SYSTEM ANALYSIS

PHASE II

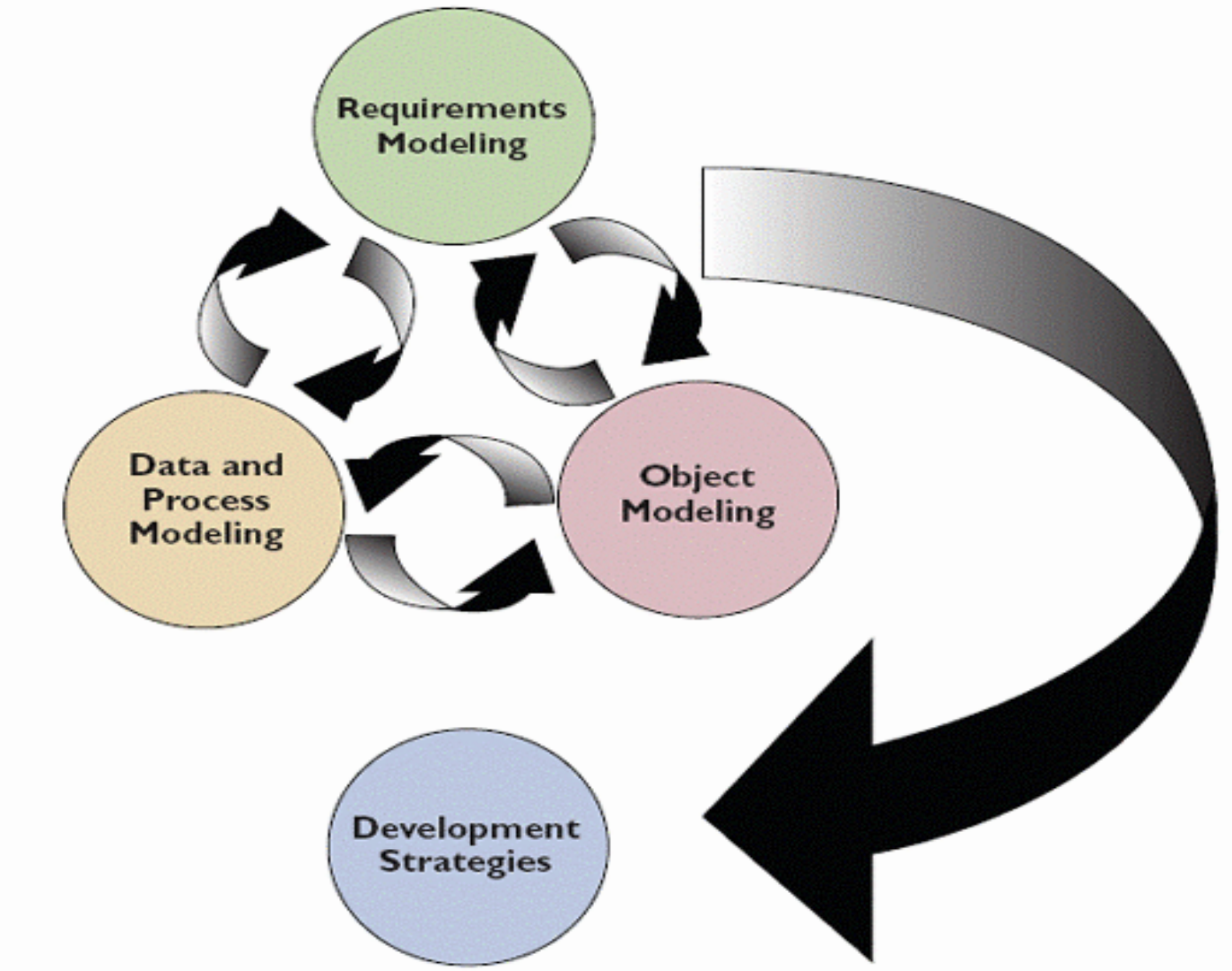
McRoyal Enoch Mensah
for NUGS president
hopeful

Introduction

McRoyal Enoch Mensah
for NUGS president
hopeful

- The overall objective of the systems analysis phase is to understand the proposed project, ensure that it will support business requirements, and build a solid foundation for system development.
- In this phase, models and other documentation tools are used to visualize and describe the proposed system.
- The system analysis phase includes four main activities; requirements modeling, data and process modeling, object modeling and consideration of development strategies.

System Analysis Phase Tasks



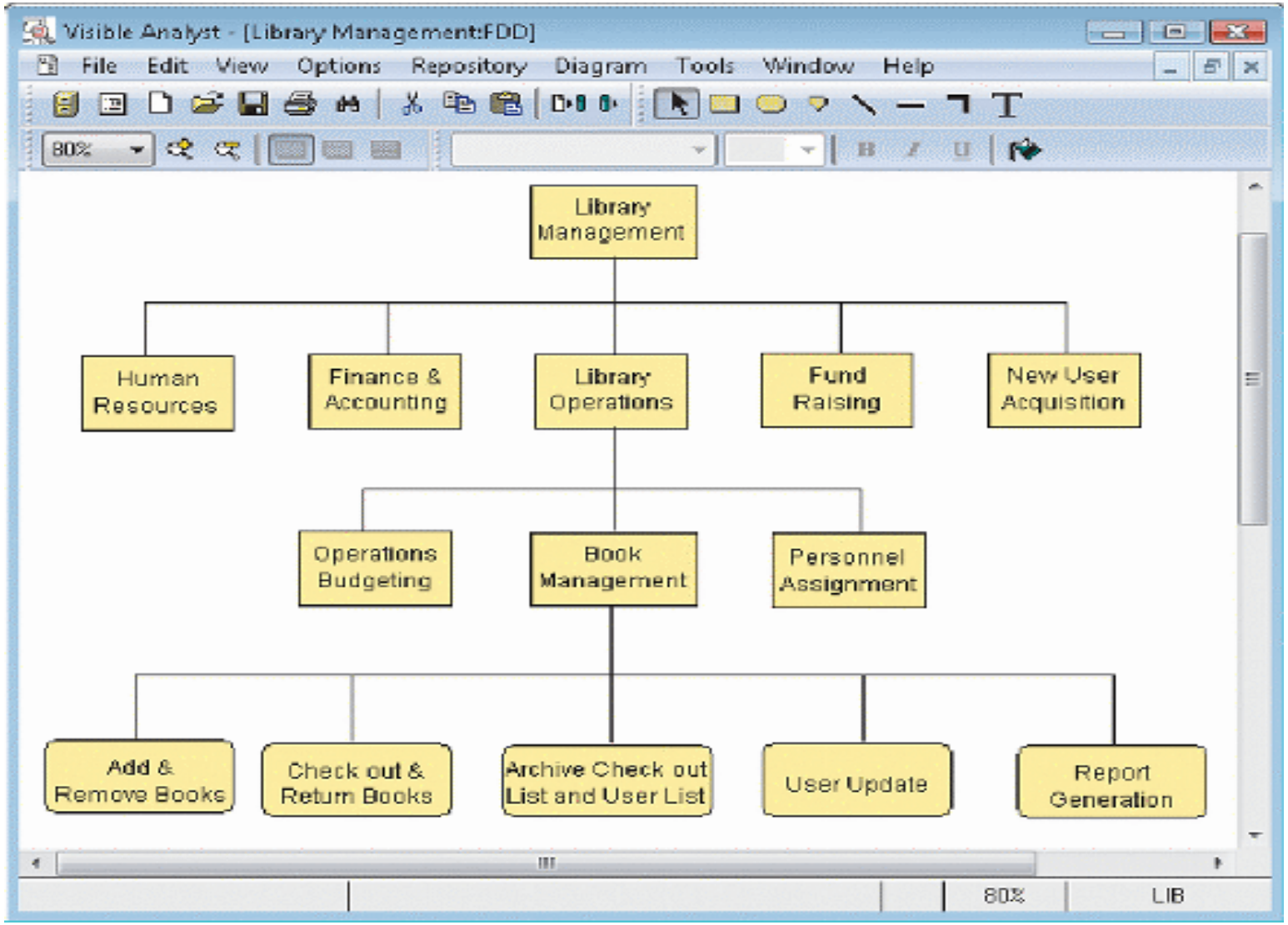
Modeling Tools and Techniques

- Modeling involves graphical methods and nontechnical language that represent the system at various stages of development.

Modeling Tools and Techniques (cont'd)

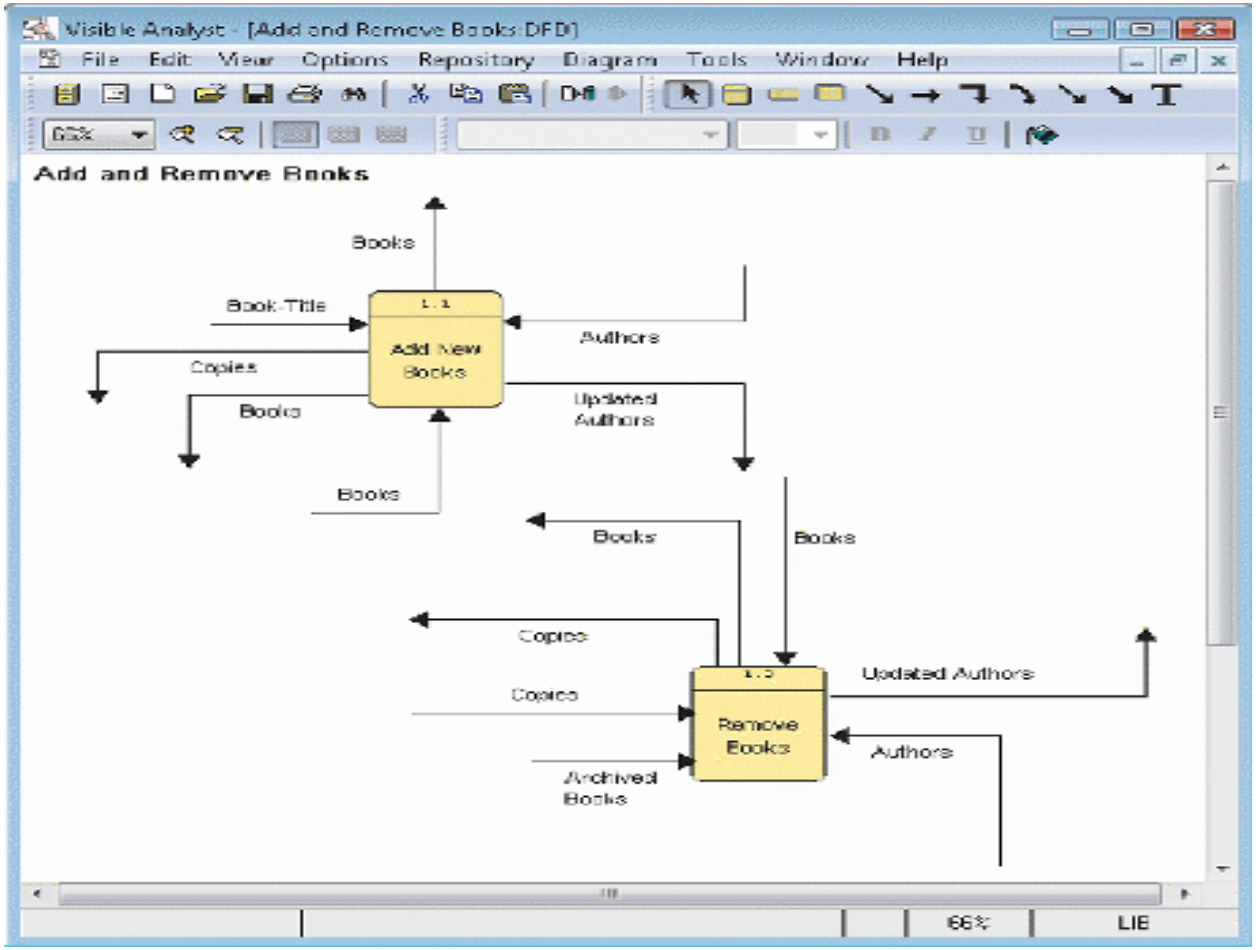
- CASE Tools
- Functional Decomposition Diagrams (FDD)
  - FDD or Structured Charts is a top-down representation of a function or process.
  - During requirements modeling analysts use FDD to model business functions and show how they are organized into lower-level processes. These processes translate into program modules during application development.

Four-level FDD of a Library System



Data Flow Diagrams (DFD)

- Working from a FDD, analysts can create DFDs to show how the system stores, processes and transforms data.



Unified Modeling Language (UML)

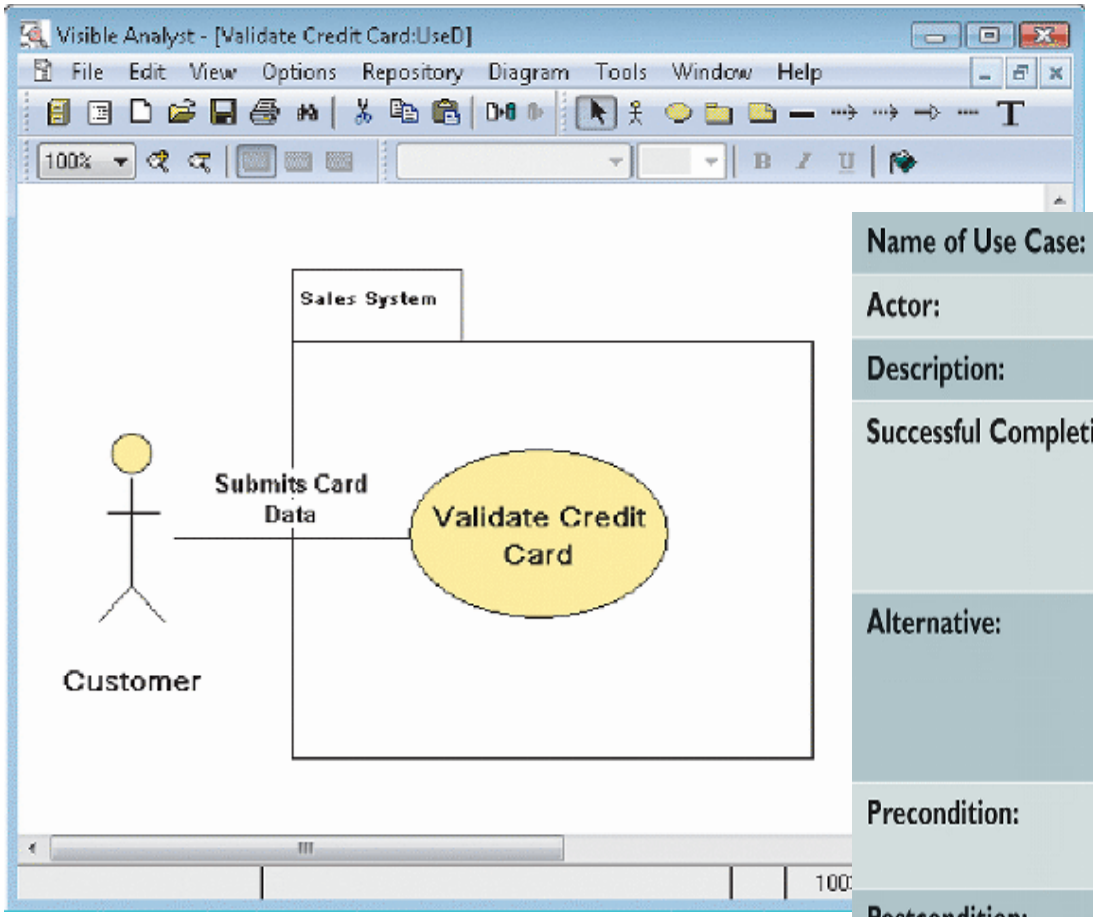
McRoyal Enoch Mensah
for NUGS president
hopeful

- Widely used method of visualizing and documenting software systems design.
- UML uses O-O design concepts but it is independent of any specific programming language and can be used to describe business processes and requirements generally.
- UML is used to represent an IS from a user's viewpoint.
- UML provides graphical tools such as;
  - **Use Case Diagrams**
  - **Sequence Diagrams**

Use Case Diagrams

- A **use case diagram** visually represents the interaction between users and the **IS.**
- The user becomes an **actor**, with a specific role that describes how he/she interacts with the system.

Use Case Diagram of a Sales System



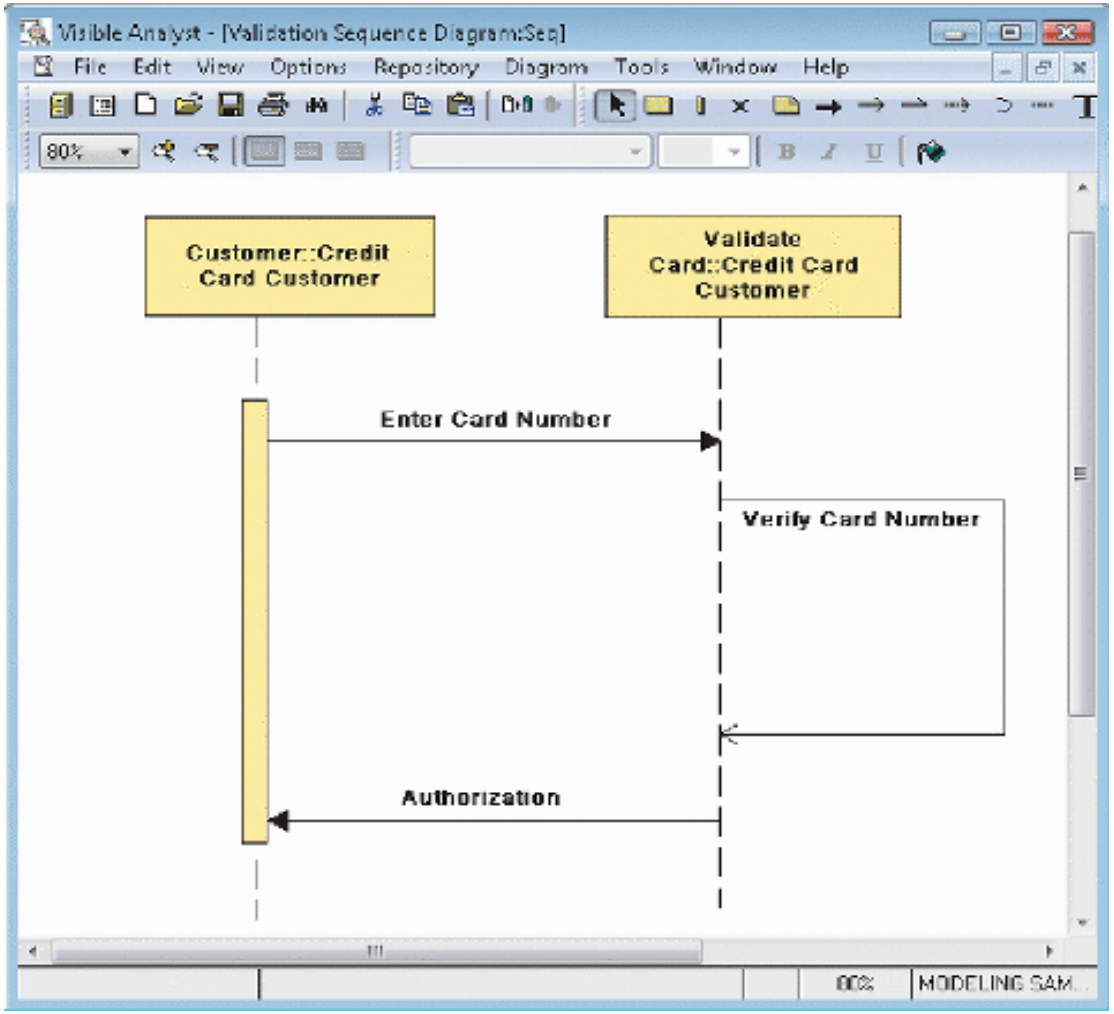
Name of Use Case:	Credit card validation process
Actor:	Customer
Description:	Describes the credit card validation process
Successful Completion:	1. Customer clicks the input selector and enters credit card number and expiration date 2. System verifies card 3. System sends authorization message
Alternative:	1. Customer clicks the input selector and enters credit card number and expiration date 2. System rejects card 3. System sends rejection message
Precondition:	Customer has selected at least one item and has proceeded to checkout area
Postcondition:	Credit card information has been validated Customer can continue with order
Assumptions:	None

Sequence Diagrams

- A sequence diagram shows the timing of interactions between objects as they occur.
- The interaction proceeds from top to bottom along a vertical timeline, while the horizontal arrow represent messages from one object to another.

A sequence diagram of a successful credit card validation

MrRoyal Enoch Mensah
for NUGS president
hopeful



Phase II- Part I:Requirements Modeling

- Requirements modeling involves fact-finding to describe the current system and identification of the requirements for the new system.
- A **system requirement** is a characteristic or feature that must be included in an IS to satisfy business requirements and be acceptable to users.
- System requirements fall into five general categories:
  - Outputs
  - Inputs
  - Processes
  - Performance
  - Control

System Requirements Checklist

- **Outputs:** Electronic or printed information produced by the system.
- Eg;
  - The Web site must report online volume statistics every four hours, and hourly during peak periods.
  - The inventory system must produce a daily report showing the part number, description, quantity on hand, quantity allocated, quantity available, and unit cost of all sorted by part number.

System Requirements Checklist(cont'd)

- **Inputs:** Necessary data that enters the system, either manually or in an automated manner.
- Eg;
  - Manufacturing employees must swipe their ID cards into online data collection terminals that record labor costs and calculate production efficiency.
  - The department head must enter overtime hours on a separate screen.

System Requirements Checklist (cont'd)

- **Processes:** The logical rules that are applied to transform the data into meaningful information.
- Eg;
  - The student records system must calculate the GPA at the end of each semester.
  - As the final step in year-end processing, the payroll system must update employee salaries, bonuses, and benefits and produce tax data required by the IRS.

System Requirements Checklist (cont'd)

- **Performance:** System characteristics such as speed, volume, capacity, availability and reliability.
- Eg;
  - The system must support 25 users online simultaneously
  - Response time must not exceed four seconds

System Requirements Checklist (cont'd)

- **Control**: Hardware, software and procedural controls that safeguard and protect the system and its data from internal or external threats.
- Eg;
  - The system must provide logon security at the operating system level and at the application level.
  - An employee record must be added, changed, or deleted only by a member of the human resources department.

Fact-Finding

- Fact-finding involves answers to five familiar questions: ***who***, ***what***, ***where***, ***when*** and ***how***.
- For each of these questions, another very important question that must be asked is: ***why***.

Fact-Finding (cont'd)

- Eg;
  - ***Who?*** Who performs each of the procedures within the system? Why? Are the correct people performing the activity? Could other people perform the tasks more effectively?
  - ***What?*** What is being done? What procedures are being followed? Why is that process necessary?
  - ***Where?*** Where are operations being performed? Why? Where could they be performed? Could they be performed more efficiently elsewhere?
  - ***When?*** When is a procedure performed? Why is it being performed at this time? Is this time the best time?
  - ***How?*** How is a procedure performed? Why is it performed in that manner? Could it be performed better, more efficiently, or less expensively in some other manner?

Fact-Finding Techniques

- Interviews
  - An interview is a planned meeting during which information from another person is obtained.
  - The interviewing process consists of seven steps:
    - Step 1: Determine the People to Interview
    - Step 2: Establish Objectives for the Interview

Fact-Finding Techniques (cont'd)

- Step 3: Develop Interview Questions
  - Avoid leading questions
  - Open-ended questions
  - Closed-ended questions
  - Range-of-response questions
- Step 4: Prepare for the Interview
- Step 5: Conduct the Interview
- Step 6: Document the Interview
- Step 7: Evaluate the Interview

Fact-Finding Techniques (cont'd)

- Document Review
  - Obtain copies of actual forms and operating documents currently in use. Review blank copies of forms, as well as samples of actual completed forms.
- Observation
  - Personal observation allows one to verify statements made in interviews and determine whether procedures really operate as they are described.
- Questionnaires and Surveys
  - Questionnaires also called surveys is a document containing a number of standard questions that can be sent to many individuals.

Fact-Finding Techniques (cont'd)

- Sampling
  - Samples include records, reports, operational logs, data entry documents, complaint summaries, work request, etc.
  - Sampling techniques include; systematic, stratified and random.
- Research

Interviews versus Questionnaires

- Advantages of Interviews
  - More familiar and personal
  - React immediately to anything the interviewee says
  - Watch for clues to determine if responses are knowledgeable and unbiased
  - Improved human relations.
- Disadvantages of Interviews
  - Costly and time-consuming process
  - Preparation and follow-up work is required

Interviews versus Questionnaires

- Advantages of Questionnaire
  - Opportunity to provide input and suggestions
  - Recipients can answer questions at their convenience
  - Allows anonymous responses
- Disadvantages of Questionnaire
  - Preparing a good questionnaire requires skill and time
  - Questions can be misinterpreted
  - Recipients might view them as intrusive, time-consuming and impersonal

Documentation

- As an analysts you should document your work according to the following principles;
  - Record information as soon as you obtain it.
  - Use the simplest recording method possible.
  - Record your findings in such a way that they can be understood by someone else.
  - Organize your documentation so that related material is located easily.

Software Tools

McRoyal Enoch Mensah
for NUGS president
hopeful

- Many software tools are available to help one record and document information;
- Eg;
  - **CASE Tools**: Use CASE tools for systems development.
  - **Word Processing**: Create reports, summaries, tables and forms.
  - **Spreadsheets**: Track and manage numerical data or financial information.
  - **Databases**: Manage information about events, observations and samples.
  - **Presentation Graphics**: Organizing and developing formal presentation.

Phase II - Part II – Data and Process Modeling

- Systems analysts use many graphical techniques to describe an information system.
- A data flow diagrams (DFD) uses various symbols to show how the system transforms input data into useful information.

Data Flow Diagrams

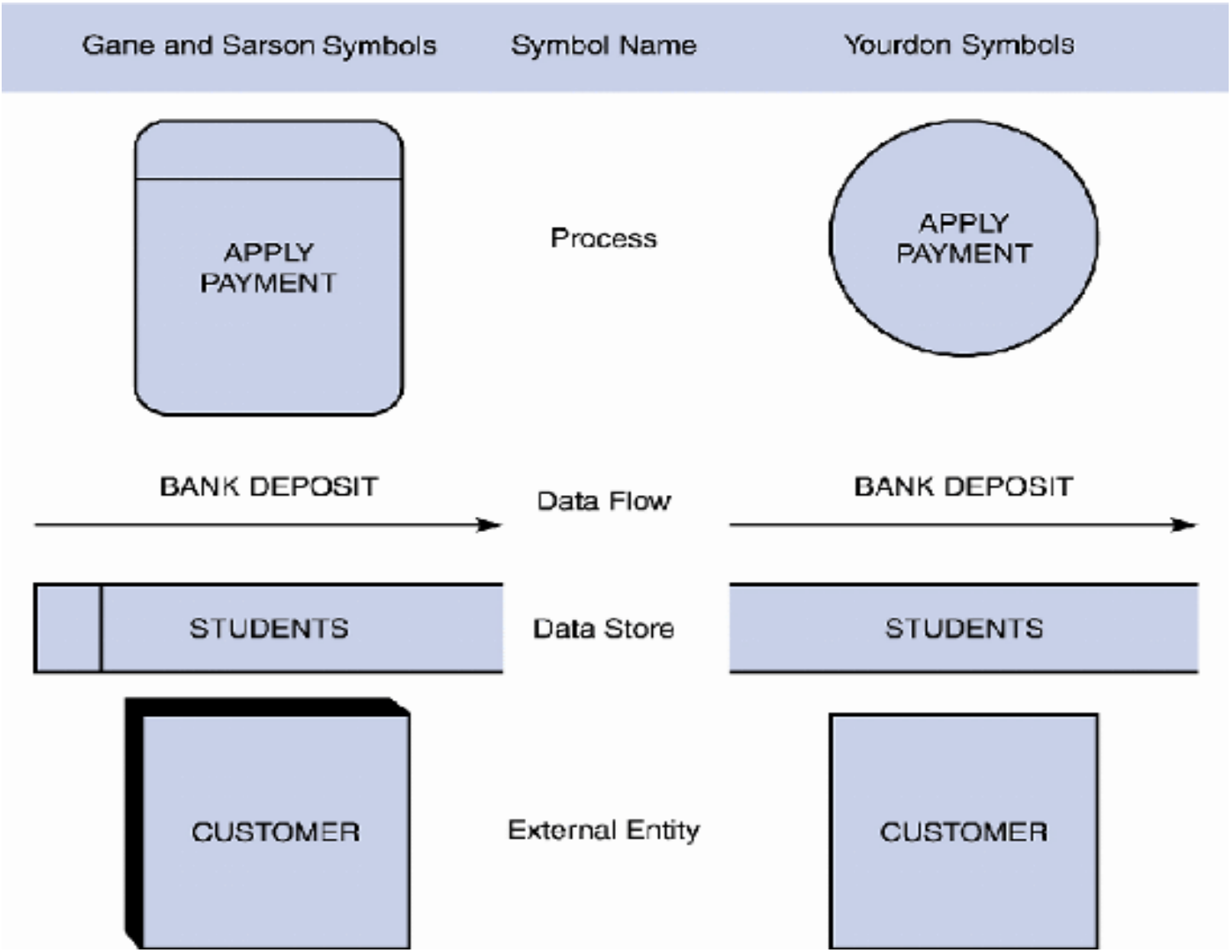
- A data flow diagram (DFD) is a graphical tool to depict the flow of data through a system and the work or processing performed by that system.
- It shows how data moves through an information system but does not show program logic or processing steps.
- It only represents a logical model that shows *what* the system does, not *how* it does it.

DFD Symbols

- DFDs use four basic symbols that represent processes, data flows, data stores, and entities.
  - Gane and Sarson symbol set
  - Yourdon symbol set
- Symbols are referenced by using all capital letters for the symbol name.

DFD Symbols (cont'd)

McRoyal Enaca Menan
for NUGS president
hopeful



DFD Symbols - Process

- A process is a work or action performed on input data flow to produce an output data flow.
- Use a verb to label the action performed by the process (not the name of the person or department who does it as in physical DFD).
- A process must have at least one input data flow and at least one output data flow.
- A process symbol can be referred to as a **black box**.
- Examples: APPLY RENT PAYMENT, VERIFY ORDER, ASSIGN FINAL GRADE.

DFD Symbols – Data Flow

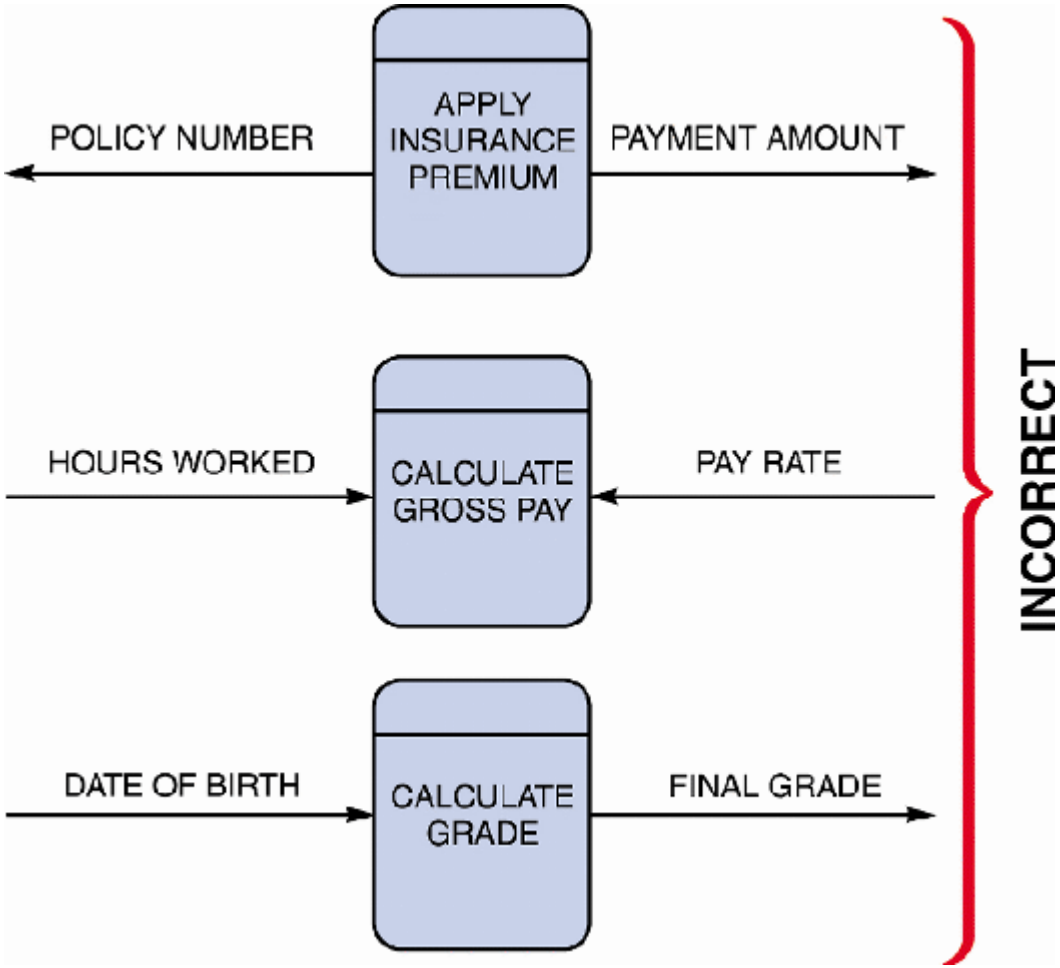
- A Data Flow represents a movement of data (information) among processes or data stores.
- A Data Flow does not represent a document or a physical good: it represents the exchange of information in the document or about the good.
- Data flow represents an input of data to a process, or the output of data from a process.
- Examples: DEPOSIT, INVOICE PAYMENT, STUDENT GRADE.

DFD Symbols – Data Flow (cont'd)

McRoyal Empoy Wensan for NUGS president hopeful

- The following data flow and process combinations must be avoided:
  - **Spontaneous generation** where a process produces output without but has no input data flow.
  - **Black hole** is a process that has input but produces no output.
  - **Gray hole** is a process that has at least one input and one output but the input is insufficient to generate the output shown.

Examples of incorrect combinations of data flow and process symbols



DFD Symbols – Data Stores

- A Data store is a storage of data: it contains information.
- Physical storage is immaterial: it can be a filing cabinet, book, computer file.
- A data store is an inventory of data.
  - A data store is "data at rest" compared to a data flow that is "data in motion".
  - Always almost one of the following: Persons (or group of persons), Places, Objects, Events(about which data is captured), Concepts (about which data is important)
- Examples: STUDENTS, ACCOUNTS RECEIVABLE, PRODUCTS.

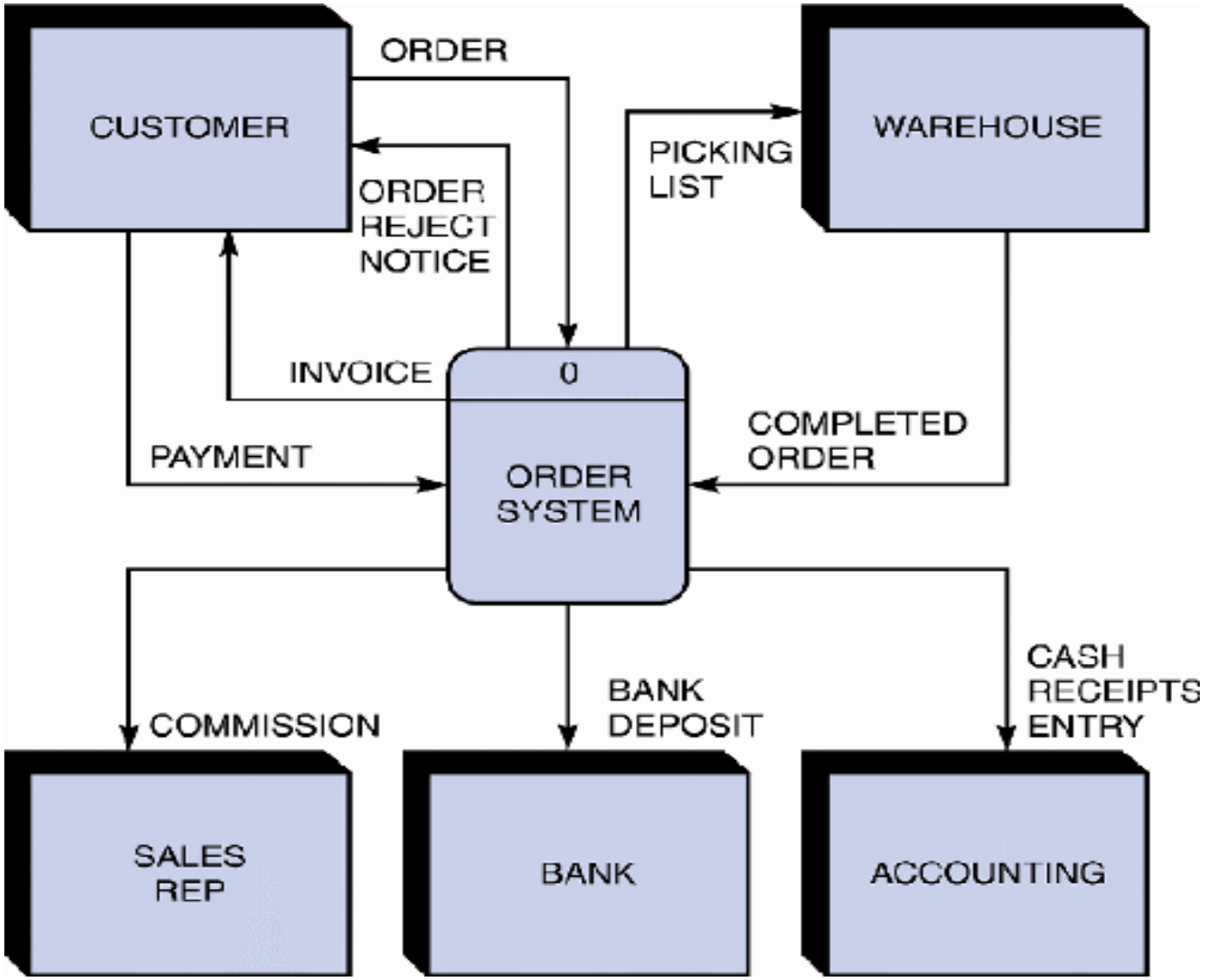
DFD Symbols – External Entity

- An external entity is a provider (source) or receiver (sink) of data and information of the system.
- An external entity is not part of the system: the externality depends on how the system is defined.
- An external entity (agent) defines a person, organization unit, or other organization that lies outside of the scope of the project but that interacts with the system being studied.
  - External agents define the "boundary" or scope of a system being modeled.
  - As scope changes external agents can become processes and vice versa.
- Examples: CUSTOMER, STUDENT, SUPPLIER

Creating DFDs

- Use the following three steps to create a set of DFDs
  - Step 1: Draw a context diagram
    - A **context diagram** is a top-level view of an IS that shows the system's boundaries and scope.
    - Single process (labeled "0") represents the entire system.
    - No data stores appears on context diagrams.

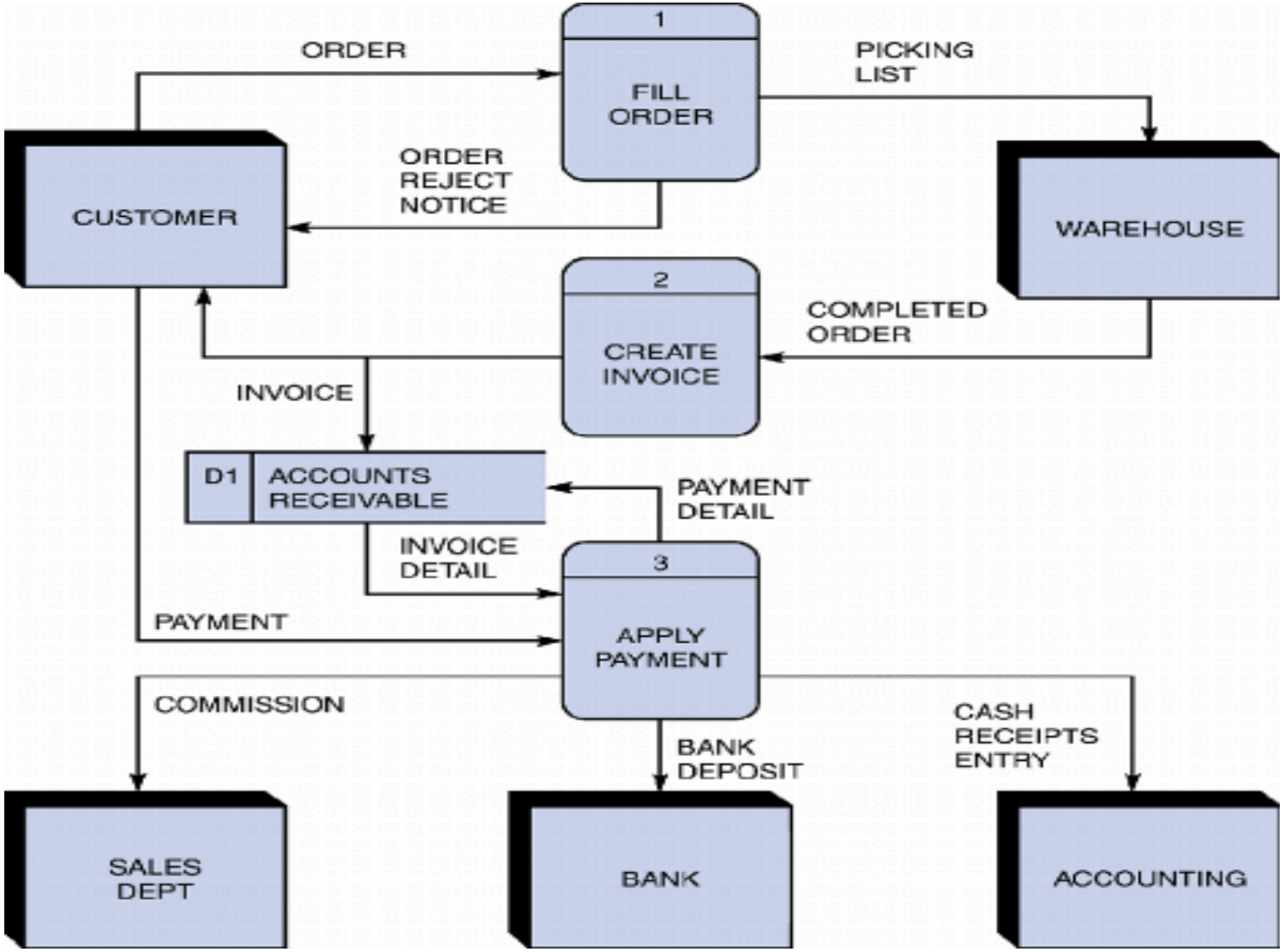
Context Diagram DFD for an Order System



Step 2: Draw a Diagram 0 DFD

- Diagram 0 zooms in on the system and shows major internal processes, data flows, and data stores.
- A **functional primitive** is a process that consists of a single function that is not exploded further.

Diagram 0 DFD for the Order System



Step 3: Draw the Lower-Level Diagrams

- To create lower-level diagrams, one must use **leveling** and **balancing** techniques.
- **Leveling** is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified.
- **Balancing** maintains consistency among a set of DFDs by ensuring that input and output data flows align properly.

Diagram 1 DFD shows details of the FILL ORDER process

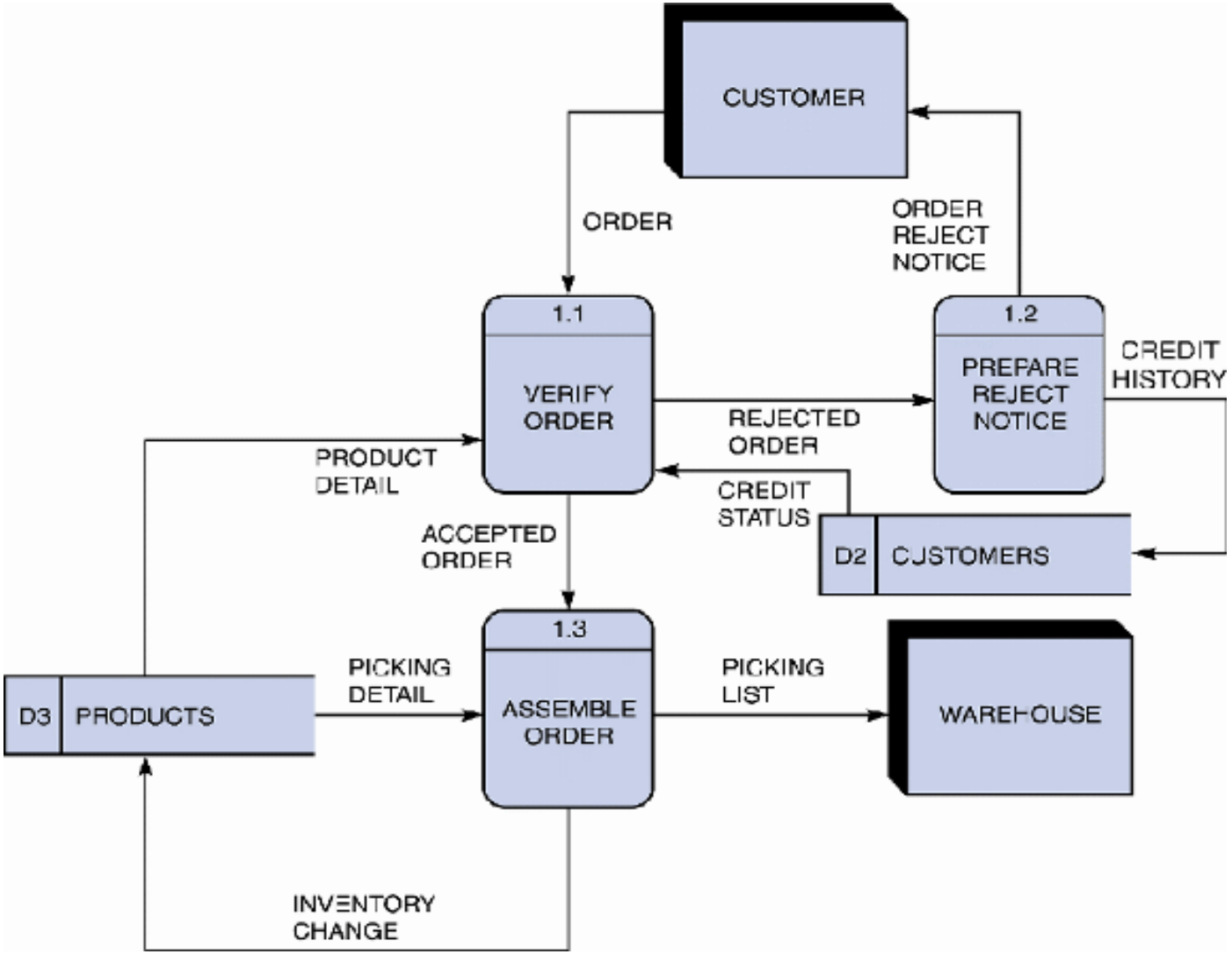
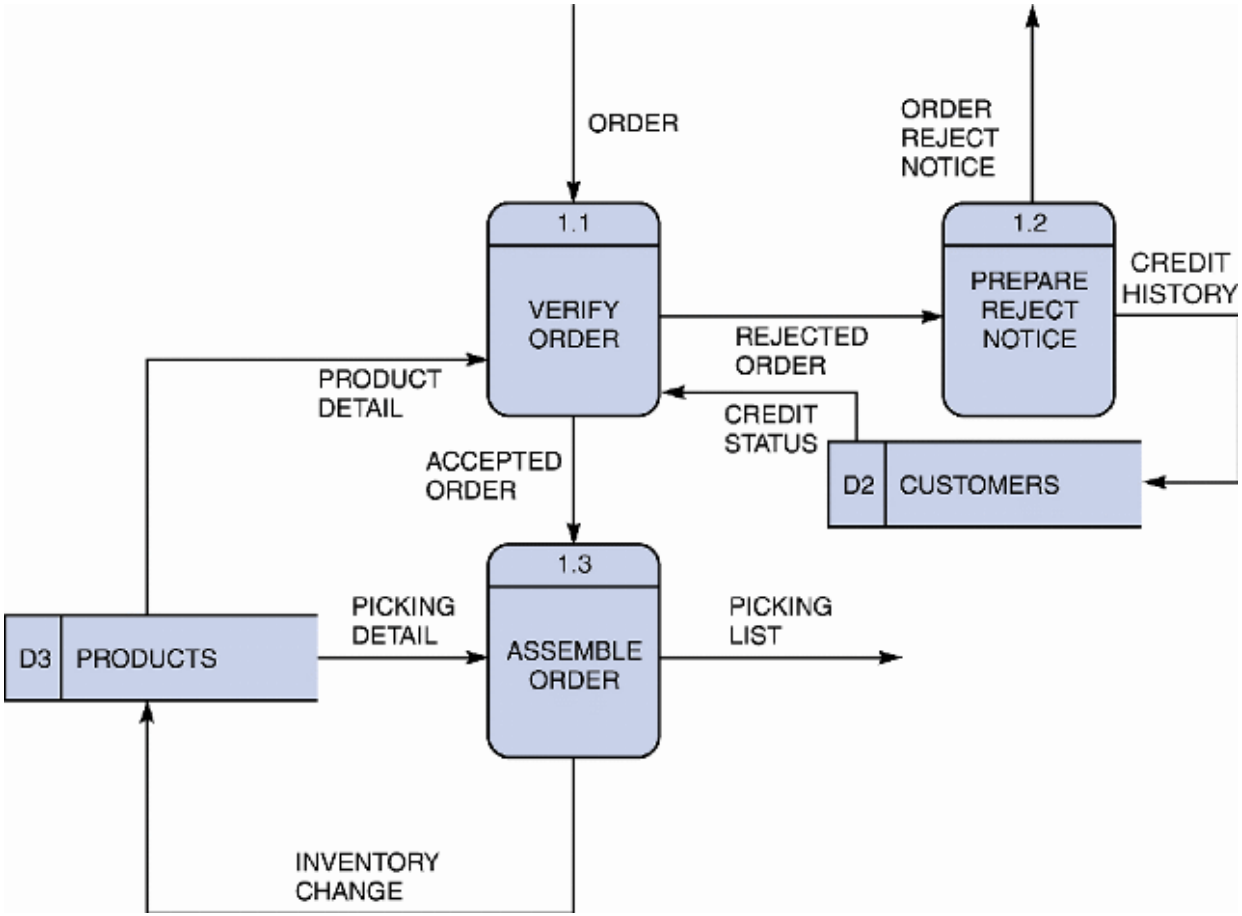
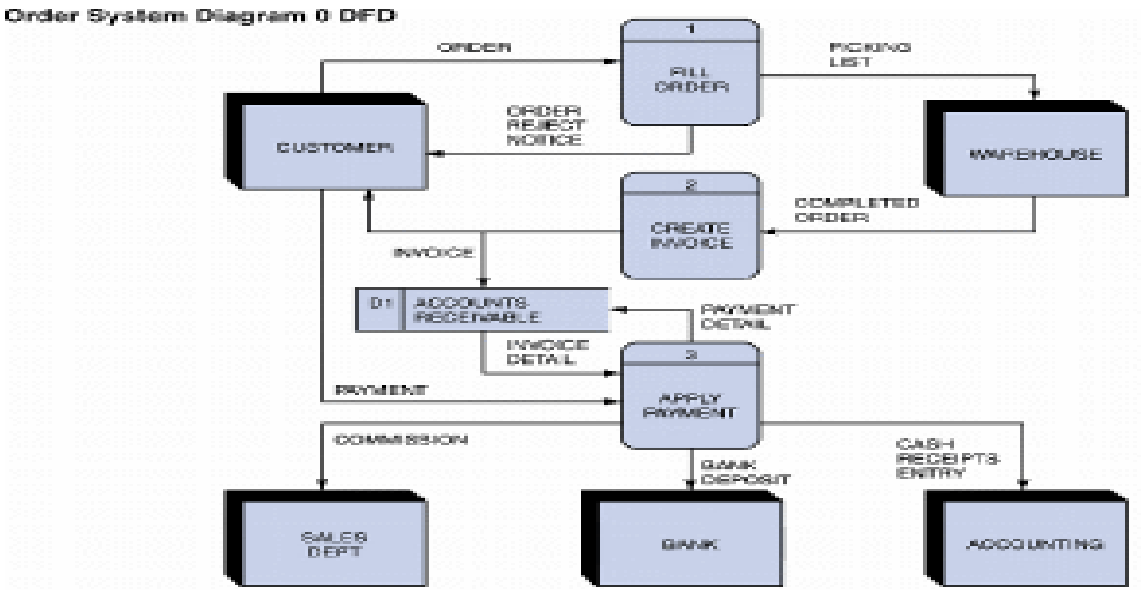
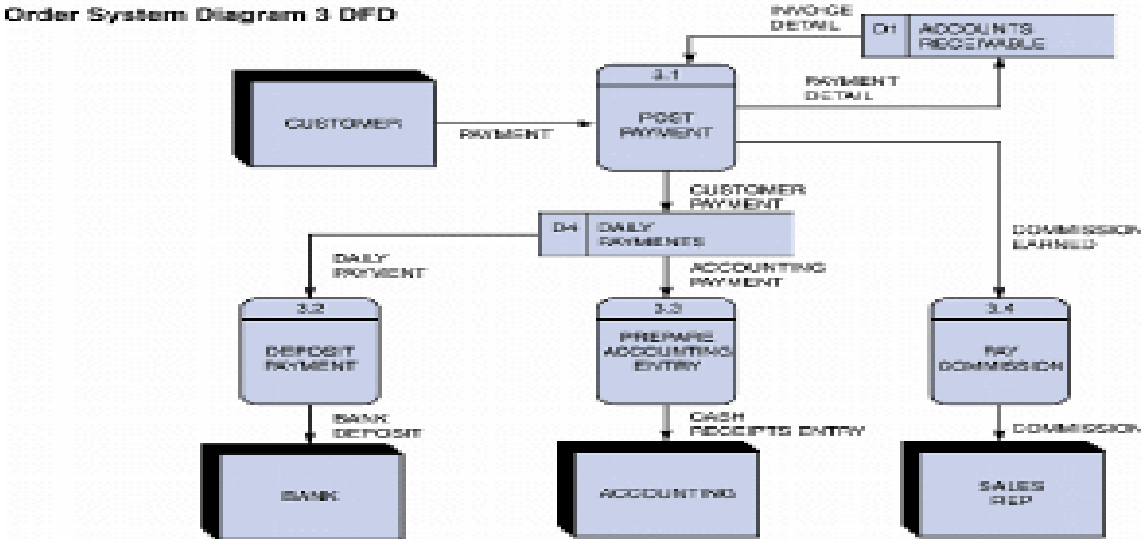


Diagram 1 DFD shows details of the FILL ORDER process without symbols that connect to data flows



Balancing Example





Data Dictionary

McRoyal Enoch Mensah
for NUGS president
hopeful

- A **data dictionary** or **data repository,** is a central storehouse of information about the system's data.
- An analyst uses the data dictionary to collect, document, and organize specific facts about the system including the contents of data flows, data stores, entities and processes.
- A **data element** or **data item** or **field**, is the smallest piece of data that has meaning within an IS.
- **Record** or **data structure** is a meaningful combination of related data elements that is included in a data flow or retained in a data store.

Process Description Tools

- A **process description** documents the details of a functional primitive, and represents a specific set of processing steps and business logic.
- Using a set of process description tools, creates a model that is accurate, complete and concise.
- Typical process description tools include structured English, decision tables, and decision trees.

Modular Design

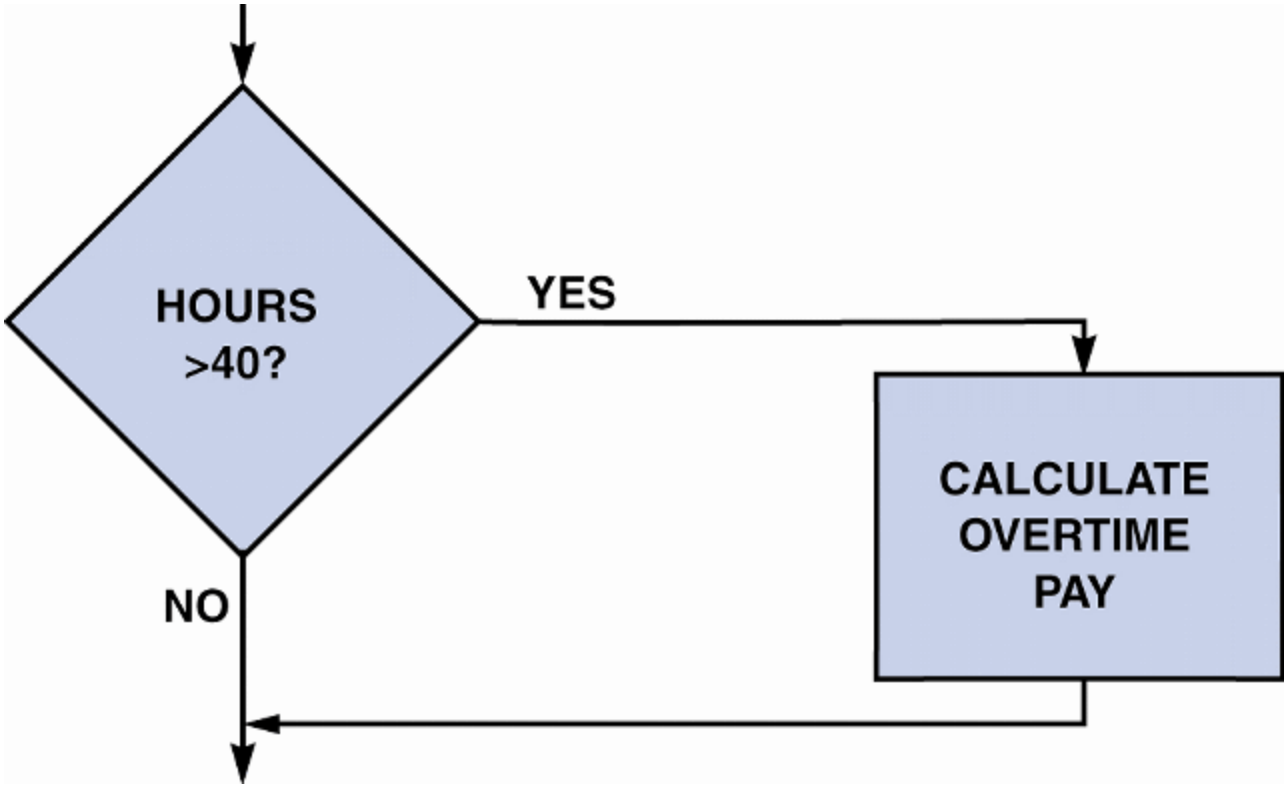
- **Modular design** is based on combinations of three **logical** or **control structures** which serve as building blocks for the process.
- Each logical structure must have a single entry and exit point.
- The three structures are **sequence**, **selection** and **iteration**.

Modular Design - Sequence

- The completion of steps in sequential order, one after another.
- One or more of the steps might represent a sub-process that contains additional logical structures.

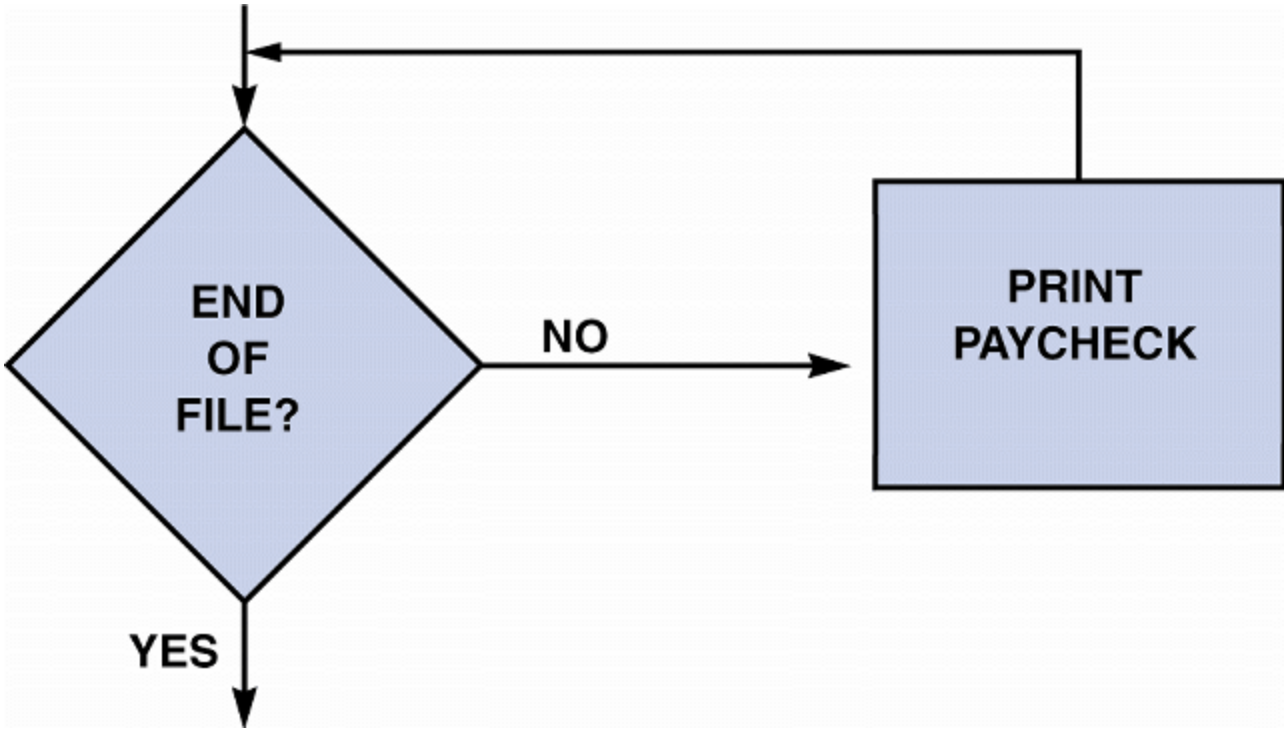
Modular Design - Selection

- The completion of one of two or more process steps based on the results of a test or condition.



Modular Design – Iteration/Looping

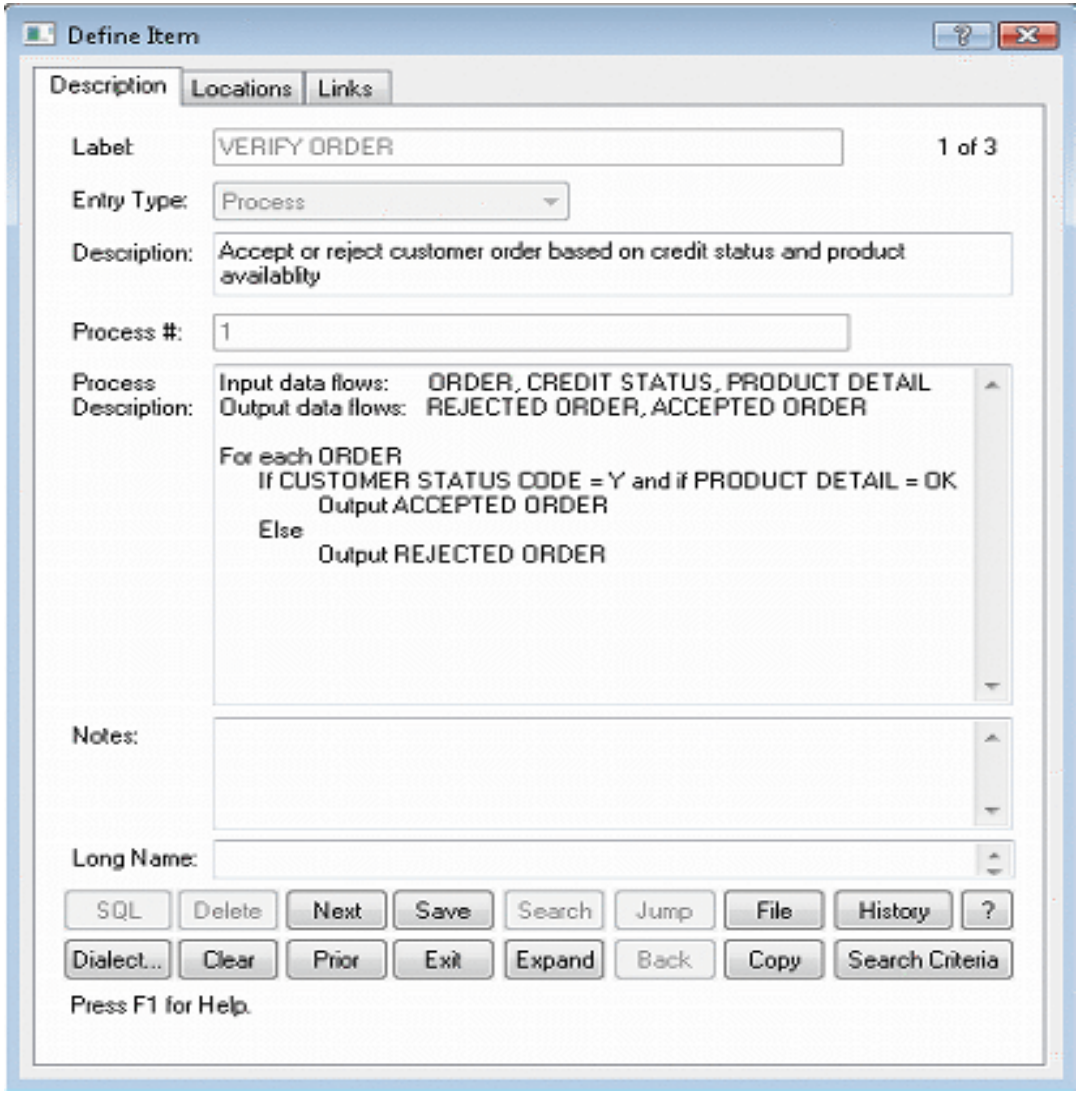
- The completion of a process step that is repeated until a specific condition changes.



Structured English

- It describes logical processes clearly and accurately.
- Must conform to the following rules
  - Use only the three building blocks of sequence, selection, and iteration.
  - Use indentation for readability.
  - Use a limited vocabulary, including standard terms used in the data dictionary and specific words that describe the processing rules.

VERIFY ORDER process description



Decision Tables

- **Decision tables** show a logical structure, with all possible combinations of conditions and resulting actions.
- Analysts often use decision tables in addition to structured English to describe a logical process and ensure that they do not overlook any logical possibility.

Decision Tables (cont'd)

- The number of rules doubles each time you add a condition.
- Can have more than two possible outcomes.
- Often are the best way to describe a complex set of conditions.

Example of Decision Table

VERIFY ORDER Process

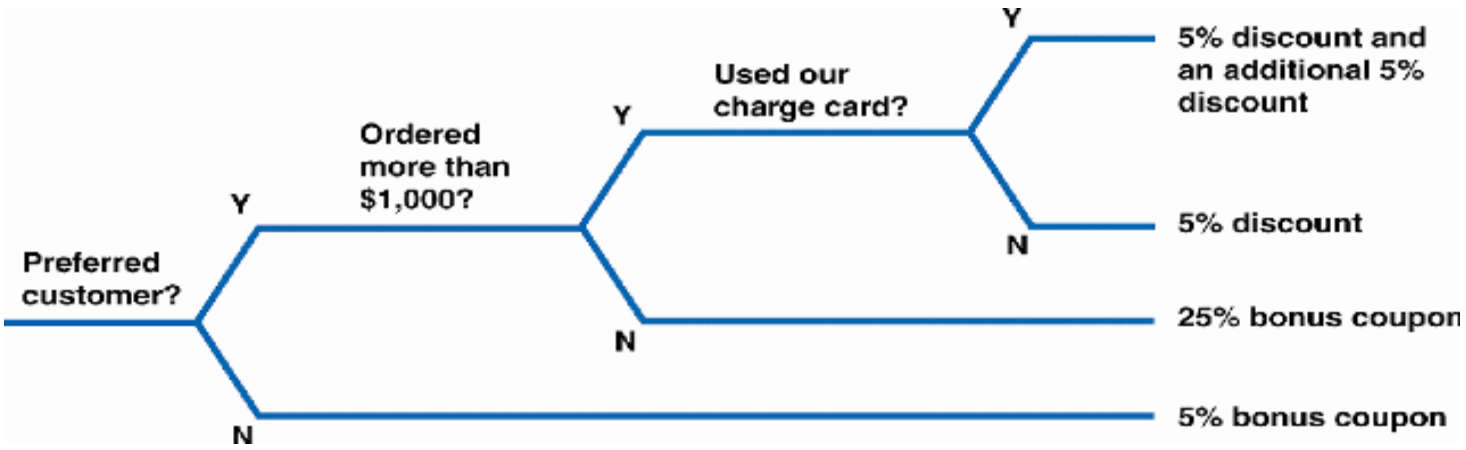
	1	2	3	4
Credit status is OK	Y	Y	N	N
Product is in stock	Y	N	Y	N
Accept order	X			
Reject order		X	X	X

Decision Trees

- A **decision tree** is a graphical representation of the conditions, actions, and rules found in a decision table.
- Decision trees show the logic structure in a horizontal form that resembles a tree with the roots at the left and the branches to the right.
- Whether to use a decision table or tree often is a matter of personal preference.
- Is an effective way to describe a relatively simple process.

Decision Tree Example

- Sales Promotion Policy
  - Preferred customers who order more than $1,000 are entitled to a 5% discount, and an additional 5% discount if they used our charge card.
  - Preferred customers who do not order more than $1,000 receive a $25 bonus coupon.
  - All other customers receive a $5 bonus coupon.



Phase II- Part III:Object Modeling

- **Object-oriented analysis** describes an IS by identifying things as objects.
- An object represents a real person, place, event or transaction.
- Object-oriented analysis is a popular approach that sees a system from the viewpoint of the objects themselves as they function and interact.

Object-Oriented Terms and Concepts

McRoyal Enoch Mensah
for NUGS president
hopeful

- **Unified Modeling Language (UML)**
- **Attributes:** Characteristics that describe an object.
- **Methods**: Tasks or functions that an object performs when it receives a **message** or command.
- **Message**
- **Class**: A group of similar objects.
- **Instance:** A specific member of a class.

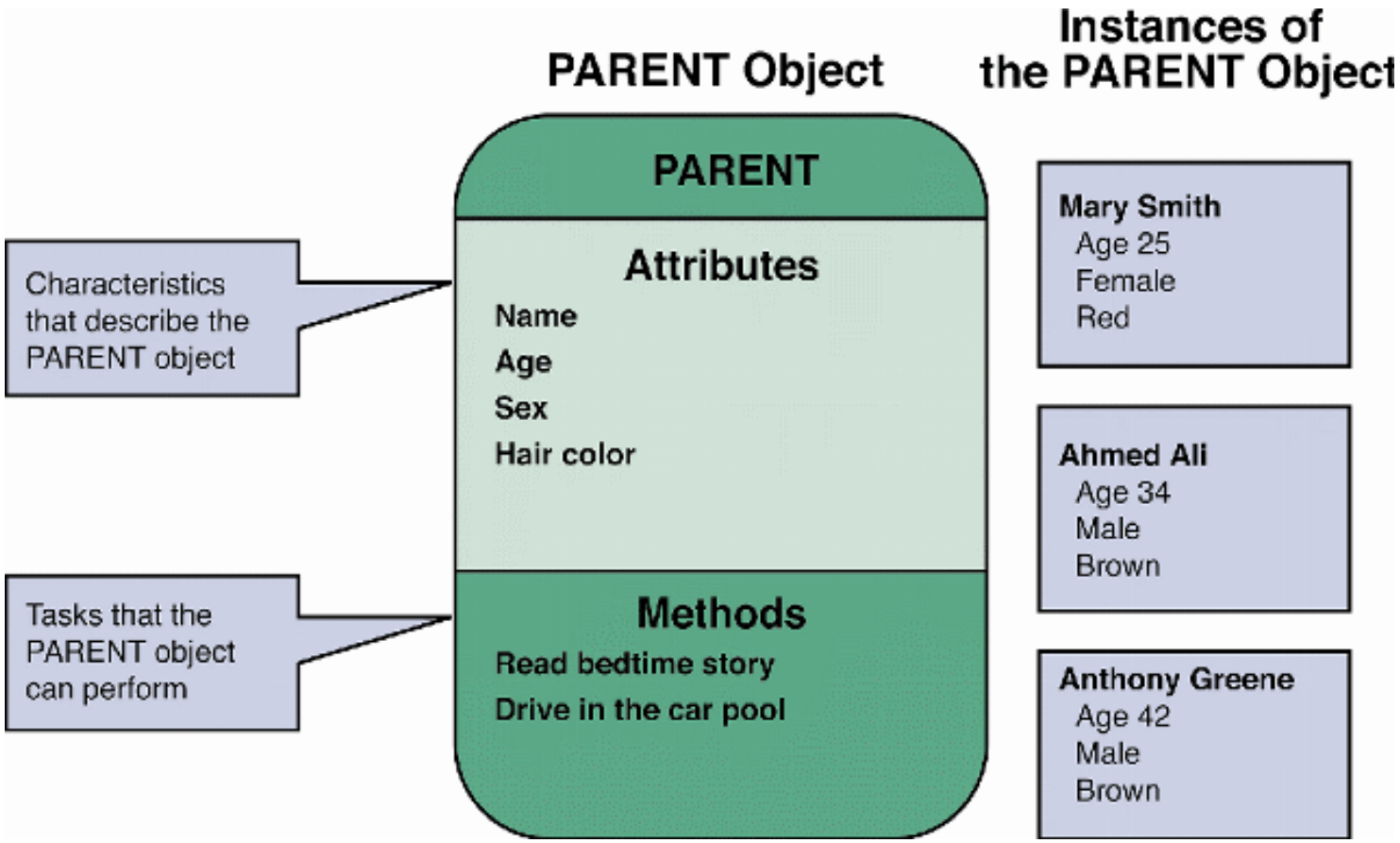
Objects

McRoyal Enoch Mensah
for NUGS president
hopeful

- The UML represents an object as a rectangle with the object name at top, followed by the object's attributes and methods.

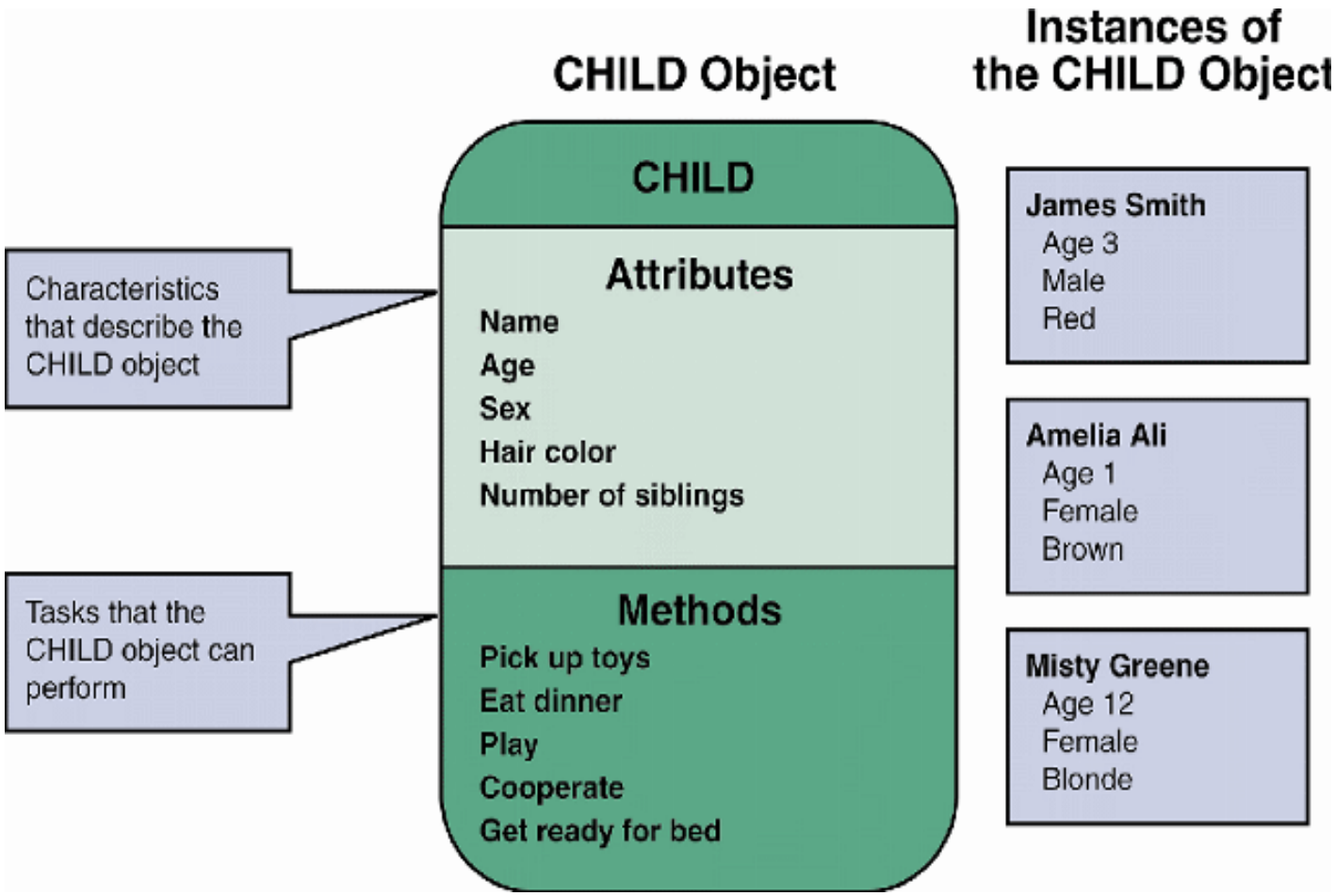
Object examples

McRoyal English Mensah
for NUGS president
hopeful



Object examples

McRoyal English Mensah
for NUGS president
hopeful



Attributes

McRoyal Enoch Mensah
for NUGS president
hopeful

- Systems analysts define an object's attributes during the systems design process.
- In an O-O system objects inherit or acquire certain attributes from other objects.
- The state of an object is an adjective that describes the object's current status.

Methods

McRoyal Enoch Mensah
for NUGS president
hopeful

Method:

MORE FRIES

Steps:

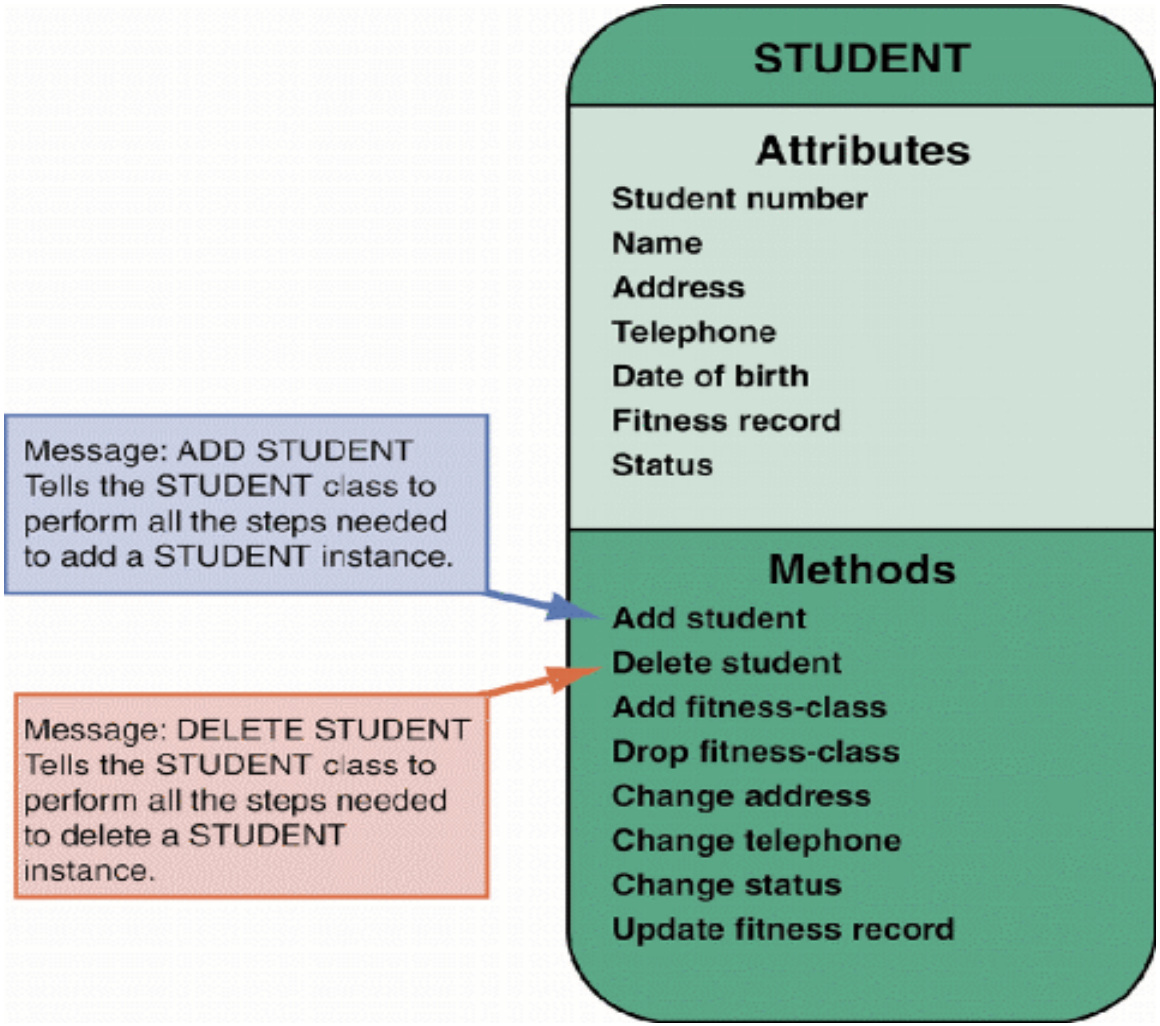
1. Heat oil
2. Fill fry basket with frozen potato strips
3. Lower basket into hot oil
4. Check for readiness
5. When ready raise basket and let drain
6. Pour fries into warming tray
7. Add salt

Message

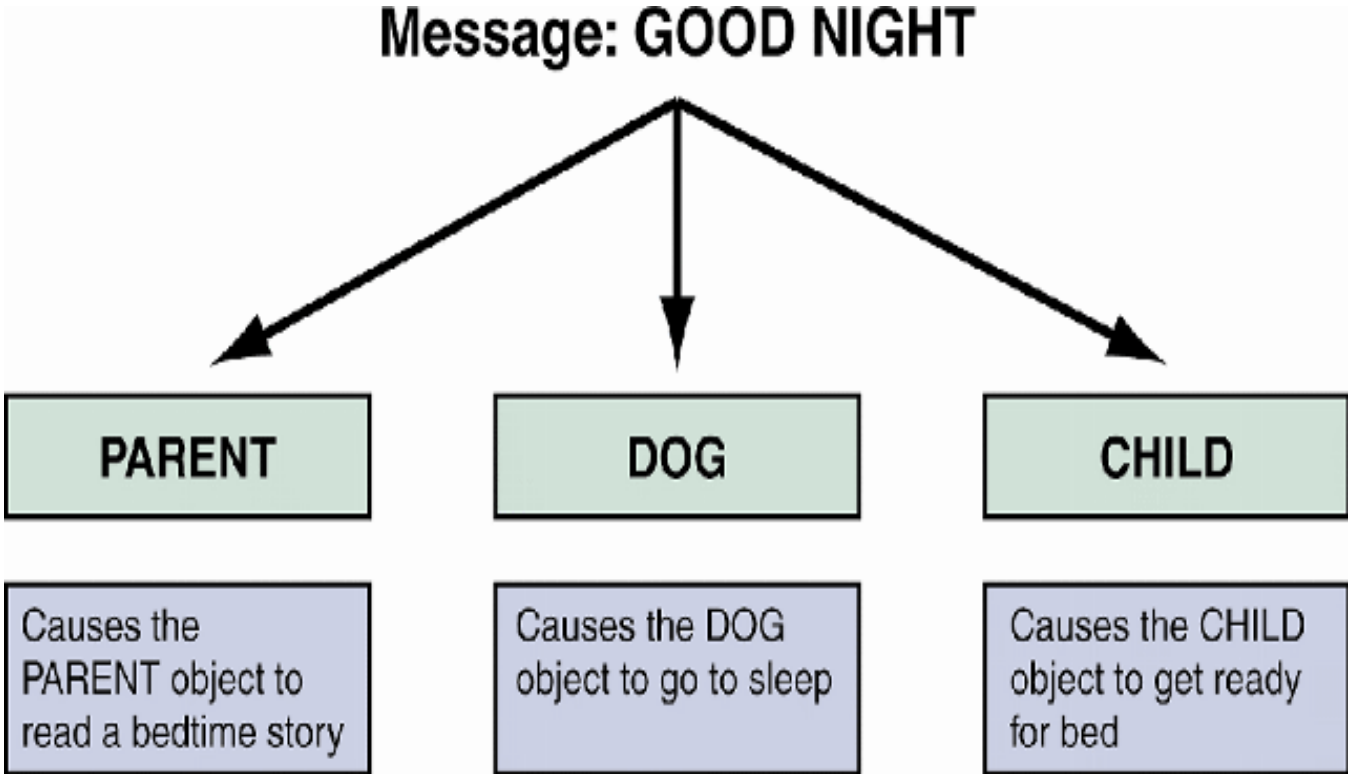
McRoyal Enoch Mensah
for NUGS president
hopeful

- A message is a command that tells an object to perform a certain method.
- The same message to two different objects can produce different results.
- The concept that a message gives two meanings to different objects is called **polymorphism**.
- An object can be viewed as a **black box** because a message to the object triggers changes within the object without specifying how the changes must be carried out.
- **Encapsulation** means that all data and methods are self-contained.

Message example



Polymorphism example

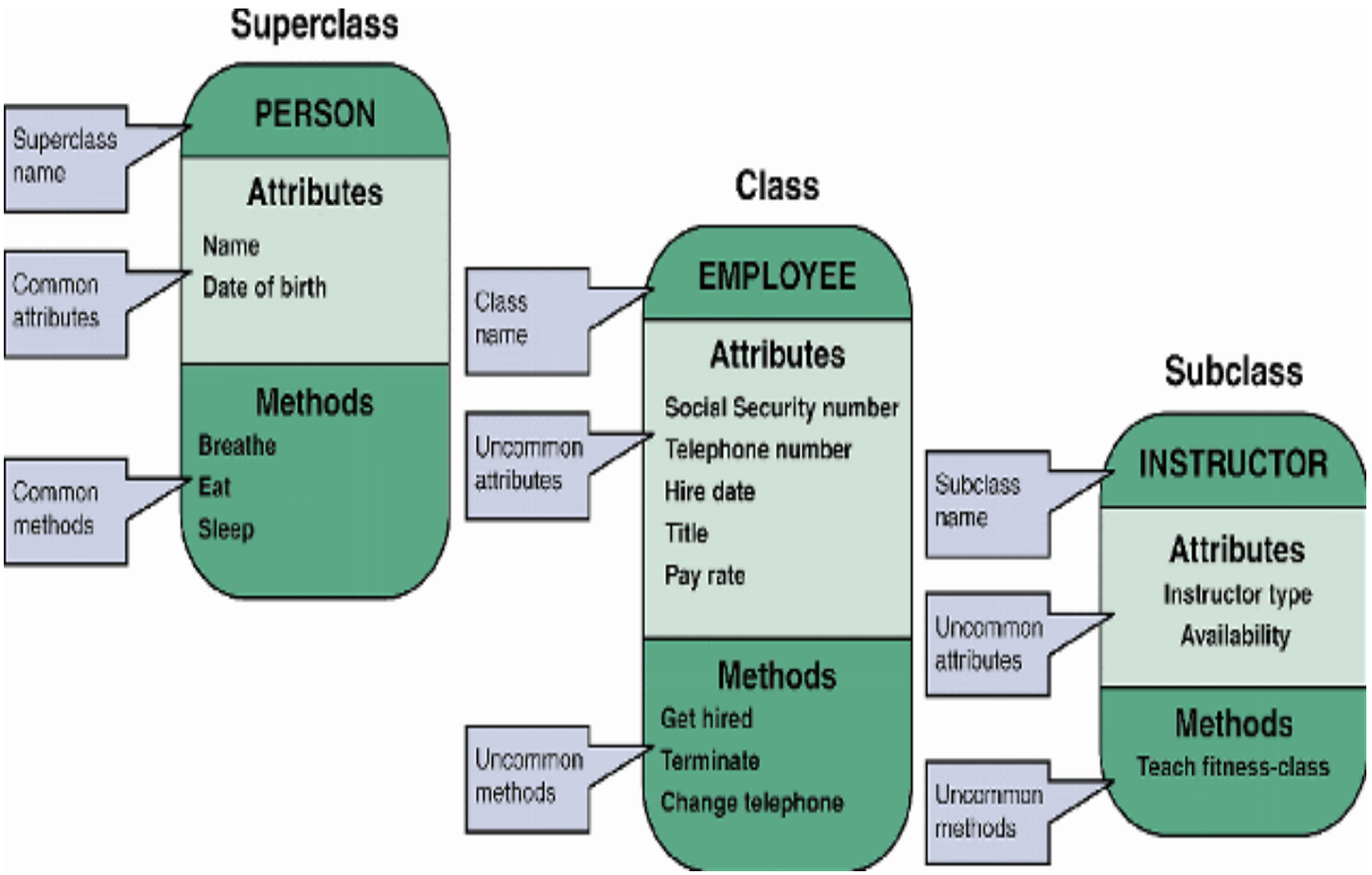


Classes

McRoyal Enoch Mensah
for NUGS president
hopeful

- An object belong to a group or category called a class.
- Objects within a class can be grouped into **subclasses**, which are more specific categories within a class.
- A class can belong to a more general category called a superclass.

Example of a class

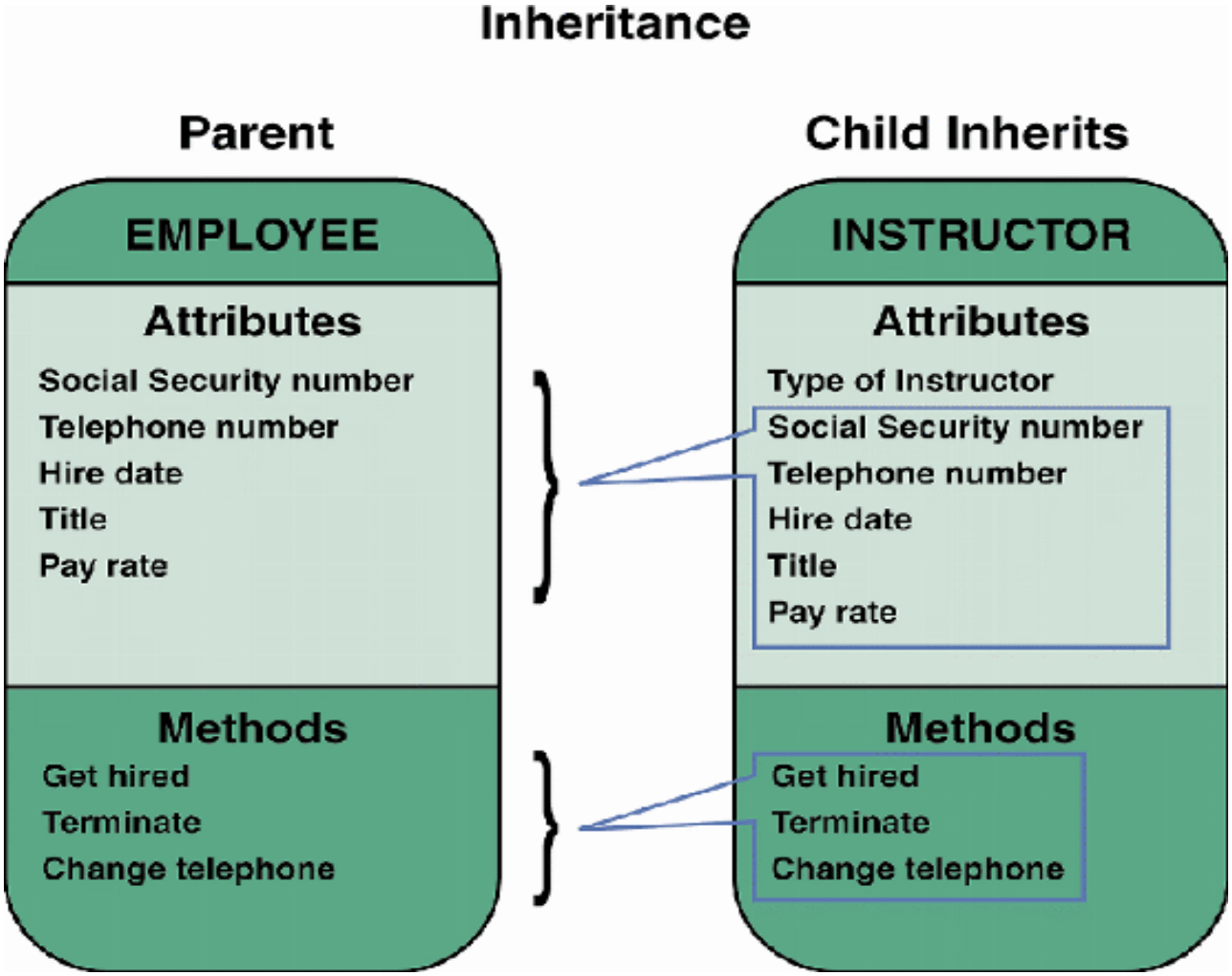


Relationship Among Objects and Classes

- **Relationships** enables objects to communicate and interact as they perform business functions and transactions required by the system.
- Relationships describe what objects need to know about each other, how objects respond to changes in other objects, and the effects of membership in classes, superclasses, and subclasses.
- **Inheritance** enables an object, called a **child**, to derive one or more of its attributes from another object, called a **parent.**

Inheritance

McRoyal Enoch Mensah
for NUGS president
hopeful

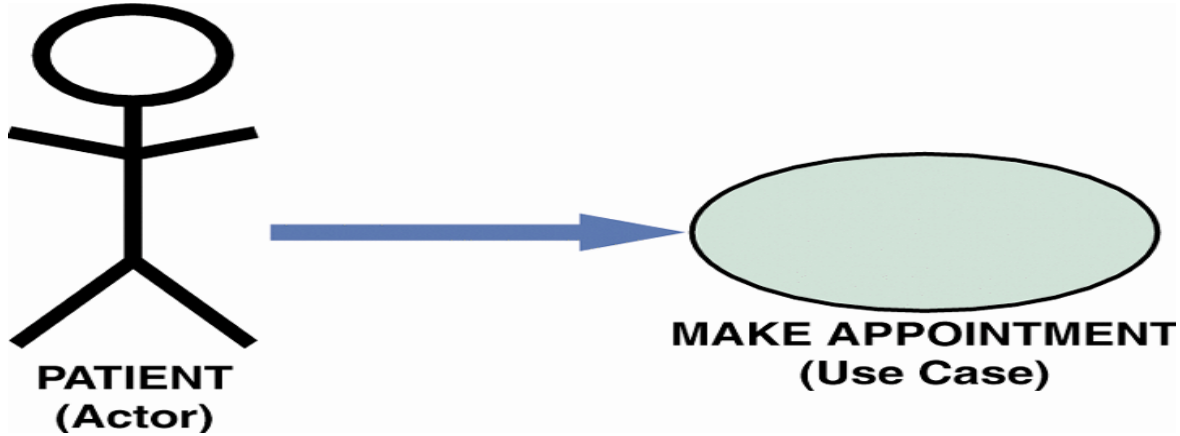


Object Modeling with UML

- As structured analysis uses DFDs to model data and processes, systems analysts use UML to describe object-oriented systems.

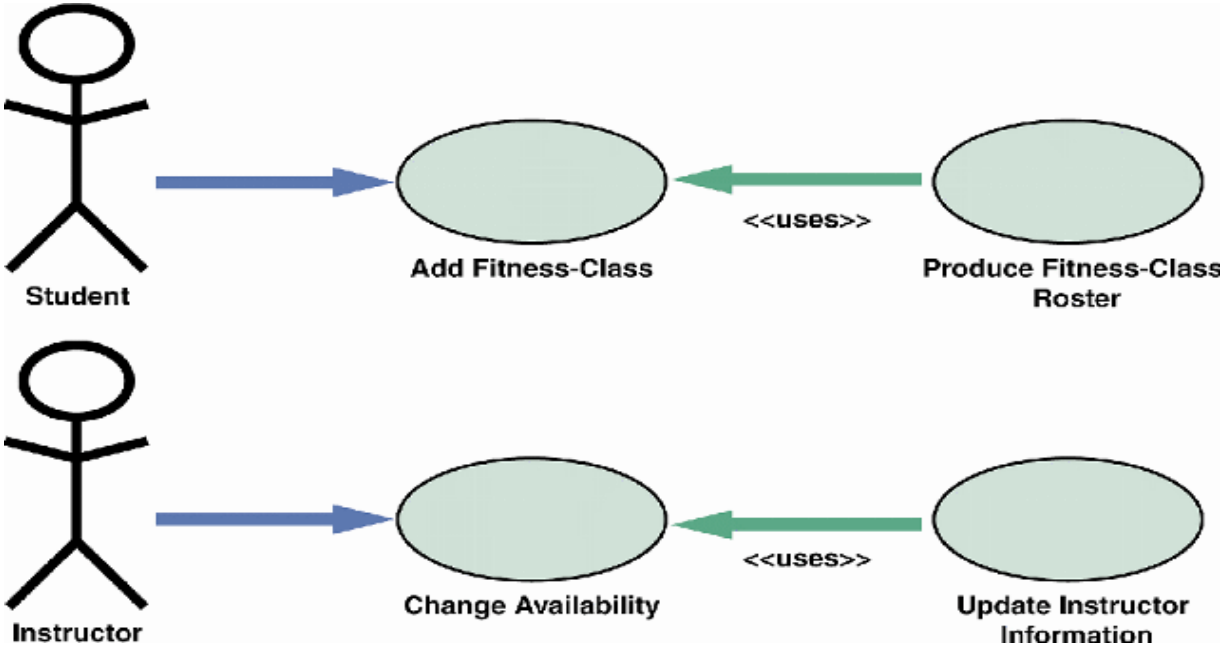
Use Case Modeling

- A **use case** represents the steps in a specific business function or process. An external entity, called an actor, initiates a use case by requesting the system to perform a function or process.



Use Case Modeling (cont'd)

- Use cases can also interact with other use cases. When the outcome of the use case is incorporated by another use case, then the second use case *uses* the first case.

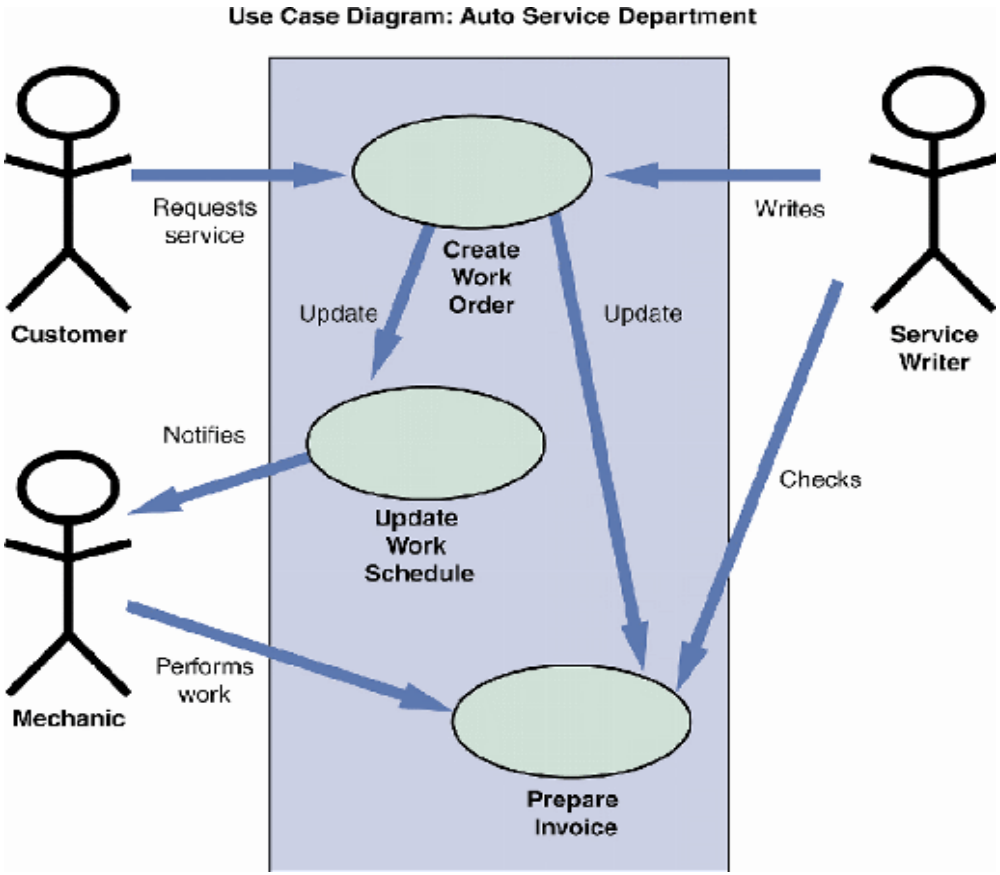


Use Case Modeling (cont'd)

- For each use case, develop a **use case description** in the form of a table.
- A use case description documents the name of the use case, the actor, a description of the use case, a step-by-step list of the tasks and actions required for successful completion, a description of alternative courses of action, preconditions, postconditions and assumptions.

Use Case Diagrams

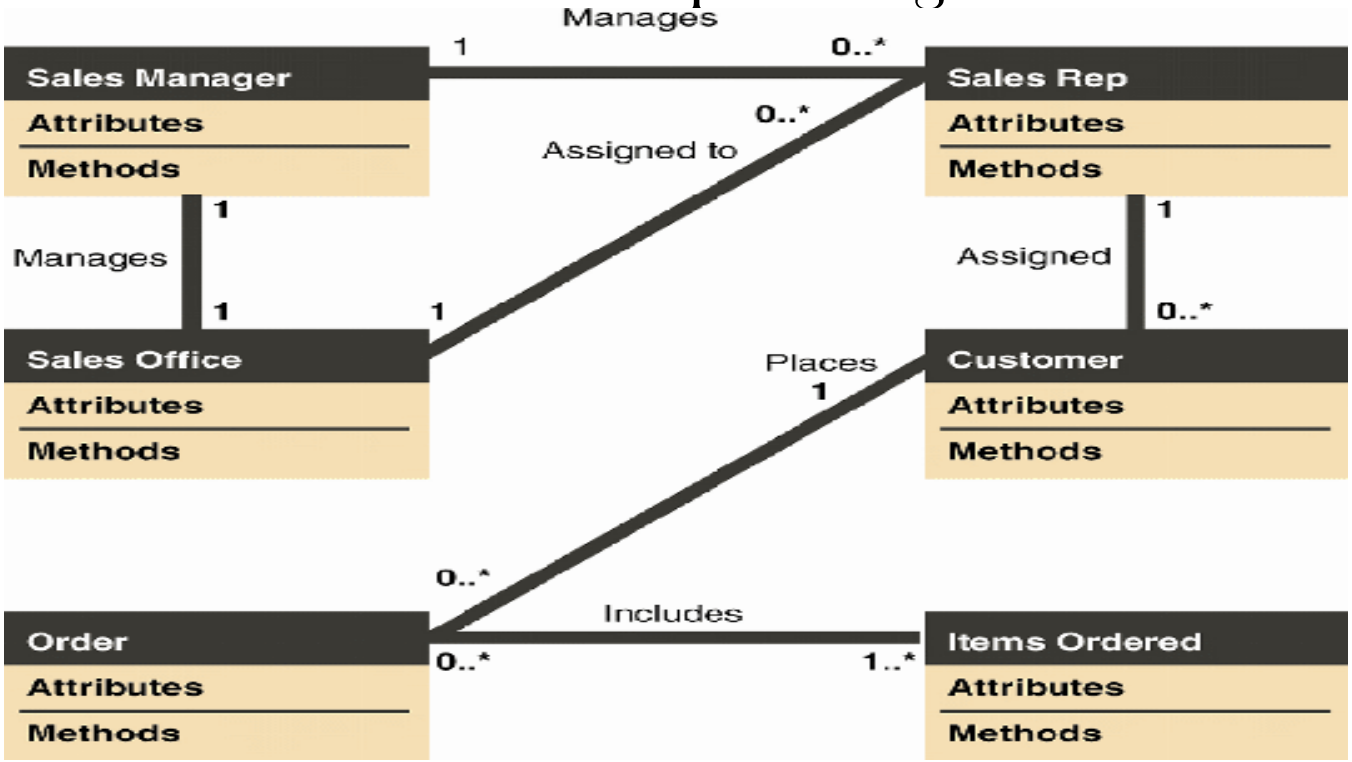
- A **use case diagram** is a visual summary of several related use cases within a system or subsystem.



Class Diagrams

McRoyal Enoch Mensah
for NUGS president
hopeful

- A class diagram represents a detailed view of a single use case.
- It shows the classes that participate in the use case, and documents the relationship among the classes.



Cardinality

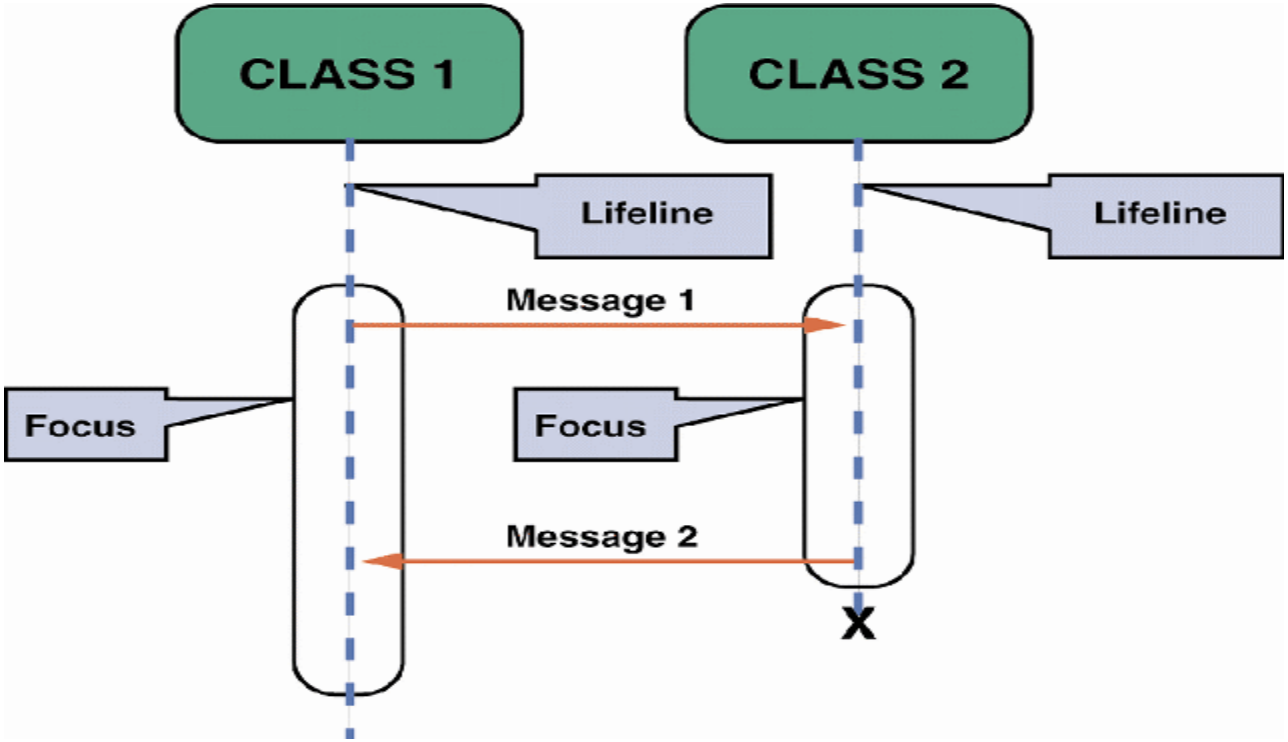
McRoyal Enoch Mensah
for NUGS president
hopeful

- Cardinality describes how instances of one class relate to instances of another class.

UML Notation	Nature of the Relationship	Example	Description
0..*	Zero or many	Employee — Payroll Deduction (1 — 0..*)	An employee can have no payroll deductions or many deductions.
0..1	Zero or one	Employee — Spouse (1 — 0..1)	An employee can have no spouse or one spouse.
1	One and only one	Office Manager — Sales Office (1 — 1)	An office manager manages one and only one office.
1..*	One or many	Order — Item Ordered (1 — 1..*)	One order can include one or many items ordered.

Sequence Diagrams

- A sequence diagram is a dynamic model of a use case.
- It shows the interaction among classes during a specified time period.

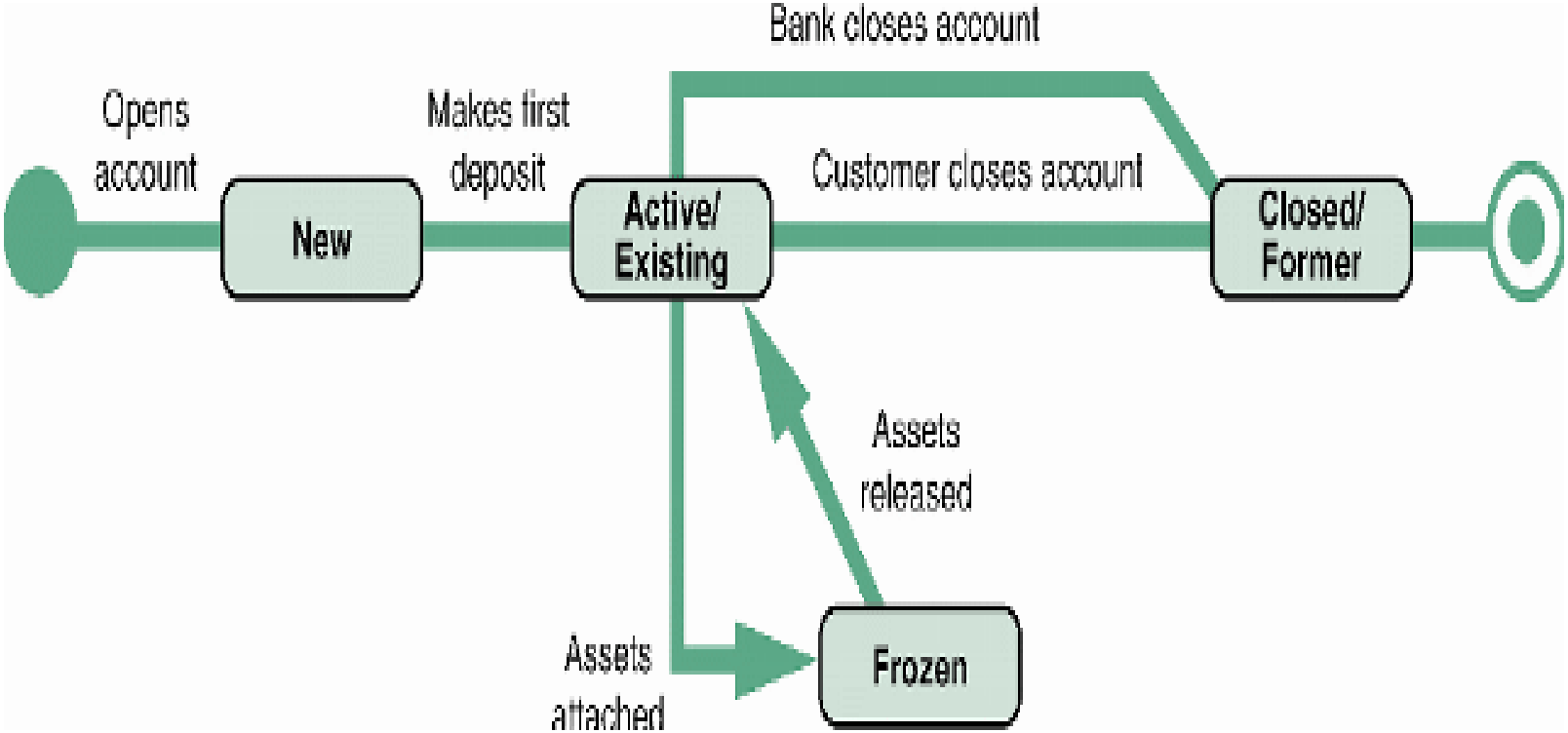


Sequence Diagrams (cont'd)

- CLASSES – A **class** is identified by a rectangle with the name inside.
- LIFELINES – A **lifeline** represents the time during which an object is able to interact with the other objects in the use case. An X marks the end of the lifeline.
- MESSAGES – A **message** is identified by a line showing direction that runs between two objects.
- FOCUSES – A **focus** indicates when an object sends or receives a message.

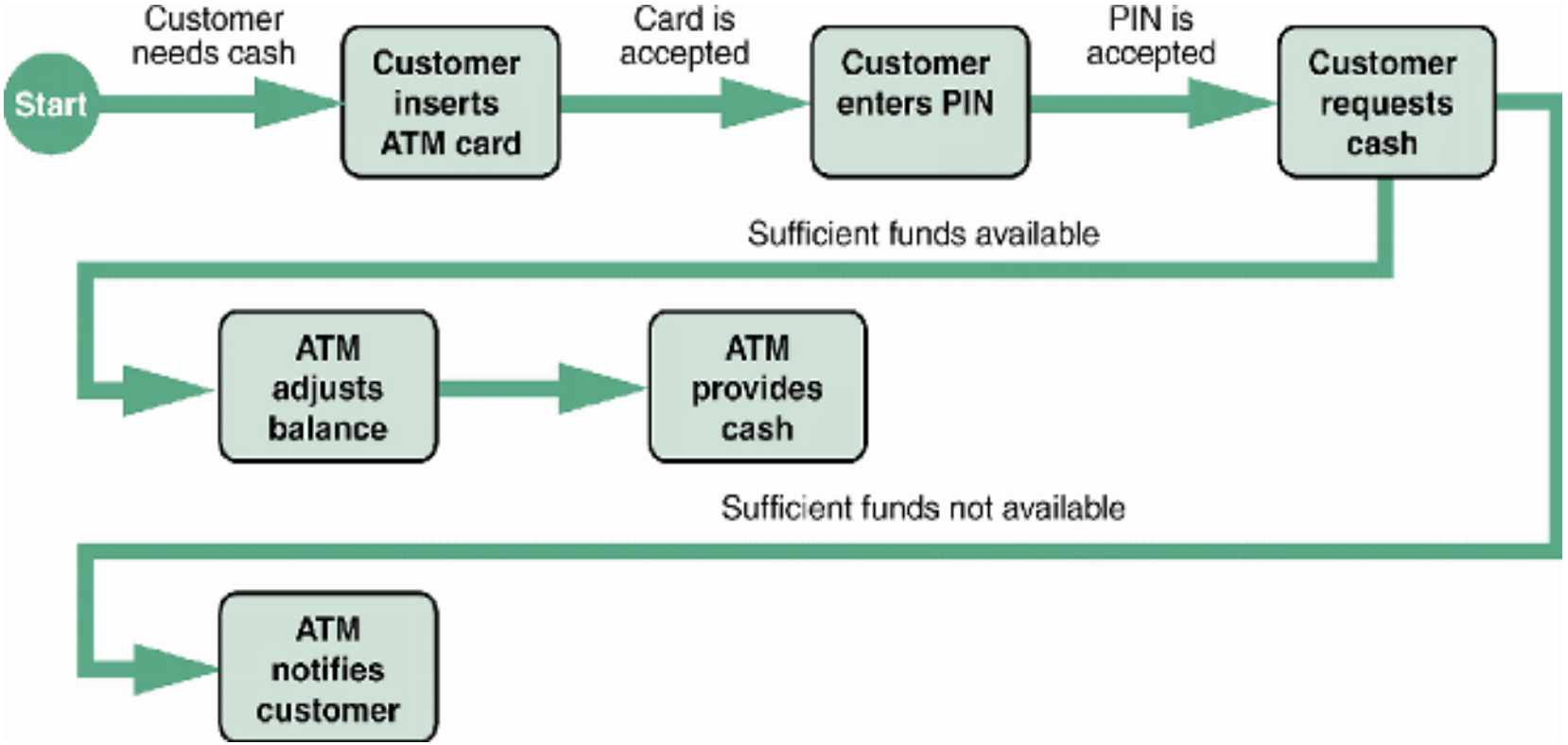
State Transition Diagrams

- A **state transition diagram** shows how an object changes from one state to another, depending on events that affect the object.



Activity Diagrams

- An activity diagram resembles a horizontal flowchart that shows the actions and events as the occur.



Phase II - Part IV – Development Strategies

- Selecting the best development path is an important decision that requires companies to consider.
  - Outsourcing
  - In-house software development options

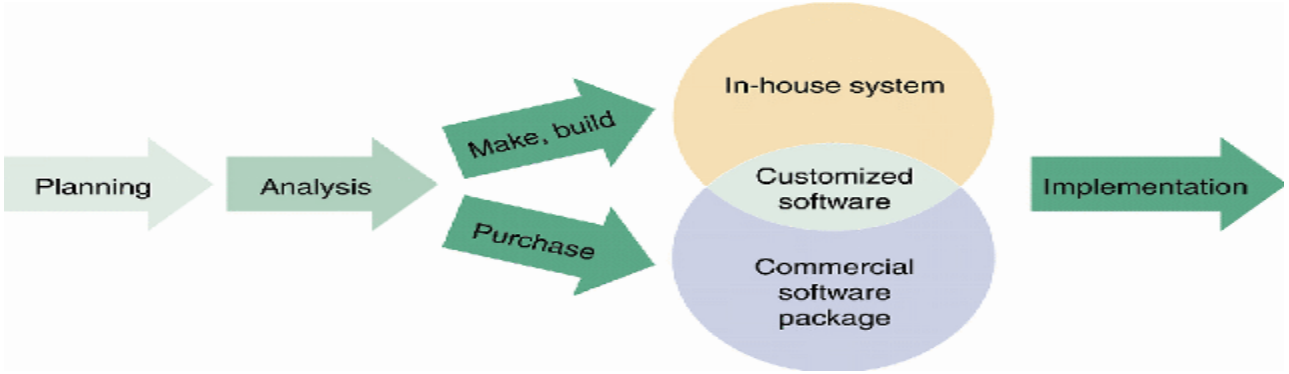
Outsourcing

McRoyal Enoch Mensah
for NUGS president
hopeful

- **Outsourcing** is the transfer of information systems development, operation or maintenance to an outside firm that provides these services for a fee, on a temporary or long-term basis.
- It can refer to relatively minor programming tasks, the rental of software from a service provider, the outsourcing of a basic business process (often called business process outsourcing, or BPO), or the handling of a company's entire IT function.

In-House Software Development Options

- A company can choose to develop its own systems, or purchase, possibly customize and implement a software package.
- The choice between developing versus purchasing software is called a **make or buy,** or **build or buy** decision.



REASONS FOR IN-HOUSE DEVELOPMENT	REASONS FOR PURCHASING A SOFTWARE PACKAGE
Satisfy unique business requirements	Lower costs
Minimize changes in business procedures and policies	Requires less time to implement
Meet constraints of existing systems	Proven reliability and performance benchmarks
Meet constraints of existing technology	Requires less technical development staff
Develop internal resources and capabilities	Future upgrades provided by the vendor
Satisfy unique security requirements	Obtain input from other companies