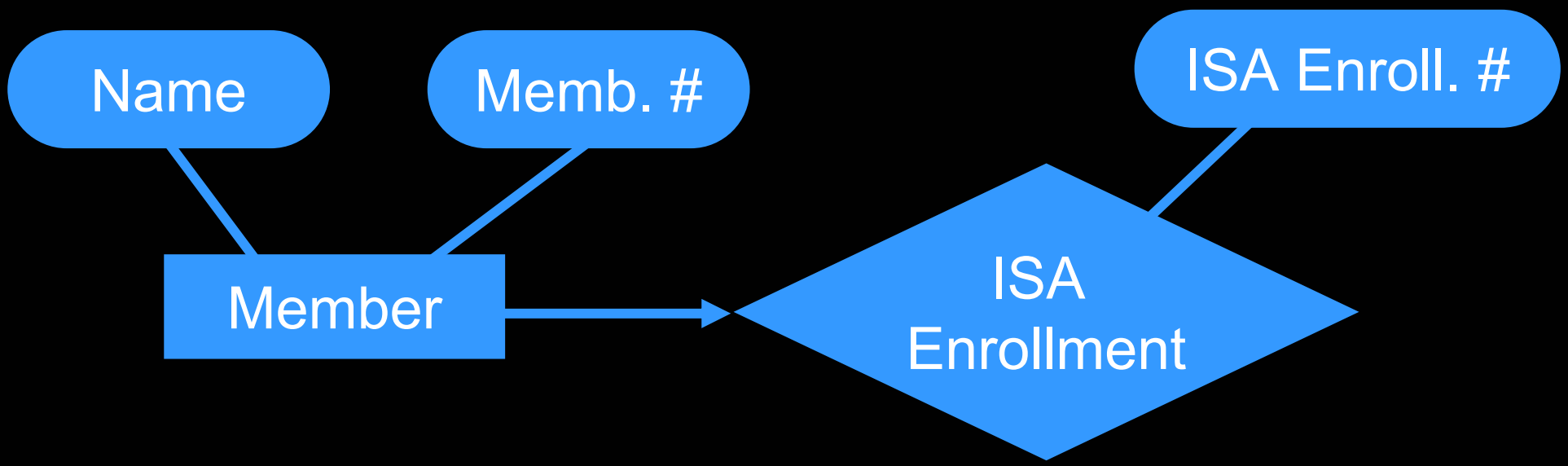
Design of Data / Information Flow

- Inputs
  - Existing data / information flows,
  - Problems with the present system
  - Objective of the new system
  - Frequency and timings of data flows

Design of Data Base

- ✓ Conceptual Modelling
  - Entities / objects and their relationships.
  - Attributes are the characteristics of an entity
  - E-R diagram
- ✓ Data Modelling
  - Relational
- ✓ Storage structure design
- ✓ Physical layout design

E-R Diagrams



Design of User Interface

- Considerations
  - Level of user – educational background, computer exposure
  - Source document format
  - Output requirements
  - Enquiry screens
  - Graphic and color displays
  - Requirement for special input / output device

Physical Design

- Issues:
  - Type of hardware for client and server
  - Operating systems to be used
  - Type of networking
  - Processing – Batch. On-line , real-time
  - Frequency of input, output
  - Month-end, Year-end functions

Selection of Hardware and Software

- Request for Proposal (RFP)
  - Towards purchase of new hardware and software, a request for proposal should be prepared containing
  - System requirements such as transaction volume, size of data proposed to be held on-line, response time and turn around time requirements
  - Technology specifications, if any: For example, choice of a data base software might restrict the choice of hardware and vice versa
  - Commercial terms: any special condition the organization would like to insist on
  - Vendor responsibilities: apart from supply of hardware / software, the vendor is expected to deliver other things such as training, documentation and delivery dates thereof

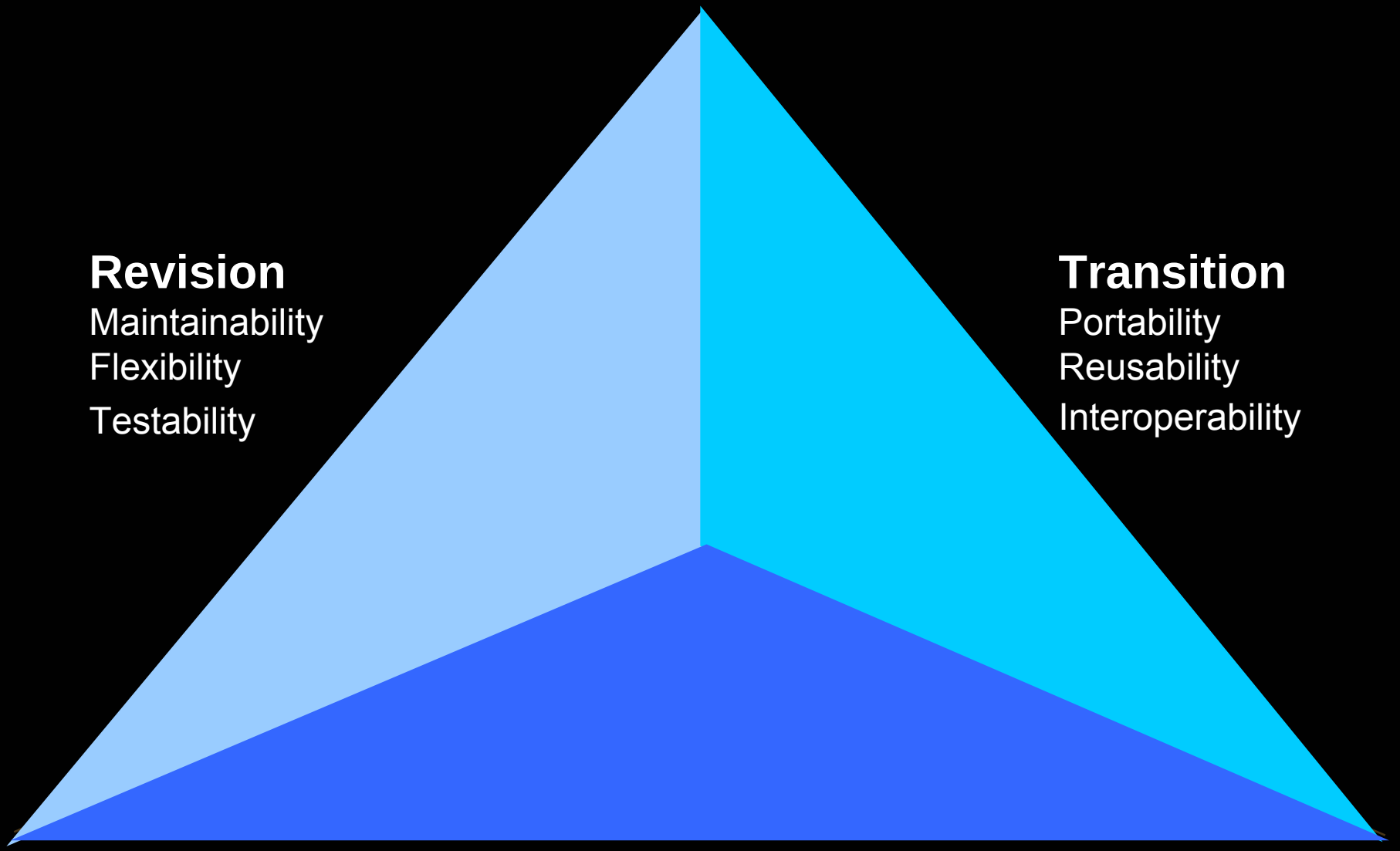
Selection of Hardware and Software

- Criteria for Vendor Selection
  - Some indicative criteria are listed here:
  - Reliability
  - years of experience with product
  - referrals from users, publications and consultants
  - Financial viability
  - Service record
  - Training capability
  - Documentation support
  - Source code availability / Escrow
  - Delivery time
  - Commercial terms

Selection of Hardware and Software

- Contents of the Contract
  - Deliverables and commitment dates
  - Cost
  - Acceptance criteria
  - Source code / Escrow
  - Commitment for fixes, upgrades, new releases
  - Support during installation

McCall's Quality Factors



Operational Characteristics

- Correctness
  - The extent to which a program satisfies its specification and fulfills the customer's mission objectives.
- Reliability
  - The extent to which a program can be expected to perform its intended function with required precision.
- Usability
  - Effort required to learn, operate, prepare input, and interpret output of a program.
- Integrity
  - Extent to which access to software or data by unauthorized persons can be controlled.
- Efficiency
  - The amount of computing resources and code required by a program to perform its function.

Ability To Undergo Change (Revision)

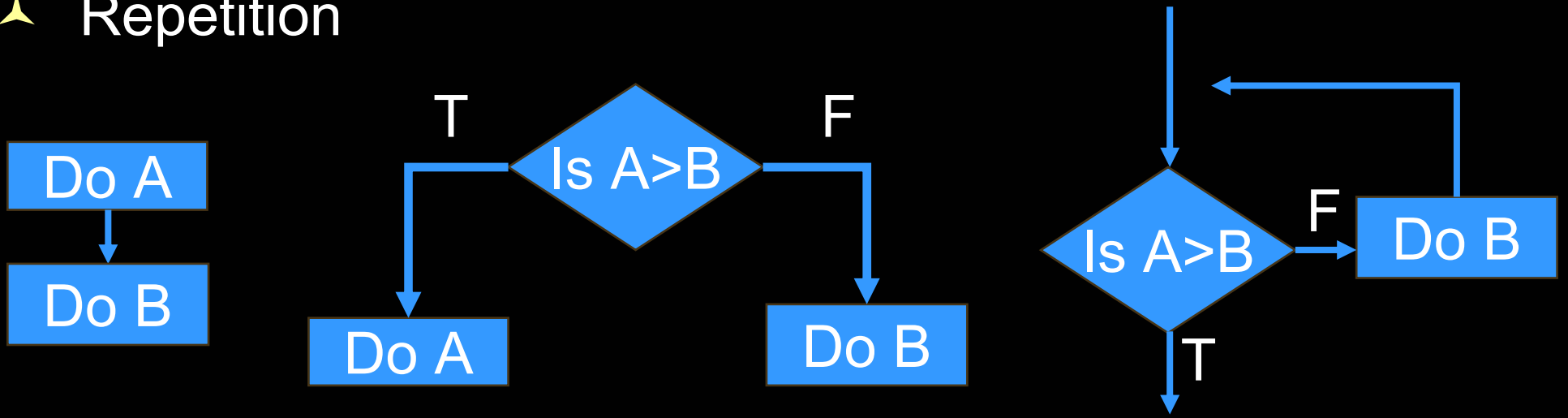
- Maintainability
  - Effort required to locate and fix an error in a program.
- Flexibility
  - Effort required to modify the program.
- Testability
  - Effort required to test a program to ensure that it performs its intended function.

Adaptability to New Environments (Transition)

- Portability
  - Effort required to transfer the program from one hardware and/or software system environment to another.
- Reusability
  - Extent to which a program can be reused in other applications.
- Interoperability
  - Effort required to couple one system to another.

Rules of Structured Programming

- ✓ Every module/construct should have only one entry and one exit.
- ✓ The program should have only three basic building block
  - Sequence
  - Condition
  - Repetition



Rules of Structured Programming

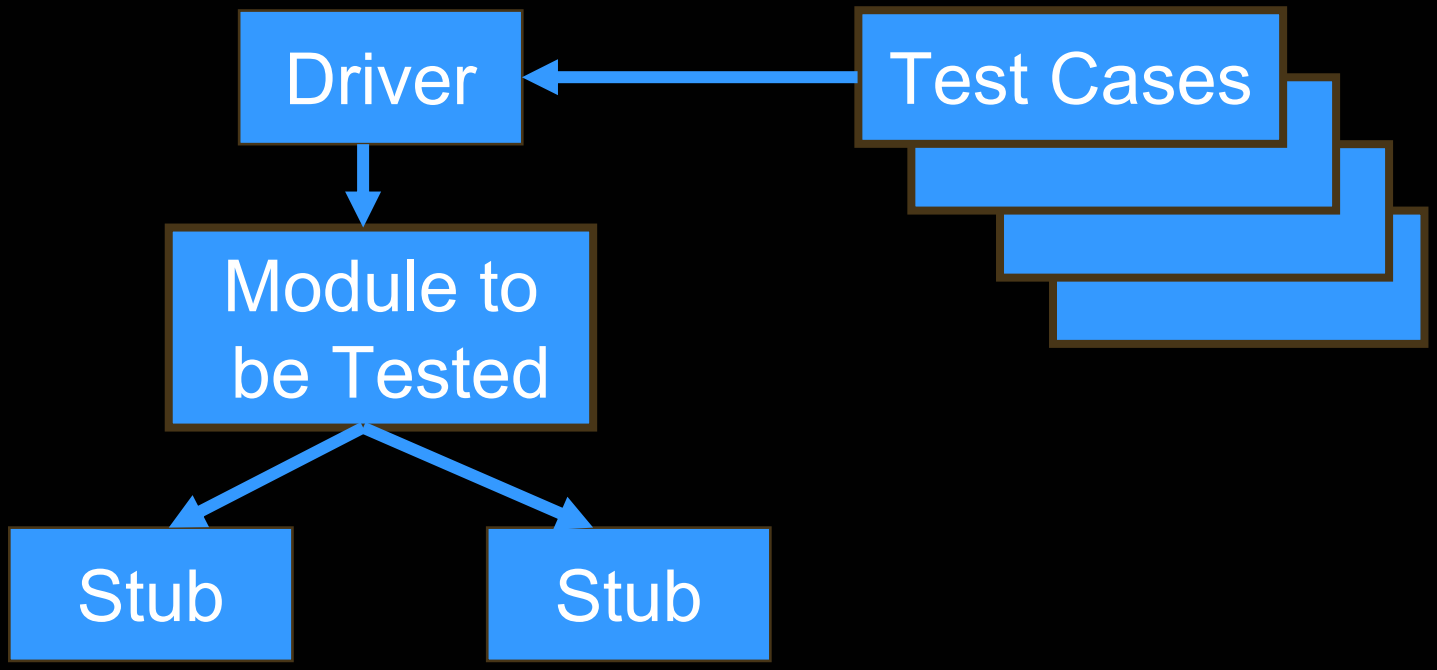
- ✓ The GOTO syntax is to be avoided in structured programming.
- ✓ Programs can be developed either top-down or bottom-up.
  - In the top-down approach, high level modules are coded and tested before lower level modules
  - In the bottom-up approach lower level modules are coded and tested before higher level modules are developed.

Software Testing- Approaches

- Bottom Up
  - Testing of atomic units, such as programs or modules and work upwards until a complete system testing has taken place
  - Advantages
    - No need for stubs or drivers
    - Errors in critical modules are found early
- Top Down
  - Follow the opposite path, either depth-first or breadth-first
  - Advantages are:
    - Tests of major functions and processing are conducted early
    - Interface errors can be detected sooner
    - Raises confidence in the system, as programmers and users actually see a working system
- Sandwich Testing
- Big Bang approach

Stubs and Drivers

- ✓ Stubs and drivers represent testing overheads. Both are software that must be developed but is not delivered with the final product
- ✓ A driver is a dummy main program and a stub is a dummy subprogram.



Software Testing- Approaches

- Black Box Testing
  - Focus on functional requirements
  - Set of input conditions that will fully exercise all functional requirements
  - Specification-based, Data driven
- White Box Testing
  - Focus is on structural correctness
  - Uses control structures of the procedural design to derive test cases.
  - Code based, Logic driven

Testing Techniques I

- Pilot Testing
  - Preliminary test that focuses on specific and predetermined aspects of a system.
  - Limited evaluation of the system.
- Unit Testing
  - Testing of an individual program or module
  - It uses a set of test cases that focus on the control structure of the procedural design
- Interface Testing
  - A test that evaluates the connection of two or more components that pass information from one area to another

Testing Techniques II

- System Testing
  - Recovery testing
    - Checking the system's ability to recover after a failure.
  - Security testing
    - making sure the system includes appropriate access controls and does not introduce any security holes that might compromise other systems
  - Stress / Volume Testing
    - Testing an application with large quantities of data in order to evaluate its performance during peak hours
  - Performance testing
    - Comparing the system's performance to other equivalent systems, using well defined benchmarks

Testing Techniques III

- Function / Validation Testing
  - Similar to system testing, but often used to test the functionality of the system against the detailed requirements
- Regression Testing
  - The process of rerunning a portion of a test scenario or test plan to ensure that changes or corrections have not introduced new errors.
- Parallel Testing
  - Feeding test data into two systems: the new/modified and an alternative/original system and comparing the results.

Testing Techniques IV

- Acceptance testing
  - Test conducted by an end user
  - Ensuring the system is ready to be implemented and satisfies all documented requirements
  - Methods include:
    - Definition of test strategies and procedures
    - Design of test cases and scenarios
    - Execution of the tests
    - Utilization of the results to verify system readiness

Debugging Tools

- Memory Dumps
  - These provide a picture of the contents of internal memory at any point of time (usually abort time)
- Trace
  - Trace lists the changes in selected variables at different stages of the program
- Logic Path Monitors
  - They provide sequence of events taken by a program
- Output Analyzers
  - These help check program results for accuracy by comparing the actual results with expected results

Implementation

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

Implementation

- Activities during Implementation Stage
  - Installation of new hardware / software
  - Data conversion
  - User training
    - Manager's training on overview and MIS
    - Operational user training – data entry, outputs etc,
    - EDP training on the technical aspects
  - Test run

Post Implementation Review & Maintenance

- ✓ Post implementation Review
  - ► System has fulfilled the objectives originally envisaged?
- ✓ Maintenance
  - Corrective Maintenance
    - ► Correcting the errors that may arise during application run
  - Adaptive Maintenance
    - ► Modification to adapt to a the new (external) environment
  - Perfective Maintenance
    - ► User wants additional functionalities
  - Preventive Maintenance
    - ► Software is changed to suit future maintainability

Umbrella Activities

- ✓ Project Tracking and Control
- ✓ Quality Assurance
- ✓ Configuration Management
- ✓ Documentation
- ✓ Change Management
- ✓ Risk Management

Alternative Development Methodologies

Chapter 30

Topics

- ✓ Data -oriented Systems Development
- ✓ Object-oriented Systems Development
- ✓ Prototyping
- ✓ Rapid Application Development (RAD)
- ✓ Reengineering & Reverse Engineering
- ✓ Structured Analysis
- ✓ Information Systems Maintenance Practices

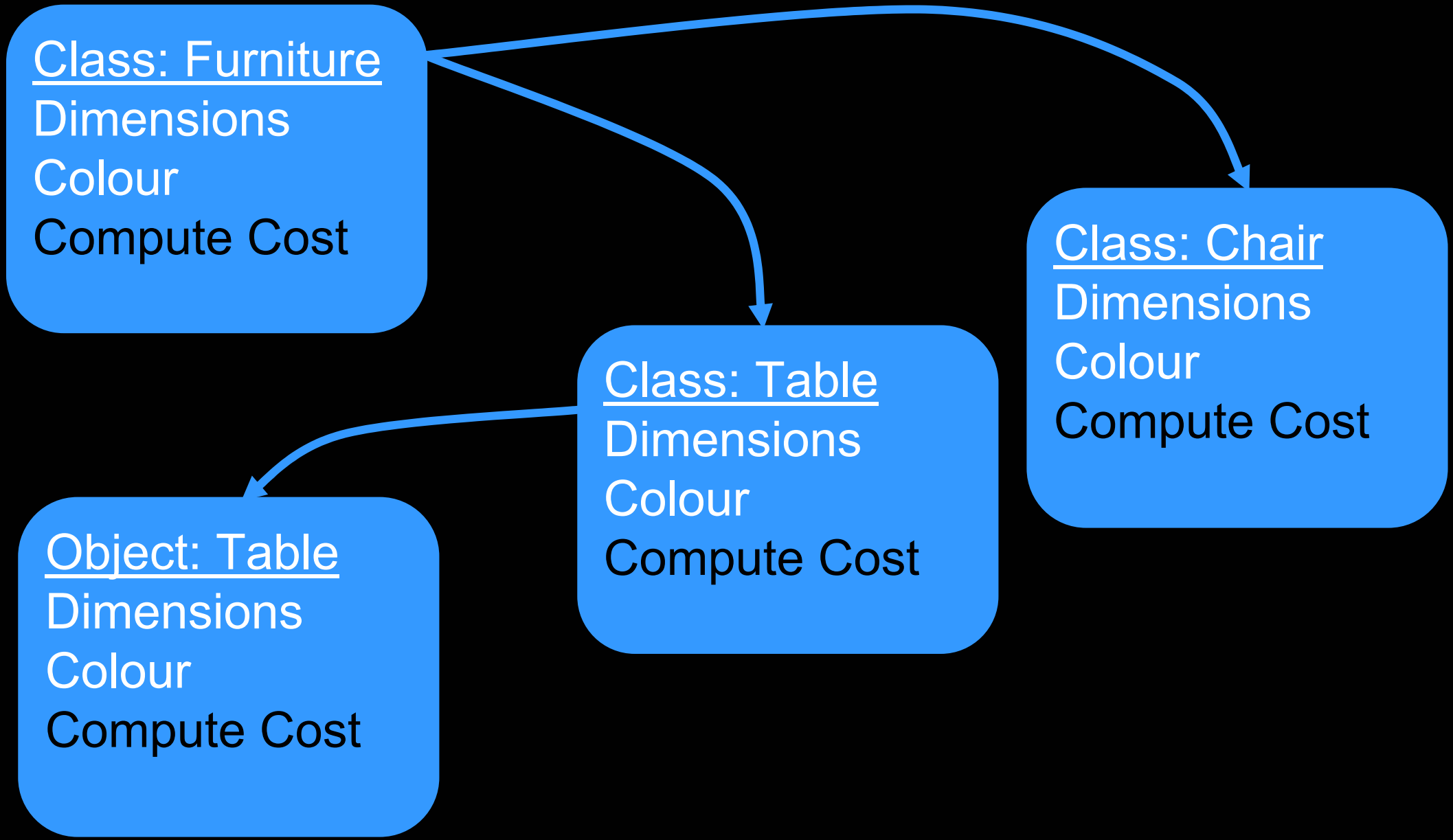
Data Oriented Systems Development

- Traditional approach
  - Transaction processing
  - Management Information Systems (MIS)
  - Data Warehousing

Object-oriented Systems Development

- Key Concepts
  - Encapsulation (Iinformation hiding)
    - Packaging a set of attributes and behaviors into an object.
    - Preventing access to non-essential things
  - Polymorphism
    - Different objects responding differently to the same message
  - Inheritance
    - Sharing of properties that characterize a class by all of its subclasses
  - Classes
    - Grouping of similar objects
  - Inheritance
    - Sharing of properties that characterize a class by all of its subclasses

Object-oriented Systems Development



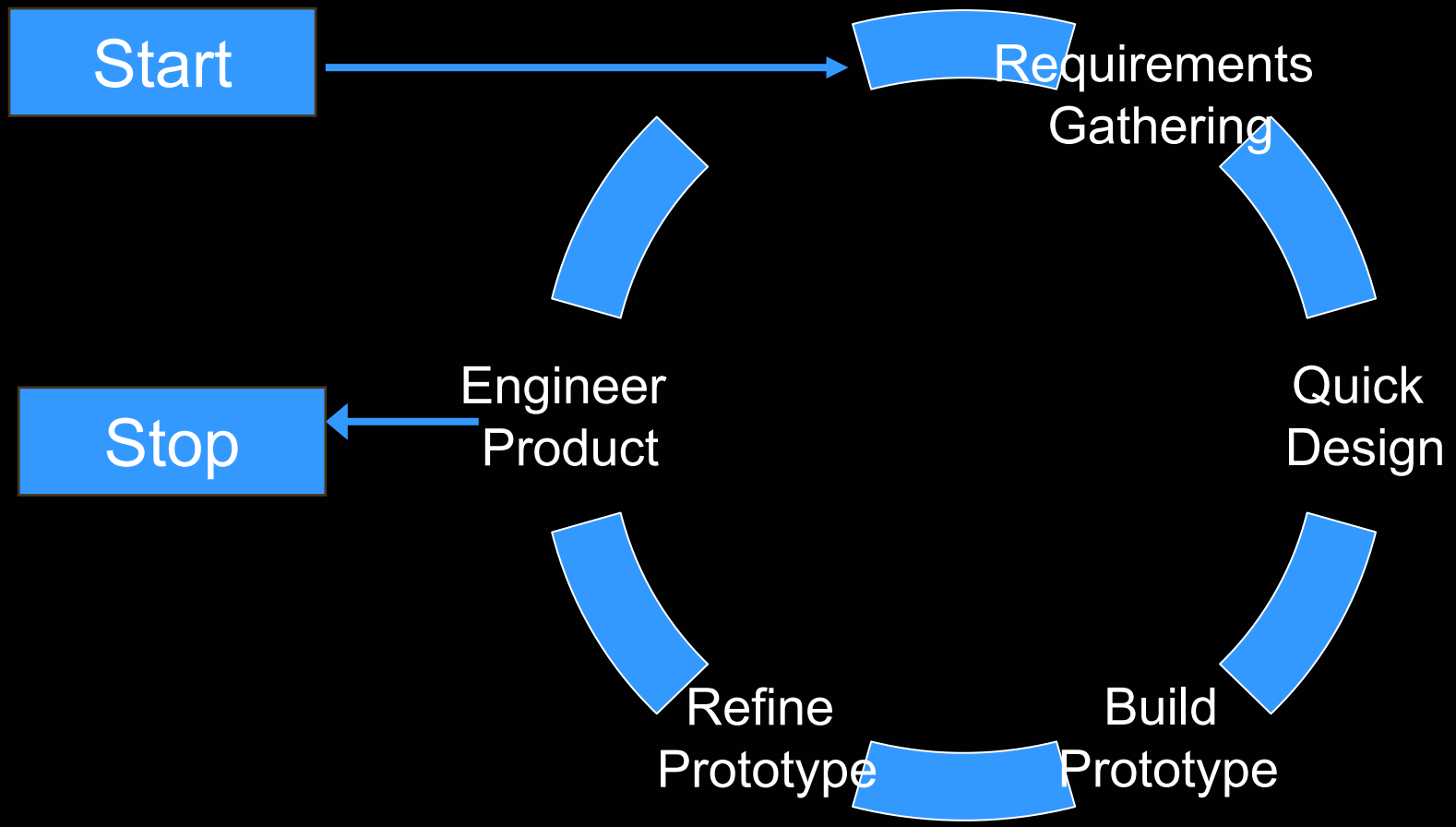
Object-oriented Systems Development

- Advantages
  - Ability to manage an unrestricted variety of data types
  - Provides a means to model complex relationships
  - Capacity to meet the demands of a changing environment
  - Object-oriented models can be manipulated easily by the user
  - Increased efficiency in programming through the ability to re-use elements of logic
  - Ability to allow a user or an application to access only the needed information, since it stores objects independently of one another

Prototyping

- Sequence of events
  - Requirements gathering
  - Quick design
  - Construction of prototype
  - Customer Evaluation
  - Refining prototype
  - Risk analysis

Prototyping



Prototyping

- The designer and the user is uncertain about the requirements and it is likely to evolve as the design progresses.
- Problems
  - Pressure to implement an early prototype
  - Function or extras being added to the system that are not included in the initial requirements document
  - Finished system may have poor controls
  - Change control often becomes much more complicated
  - Prototype becomes the finished system

Rapid Application Development (RAD)

- ✓ Methodology that enables organizations to develop strategically important systems faster while reducing development costs and maintaining quality
- ✓ Techniques include
  - Small, well-trained development teams
  - Evolutionary prototypes
  - Integrated power tools that support modeling, prototyping and component re-usability
  - A central repository
  - Interactive requirements and design workshops
  - Rigid limits on development time frames

Rapid Application Development (RAD)

- RAD methodology has four stages:
  - Definition of scope
  - Functional design
  - Construction of application
  - Deployment
- Drawbacks of RAD
  - For large projects, requirement of human resources is very high.
  - Requires strong commitment from customer and developer.
  - Monolithic applications are not suitable for RAD
  - RAD is not recommended for mission critical applications.

Reengineering & Reverse Engineering

- Reengineering
  - Process of updating an existing system by extracting and reusing design and program components.
- Reverse Engineering
  - Reverse engineering is the technique of drawing design specifications from the actual product
  - Process of taking apart an application or a software application or a product, in order to see how it functions and to use that information to develop a similar system

Structured Analysis

- ✓ Framework for the physical components (data and process) of an application using data flow diagrams.
- ✓ Project team member leads one or more team members through a segment of a deliverable s/he has produced can be implemented.

Information Systems Maintenance Practices

- ✓ Change Control
- ✓ Continuous Update of Systems Documentation
- ✓ Program Migration Process
- ✓ Testing Program Changes
- ✓ Library Control Software
- ✓ Executable and Source Code Integrity
- ✓ Program Code Comparison
  - Source Code Comparison
  - Object Code Comparison
- ✓ Emergency Changes

Change Control

- ✓ Users should convey system change requests to the system management using formal correspondence (change request form)
- ✓ An appropriate level of management should authorize changes to all production programs
- ✓ User and system management should review such changes.
- ✓ Maintenance records of all program changes should exist either manually or automatically
- ✓ Programmers should not have write, modify or delete access to production data.

Update of Systems Documentation

- ✓ System documentation be kept up-to-date
- ✓ Documentation may consist of
  - Program and/or system flowcharts
  - Program narratives
  - Data dictionaries
  - Entity-relationship models
  - Data flow diagrams (DFDs)
  - Operator run books
  - End-user procedural manuals
- ✓ Documentation stored off-site for disaster recovery purposes should also be updated.

Program Migration Process

- Once user management has approved the change, the modified programs are moved into the production environment
- Migration (from test to production) should be performed a group that is independent of computer programming team.
  - Groups such as computer operations, quality assurance or a designated change control group should perform this function.
- Proper access restrictions must be in place to ensure that only authorized individuals have the ability to migrate programs into production

Testing Program Changes I

- Controls should in place to protect production application programs from unauthorized changes
- Control objective are:
  - Access to program libraries should be restricted
  - Supervisory reviews should be conducted.
  - Change requests should be approved and documented.

Testing Program Changes II

- Change Request Form:
  - The change specifications should be adequately described, a cost analysis developed and a target data established.
  - The change from should be signed by the user to designate approval.
  - The change from should be reviewed and approved by programming management.
  - The work should be assigned to an analyst, programmer and programming group leader for supervision.

Testing Program Changes III

- IS Auditor's Role
  - Select a sample of program changes made during the audit period and trace them to the maintenance form to determine whether the changes are authorized
  - If an independent group updates the program changes in production, the IS Auditor must determine whether procedures exist to ensure possession of the change request from before the update.

Library Control Software

- Purpose of library control software is to separate production libraries from test libraries
- Ensure that all the changes to the programs are authorized
- Library control software:
  - Prohibits programmers from accessing production source and object libraries
  - Requires the control group or operator to release source code and place it in the programmer's library
  - Allows read-only access to source code

Executable and Source Code Integrity

- ✓ The timestamp of the source's module should not be later than that of the corresponding executable module
- ✓ User or application programmer should not have access to production source code or load libraries
- ✓ IS Auditors must review with special care those applications developed by autonomous end users, especially when the applications can be used to access or manipulate central data files.

Program Code Comparison

- ✓ Source Code Comparison and Object Code Comparison
- ✓ Source code comparison software is an effective and easy-to-use method for tracing changes to programs.
- ✓ After comparison, a list of additions, deletions and changes is printed and used by the auditor to review such change authorization approval documentation as work requests, approvals and program testing.

Emergency Changes

- ✓ Procedures should exist to ensure emergency fixes can be performed without compromising the integrity of the system
- ✓ This typically involves the use of special logon-Ids that grant temporary access to the production environment during these emergency situations.
- ✓ As emergency Ids possess powerful privileges, their use should be logged and carefully monitored

Project Management Tools and Techniques

Chapter 31

Topics

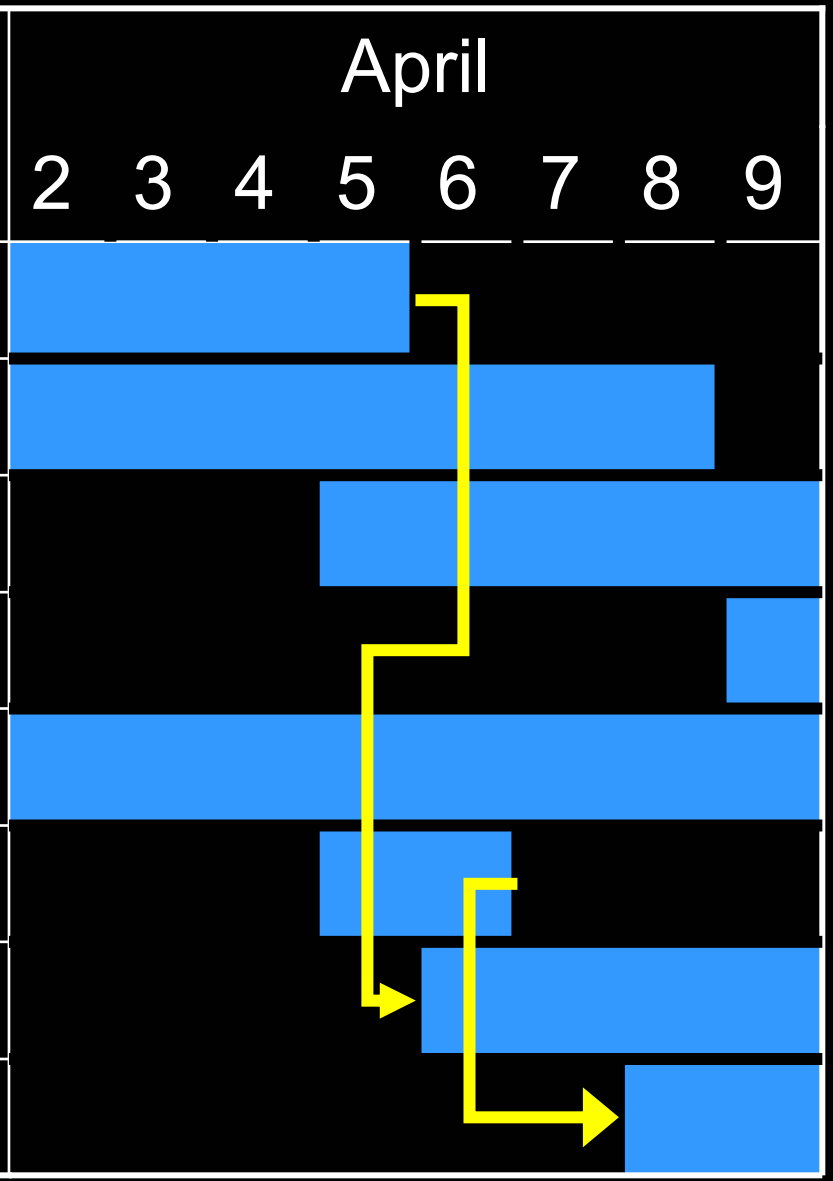
- ✓ Budgets and Schedules
- ✓ GANTT Charts
- ✓ Program Evaluation Review Technique (PERT)
- ✓ Critical Path Method (CPM)
- ✓ System Development Tools and Productivity Aids
  - Code Generators
  - Computer Aided Software Engineering (CASE)
  - Fourth Generation Languages (4GLs)

Budgets and Schedules

- Estimating the amount of effort that will be required to carry out each task
- Estimates for
  - Person hours by type (system analyst, programmer…)
  - Machine hours (computer time, office equipment, communication equipment)
- Function Point Analysis is a measure of the size of an information system based on the number and complexity of the inputs, outputs and files that a user sees and interacts with.

Gantt Chart

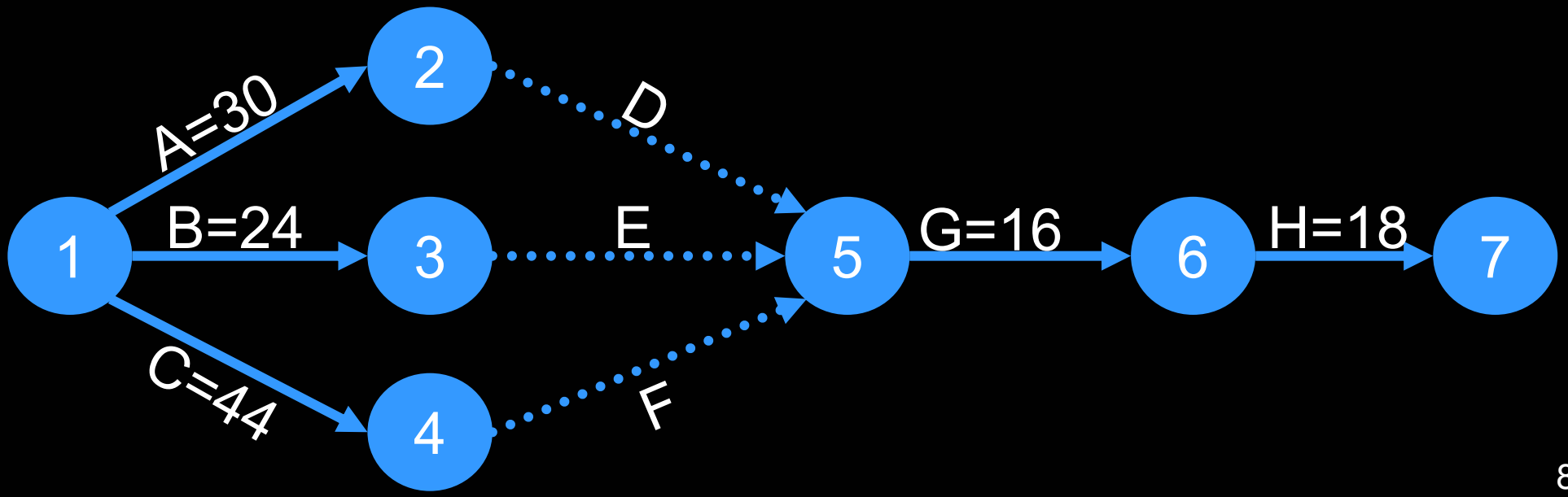
ID		Start	Finish
1	Task 1	2-Apr	5-Apr
2	Task 2	2-Apr	8-Apr
3	Task 3	5-Apr	9-Apr
4	Task 4	9-Apr	9-Apr
5	Task 5	2-Apr	9-Apr
6	Task 6	5-Apr	6-Apr
7	Task 7	6-Apr	9-Apr
8	Task 9	8-Apr	9-Apr



Program Evaluation Review Technique

- Activity
  - Successor
  - Predecessor
  - Dummy
- Event
- Slack
- Time Estimates

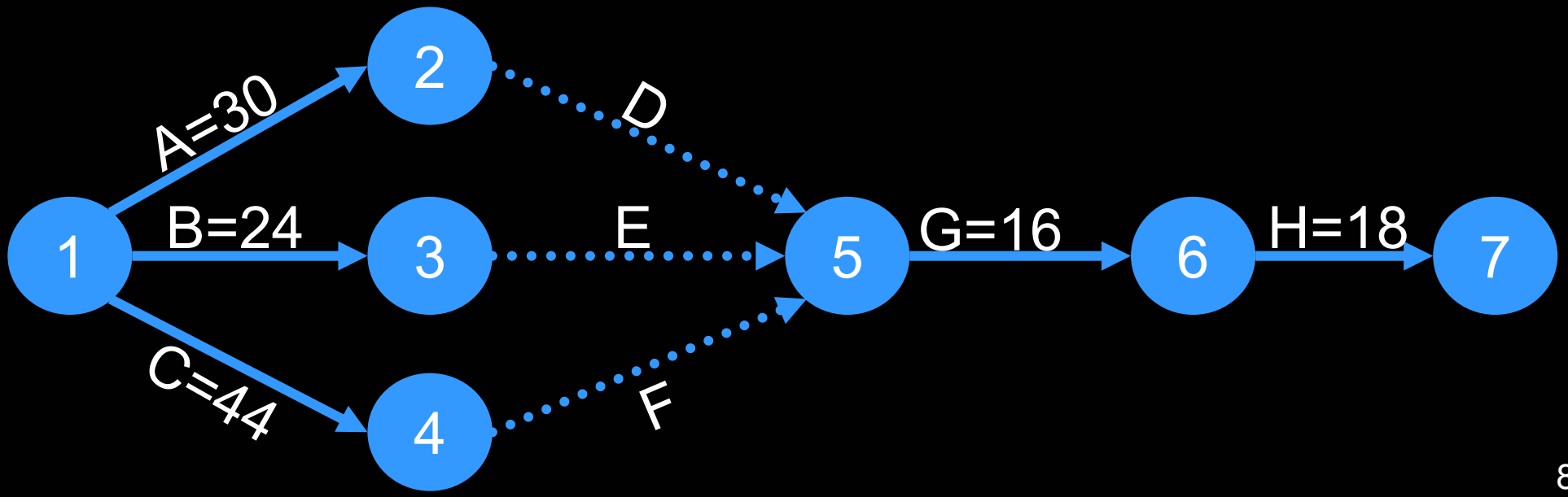
Activity	Duration	Predecessor
A	30	
B	24	
C	44	
G	16	A, B, C
H	18	G



Critical Path Method (CPM)

- ✓ Path which has the highest duration of time among different paths.
- ✓ Activities in critical path have zero slack.

Path	Duration	Slack Time
1-2-5-6-7 A-G-H	64	24
1-3-5-6-7 B-G-H	58	20
1-4-5-6-7 C-G-H	78	0



System Development Tools and Aids

- ✓ Code Generators
- ✓ CASE Tools
- ✓ 4GLs

Code Generators

- Code generators are tools that generate program code base upon parameters defined by a systems analyst or on data/entity flow diagrams developed by the design module of a CASE product.

CASE Computer Aided Software Engineering

- Upper CASE
  - Describe and document business and application requirements
  - Data object and Process definitions and relationships
- Middle CASE
  - Develop the detailed designs
  - Screen and report layouts
- Lower CASE
  - Generation of program code and database definitions
  - Generate program logic, data file formats or entire applications.

CASE Advantages and Disadvantages

- Advantages of using CASE
  - Enforces SDLC discipline
  - Standardization of processes
  - Consistency of application quality
  - Monotonous tasks such as diagramming, can be automated
  - Documentation is easier
  - Productivity gains
- Disadvantages of CASE
  - CASE tools are costly
  - Use of CASE tool requires extensive training

4GLs Characteristics

- Non-Procedural Language
  - Most 4GLs do not obey the procedural paradigm of continuous statement execution, subroutine call and control structures.
  - They are event driven and make extensive use of object-oriented programming
  - Unlike traditional languages, the 4GL can lack the lower level detail commands necessary to perform certain types of data intensive or online operations
- Environmental Independence (Portability)

4GLs Characteristics

- ✓ Software Facilities
  - Ability to design or paint retrieval screen formats, develop computer aided training routines or help screens and produce graphical outputs.
- ✓ Programmer Workbench Concepts
  - Easy filing facilities, temporary storage, text editing and operating system commands. Integrated development environment (IDE).
- ✓ Simple Language Subsets
  - 4GLs generally have simple language subsets which can be used by less skilled users in an information center.

4GLs Classification

- Query and Report Generator
  - Extract and produce reports (audit software)
- Embedded Database 4GLs
  - These depend upon self-contained database management systems. Examples include FOCUS, RAMIS II and NOMAD 2.
- Relational Database 4GLs
  - Optional feature on a vendor's DBMS product line. Examples include SQL+, MANTIS and NATURAL.
- Application Generators
  - Development tools generate lower level programming languages (3GLs) such as COBOL and C. Data processing development personnel, and not end users, use application generators.

Specialized Systems

Chapter 32

Topics

- ✓ Artificial Intelligence
- ✓ Expert Systems
- ✓ Data Warehouse
- ✓ Decision Support Systems

Artificial Intelligence

- ✓ Cognitive Science
- ✓ Robotics
- ✓ Natural Language Processing

Cognitive Science

- Expert Systems
  - These are information systems with reasoning capability. These are discussed in detail later.
- Learning Systems
  - Modify their behaviour based on information they acquire
- Fuzzy Logic
  - These are systems that can process data that are ambiguous and incomplete.

Cognitive Science

- Neural Networks
  - Modelled after the human
  - Pattern recognition, credit risk determination.
- Genetic Algorithms
  - Simulate biological evolutionary processes
- Intelligent Agents
  - Intelligent agents are software that use built-in and learned knowledge base about a person or process to make decisions and accomplish tasks in a way that fulfils the intentions of user.

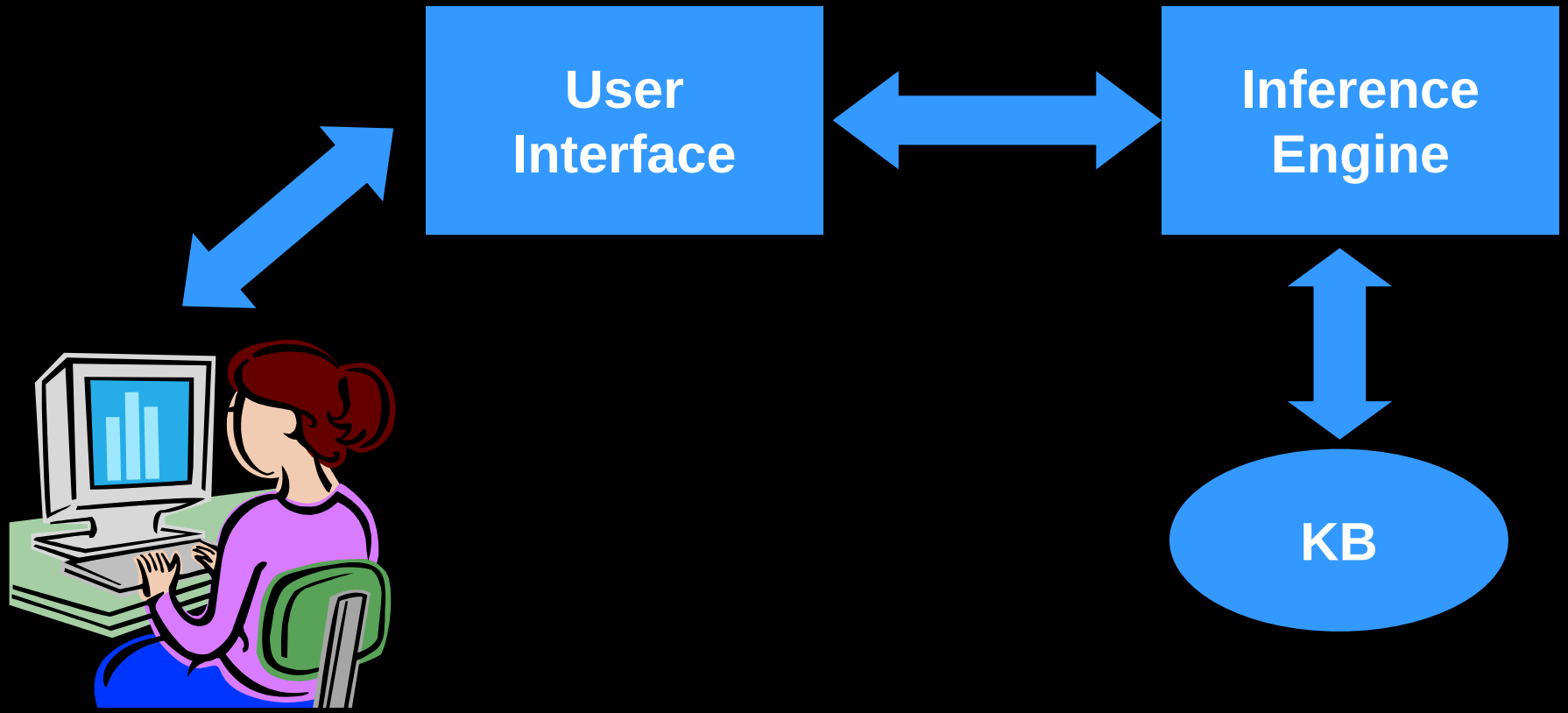
Robotics

- This technology to produce machines with human-like physical capabilities
  - Visual Processing
  - Tactility
  - Dexterity
  - Locomotion
  - Navigation

Natural Language Processing

- Technology to talk to computers in conversational human languages and make them understand.
  - Language Understanding
  - Speech Recognition
  - Machine Translation

Components Of an ES



Inference Engine

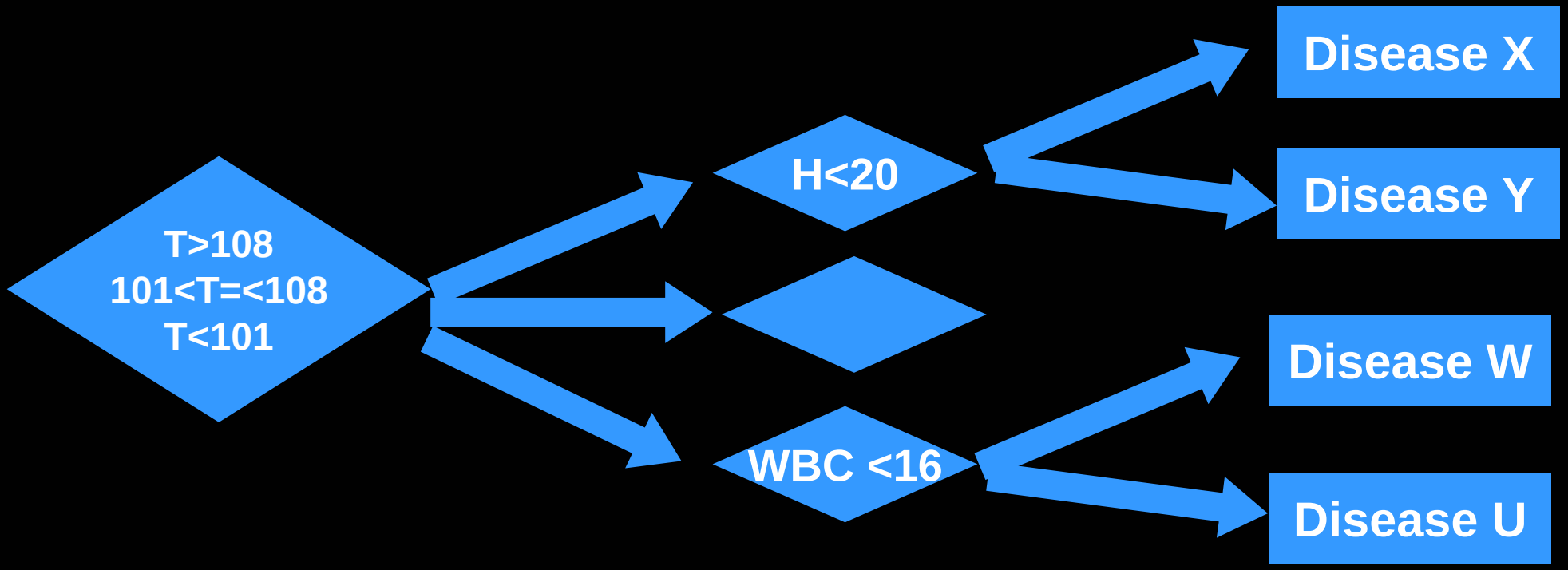
- ✓ Processes the rule and facts related to the specific problem
- ✓ It makes associations and inferences resulting in recommended courses of action

Knowledge Base

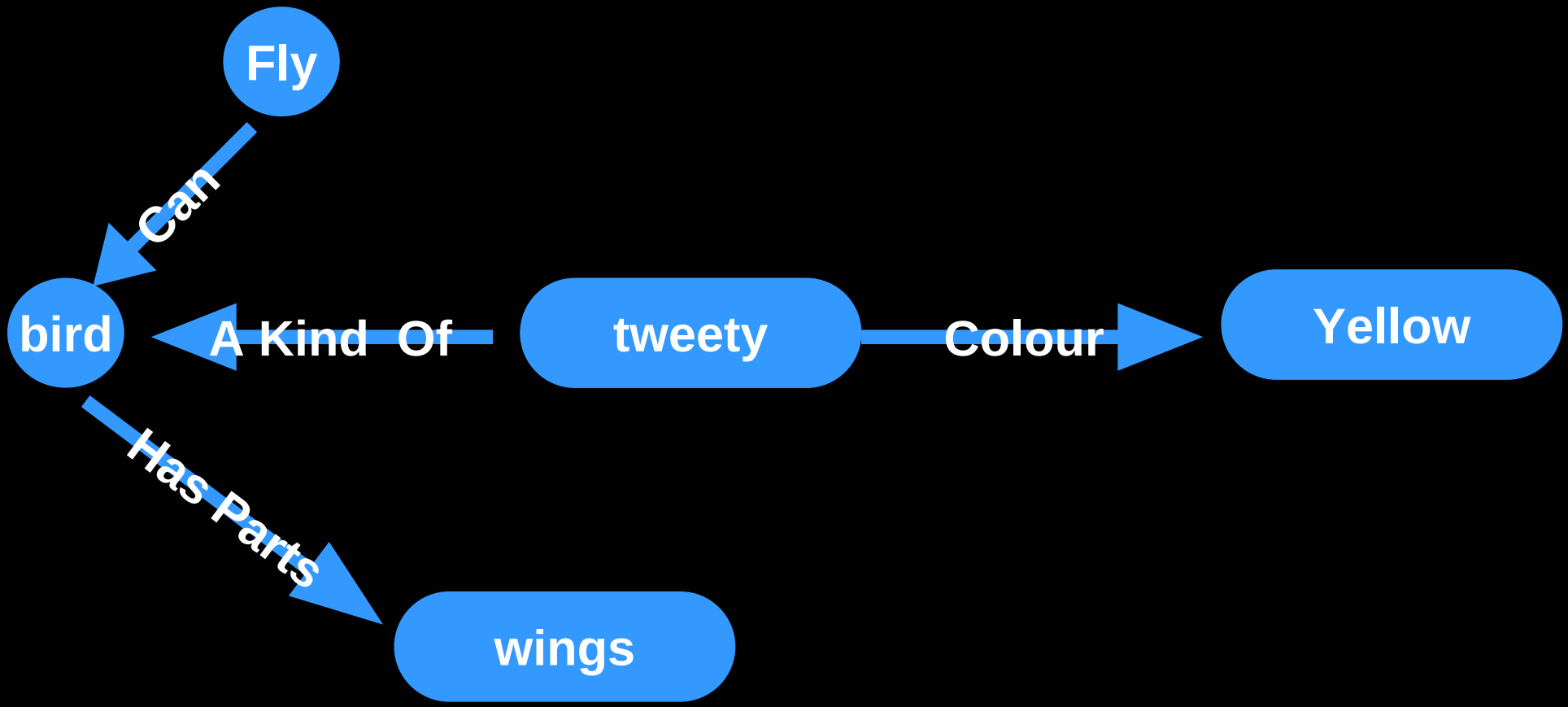
- ✓ Facts about a domain
- ✓ Heuristics/Reasoning Procedures
- ✓ Representation
  - ► Decision Tree
  - ► Rules
  - ► Semantic Nets

KB Representation

- ✓ Decision Tree
- ✓ Rules
  - ★ IF (Temp>108 and WBC>16)
    Then Probability of Disease U is 0.6



Semantic Nets



Other Components

- ✓ Knowledge Acquisition
  - ► Knowledge Interface
  - ► Data Interface

Expert Systems

- ✓ Benefits to an Organization:
  - Knowledge Capture
  - Sharing Knowledge
  - Consistency and Efficiency in decision making
  - Productivity and performance gains
- ✓ Applications
  - Portfolio analysis
  - Demographic forecasts
  - Help Desk operations
  - Medical Diagnostics

IS Auditor's Role

- ✓ Understand the purpose and functionality of the system
- ✓ Assess system's significance and potential risks.
- ✓ Review
  - Adherence to corporate policies and procedures.
  - Decision logic
  - Basic assumptions and formulas
  - Expertise
  - Updating procedures for the KB
  - Security Access

Data Warehouse

- ✓ Definition
  - ► A Data Warehouse is a subject-oriented, integrated, time-invariant, non-volatile collection of data in support of management's decision-making process

Composition:

- ✓ Data sources
  - ► Existing databases
  - ► External information
- ✓ Data extraction and transformation
  - ► Extraction
  - ► Restructuring
  - ► Cleansing
- ✓ Target database
  - ► Data warehouse (Enterprise-wide)
  - ► Data mart (division/department level)
- ✓ End user data access

Considerations

- ✓ Data sources credibility
- ✓ Accuracy of source data
- ✓ Accuracy of extraction and transformation processes
- ✓ Data stewardship

Decision Support System

- ✓ Characteristics
  - ► Less structured problems
  - ► Models or analytic techniques coupled with data access and retrieval functions
  - ► Emphasizes flexibility and adaptability
- ✓ More effective than efficient

Gorry-Scott-Morton Framework I

- ✓ Decision structure
  - Structured
  - Semi-structured
  - Unstructured
- ✓ Management level
  - Operational control
  - Management/Tactical control
  - Strategic planning

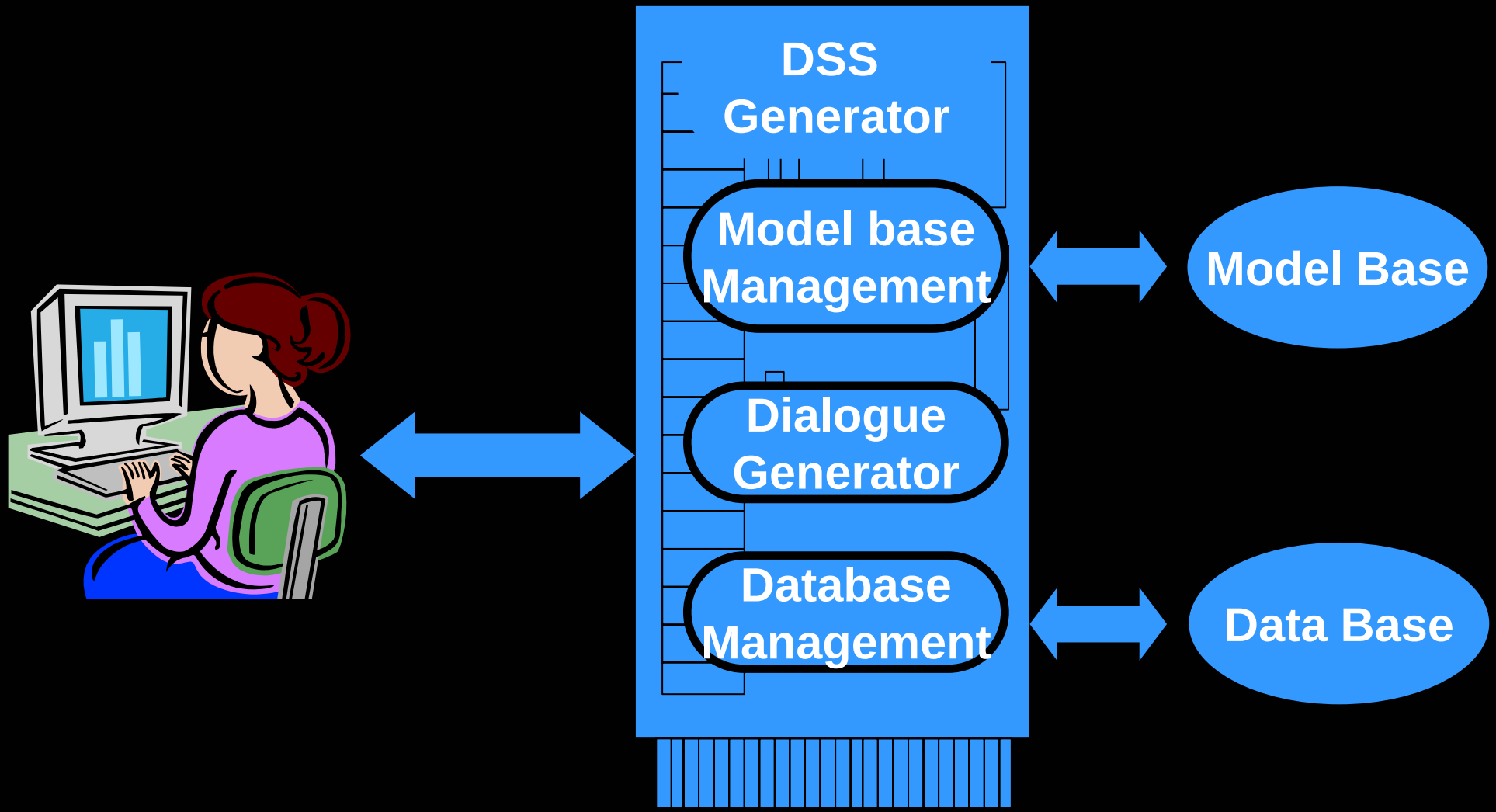
Gorry-Scott-Morton Framework II

Unstructured

Semi-Structured

Structured

Components I



Components II

- ✓ DSS Generator
  - ► Main Software Resource. Links models together
- ✓ Database Management
  - ► Creation, Use and maintenance
- ✓ Model Base Management
  - ► Development, storage and retrieval, revision and control of the models in a DSS
- ✓ Dialogue Management
  - ► Generation and control of the dialogue at the user level.

Risk Factors

- Nonexistent or unwilling users
- Multiple users or implementers
- Disappearing users, implementers or maintainers
- Inability to specify purpose or usage patterns in advance
- Inability to predict and cushion impact on all parties
- Lack or loss of support
- Lack of experience with similar systems
- Technical problems and cost-effectiveness issues

IS Auditor's Role in Systems Development

Chapter 33

Topics

- ✓ IS Auditor's Role in
  - Systems Development, Acquisition and Maintenance
  - Reviewing Feasibility Study
  - Reviewing the Requirement Definition
  - Reviewing the Detailed Design and Programming Phases
  - Reviewing the Testing Phase
  - Reviewing the Implementation Phase
  - Reviewing the Post-implementation Review
  - Reviewing System Change Procedures and Program Migration Process

Role in SDLC

- ✓ Determining main components, objectives and user requirements
- ✓ Determine and rank major risks to, and exposures of, the system
- ✓ Identify controls to mitigate the risks to, and exposures of, the system
- ✓ Advise the project team regarding the design of the system and implementation of controls
- ✓ Monitor the systems development process
- ✓ Participation in the post implementation reviews.
- ✓ Evaluate system maintenance standards and procedures
- ✓ Test system maintenance procedures to ensure that they are being applied
- ✓ Evaluate the system maintenance process
- ✓ Determine the adequacy of production library security

Reviewing Feasibility Study

- ✓ The IS Auditors should review the documentation produced in this phase for reasonableness.
- ✓ All cost justifications / benefits should be verifiable and be presented showing anticipated benefits to be realized.
- ✓ Identify and determine the criticality of the need
- ✓ Determine if a solution can be achieved with systems already in place. If not, review the evaluation of alternative solutions for reasonableness.
- ✓ Determine the reasonableness of the chosen solution

Reviewing the Requirement Definition

- ✓ Obtain the detailed requirements definition document and verify its accuracy
- ✓ Verify that project initiation and cost have receive proper management approval.
- ✓ Review the conceptual design specifications (transforms, data descriptions) to ensure they address the needs of the user.
- ✓ Review the conceptual design to ensure that control specifications have been defined
- ✓ Determine whether the application is a candidate for the use of embedded audit routine(s). If so, request that the routine be incorporated in the conceptual design of the system.

Reviewing Detailed Design & Programming

- ✓ Review the system flowcharts for adherence to the general design.
- ✓ Verify that appropriate approvals were obtained
- ✓ Review the input, processing and output controls
- ✓ Interview the key users of the system to determine their understanding of how the system will operate and assess their level of input into the design of screen formats and output reports.
- ✓ Assess the adequacy of audit trails
- ✓ Verify the integrity of key calculations and processes
- ✓ Verify that the system can identify and process erroneous data correctly
- ✓ Verify that all recommended corrections to programming errors were made and that the recommended audit trails or embedded audit modules were coded into the appropriate programs.

Reviewing the Testing Phase

- ✓ Review the test plan for completeness
- ✓ Reconciliation of control totals and converted data
- ✓ Review error reports for their precision in recognizing erroneous data and for resolution of errors
- ✓ Verify cyclical processing for correctness (month-end, year-end processing, etc.)
- ✓ Review system and end user documentation
- ✓ Review parallel testing results for accuracy
- ✓ Verify that system security is functioning as designed by developing and executing access tests
- ✓ Review unit and system test plans

Reviewing the Implementation Phase

- ✓ Review the programmed procedures used for scheduling and running the system along with system parameters used in executing the production schedule
- ✓ Review all system documentation to ensure its completeness and to ensure that all recent updates for the testing phase have been incorporated
- ✓ Verity all conversion of data to ensure they are correct and complete before implementing the system in production

Reviewing the Post-implementation Review

- ✓ Determine if the system's objectives and requirements wee achieved.
- ✓ Determine if the cost benefits identified in the feasibility study are being measured analyzed and accurately reported to management.
- ✓ Review program change requests
- ✓ Review controls built into the system to ensure that they are operating according to design
- ✓ Review operators' error logs

Change Procedures and Program Migration

- ✓ Evaluate the adequacy of the organization's procedures for authorizing, prioritizing and tracking system changes
- ✓ Identify system changes and verify that appropriate authorization was given to make the change in accordance with organizational standards
- ✓ Review technical and user documentation to ensure that it is up-to-date and accurately reflects current system functionality
- ✓ Evaluate the adequacy of the organization's procedures regarding program change control
- ✓ Evaluate the adequacy of the security access restrictions over production source and executable modules

Change Procedures and Program Migration

- ✓ Evaluate the adequacy of the organization's procedures for dealing with emergency program changes
- ✓ Evaluate the adequacy of the security access restrictions over the use of the emergency logon-Ids.
- ✓ Evaluate the adequacy of the procedures in place for testing system changes
- ✓ Review evidence (test plans and test results) to ensure that procedures are carried out as prescribed by the organizational standards
- ✓ Review the procedures established for ensuring executable and source code integrity
- ✓ Review production executable modules and verify that there is one and only one corresponding version of the program source code.

Role in Project Management

- ✓ Collect documentation of each phase and check the for adequacy and completion
- ✓ Attend project meetings to check the compliance of the development process
- ✓ Advise the team on adequate and cost effective control measures
- ✓ Represent the management interest in the team by continuously assessing the ability of the team to deliver to agreed deliverables on committed dates.

Thank You

Mathew A Thomas
mat@vsnl.com