International Islamic University Chittagong (IIUC)

Department of Computer and Communication Engineering (CCE)

System Analysis, Design and Development Sessional

Lab Manual

Prepared by

Muhammad Kamal Hossen (MKH_CUET)

Adjunct Faculty, Department of CCE

International Islamic University Chittagong

Table of Contents

PART I: USER MANUAL

1. COURSE DESCRIPTION

Course Code: **CCE-3508**

Course Title: **System Analysis, Design and Development Sessional**

Credit Hour: 1.0 **Contact Hour:** 2 per week

1.1 COURSE CONTENTS (SYLLABUS)

Sessional based on the following topics:

i) Concepts of system and its environment: Information, Types of information, Quality of information, System, Types of systems, Components of system, Source of information.

ii) Information gathering: Strategy, Information searching methods, Interviewing technique, System development methodologies and life cycle.

iii) Feasibility study & Cost/Benefit analysis: Feasibility considerations, steps in feasibility analysis, feasibility report, Cost and Benefit categories, procedure for cost and benefit determination, classification of cost and benefit, cost and benefit evaluation methods.

iv) Tools of analysis and design: Data Flow Diagram (DFD), DFD symbols, Constructing DFD; Data Dictionary; Decision Tree, Structured English, Decision Tables.

v) System Design and Construction: The process of design, System design phases, Design methodologies; Structured design; Form-Driven methodology; Input design, Output design, File and database design.

vi) Testing and Quality Assurance: Testing, Types of system tests; White-Box testing; Black-box testing; Quality factors specifications.

vii) Implementation and Maintenance: Types of implementation, Documenting the system, Training and supporting users, Factor models of implementation success; The process of maintaining information system, Types of maintenance, Cost of maintenance, Reducing maintenance cost.

viii) Hardware/Software selection, control and security: Phases in selection, Criteria for software selection, Hardware selection, Financial considerations in selection; Security definitions, Threats to system security, Control measures, system failures and recovery.

2. Objective(s) of this Course:

At the end of the course students should know how writing programs with tough assurance targets, in large teams, or both, differs from the programming exercises they have engaged in so far. They should appreciate the waterfall, spiral and evolutionary models of software development and be able to explain which kinds of software project might profitably use them. They should appreciate the value of other tools and the difference between incidental and intrinsic complexity. They should understand the software development life cycle and its basic economics. They should be prepared for the organizational aspects of their group project.

3. Course outcomes:

ILOs / COs	Description
ILO / CO: 1	Acquire the generic software development skill through various stages of software life cycle as well as able to ensure the quality of software through software development with various protocol-based environment.
ILO / CO: 2	Able to generate test cases for software testing.
ILO / CO: 3	Able to handle software development models through rational method.

4. Assessment Methods:

Assessment Types	Marks
Attendance	**10**
Lab Performance	**20**
Lab Reports	**20**
Final Examination	**50**

5. Grading Policy:

Numerical Grade Marks (%)	Letter Grade (LG)	Grade Point (GP/unit)	Remarks/Status
80-100	A+ (A plus)	4.00	Excellent
75 to less than 80	A (A regular)	3.75	Very good
70 to less than 75	A- (A minus)	3.50	
65 to less than 70	B+ (B plus)	3.25	Good
60 to less than 65	B (B regular)	3.00	
55 to less than 60	B- (B minus)	2.75	Satisfactory
50 to less than 55	C+ (C plus)	2.50	
45 to less than 50	C (C regular)	2.25	Not Satisfactory
40 to less than 45	D (D regular)	2.00	
less than 40	F	0.00	Fail

6. LAB INSTRUCTIONS:

- Each lab will begin with a brief overview of the topic and the learning objectives.
- Students will be provided with clear instructions and specific tasks to complete for each lab session.
- Labs can involve individual work, group activities, or a combination of both, depending on the specific exercise.
- All lab assignments and deliverables (e.g., reports, diagrams) must be completed and submitted according to the instructor's instructions.

PART II: LAB SESSIONS

Lab #1: Problem Analysis and Requirement Specifications

Section A: Tutorial - Demonstrated by Instructor

(30-60 minutes prior to lab Experiments)

Objectives:

- Understand the basic concepts of system analysis and design.
- Familiarize students with the software tools used in system analysis and design.
- Learn techniques for gathering system requirements.
- Understand the importance of user involvement in requirements gathering.

Activities:

- Introduction to the course syllabus and objectives.
- Overview of system development life cycle (SDLC).
- Familiarization with software tools like Microsoft Visio, Lucidchart, or other modeling tools.
- Brief discussion on case studies and real-world examples of system analysis and design projects.
- Introduction to requirement elicitation techniques (interviews, questionnaires, observation, etc.).
- Practical exercises on conducting interviews and documenting requirements.
- Group discussions on the challenges and strategies for gathering accurate requirements.

- Assignment: Conduct a requirement gathering session for a selected system.

Contents:

- Introduction
- Problem Statement
- Purpose
- Scope
- Overview of the Organization/Enterprise/System
- Management Structure
- Types of Information
- Information Gathering
- Requirements Specifications
- Conclusion

A Sample of Lab Report on Assignment-1

Experiment Name: POS Billing System for Restaurant Management

Introduction

The purpose of this report is to provide a POS billing system for a Restaurant Management. The proposed system will aim to improve the efficiency and accuracy of the ordering and billing process, as well as provide valuable data and insights for the restaurant management.

Problem Statement

The current system for order-taking and billing at the restaurant is outdated and presents several challenges that negatively impact both staff and customers. These challenges include:

1. Inefficient order-taking process: The current system is cumbersome and time-consuming for staff, requiring multiple steps to process an order. This leads to long wait times for customers and increased stress for staff.
2. Inaccurate billing: The current system is prone to errors, resulting in inaccuracies in billing. This can lead to customer dissatisfaction and increased workload for staff.
3. Limited data and insights: The current system does not provide useful data and insights for the restaurant management, such as customer data, and employee performance data.
4. Inadequate security: The current system is not secure and customer and employee data is not protected and kept confidential.
5. Inefficient payment processing: The current system is not able to handle multiple payment methods which leads to long wait times for customers and increased stress for staff.

Purpose

The purpose of the proposed POS billing system for restaurant management is to improve the order-taking and billing process for the restaurant by providing an efficient, accurate, and user-friendly system. The system will include features such as menu item setup, order processing, payment processing, reporting, inventory management, customer management, data analysis and insights for the Restaurant Management.

Overview

The system will include the following features:

1. Menu item setup: Allows staff to easily add, edit, and remove menu items, as well as set prices and manage inventory levels.

2. Order processing: Allows staff to quickly and accurately process orders, with the ability to split bills, apply discounts, and print receipts.
3. Payment processing: Enables the system to handle single payment methods only cash.
4. Reporting: Provides valuable data and insights for the restaurant management, including sales reports, inventory reports, and customer data.
5. Inventory management: Allows staff to track the stock levels of items and generate reports to help management in making informed decisions on inventory.
6. Customer management: Allows staff to manage customer information, including contact details, order history, and loyalty rewards.
7. Data Analysis and insights: Provides data analysis and insights for the restaurant management, including sales reports, inventory reports, customer data, sales analysis.

Management Structure

The management structure for a POS billing system at restaurant management would typically include the following roles:

1. System Administrator: The person responsible for managing and maintaining the system, including monitoring system performance, troubleshooting any issues, performing regular backups and control users.
2. Restaurant Manager: The person responsible for overseeing the day-to-day operations of the restaurant, including managing staff, ordering supplies, and ensuring the restaurant is running smoothly.
3. Staff: The employees who will be using the system to process orders, handle payments, and manage inventory.
4. System Developer: The person responsible for providing fixing any bug, technical support and troubleshooting any issues with this system.
5. Stakeholders: These are the people who invest in the business and have a stake in the success of the restaurant, they will be provided with regular reports and insights from the system.

Types of Information

The types of information that a POS billing system for restaurant management would typically collect and store include:

1. Customer information: This includes contact details, order history, and loyalty rewards.
2. Order information: This includes the items ordered, the date and time of the order, the customer's name, total cost and VAT of the order.

3. Payment information: This includes the type of payment, amount paid, and any discounts or coupons applied.
4. Inventory information: This includes the stock levels of items, reorder points, and the cost of goods.
5. Sales information: This includes the total sales for a given period, the number of customers served, and the average order value.
6. Reporting and insights: This includes data and insights such as sales reports, inventory reports, and customer data, which can be used to inform business decisions.

Information Gathering

There are several methods that can be used for information gathering:

1. Sales Information: In this way we can collect information about our total sales, sales volume per day and sales data for any date range. Which will help us grow our business.
2. Review of existing documentation: Reviewing existing documentation such as user manuals, and other system-related documents can provide valuable information about the current system's capabilities and limitations.
3. Reviewing customer feedback: Reviewing customer feedback from various sources such as online reviews, social media comments etc. can provide valuable information about their experience with the current system and identify areas that need improvement.

Requirement Specifications

Based on the information gathered through the methods outlined in the previous section, the following requirements have been identified for the proposed POS billing system for restaurant management:

1. Hardware components: The system should include a touchscreen terminal, cash drawer, barcode scanner, and receipt printer. These components will be used for order-taking, payment processing, and receipt printing.
2. Software features: The system should include menu item setup, order processing, payment processing, and reporting capabilities. This will enable staff to easily navigate the system, process orders and payments, and generate reports for management.
3. Inventory management: The system should have an inventory management feature which enable the restaurant to track the stock levels of items and generate reports to help management in making informed decisions on inventory.
4. Data and insights: The system should provide valuable data and insights for the restaurant management, such as sales reports, inventory reports, customer data, and employee performance data.

5. Security: The system should be secure and ensure that customer and employee data is protected and kept confidential.
6. User-friendly: The system should be user-friendly and easy to navigate for staff and customers.
7. Cost and Implementation: A budget and implementation plan will need to be developed for the system, including timelines for implementation, testing, and training.

Conclusion

A POS billing system for restaurant management is a vital tool for any restaurant looking to improve their operations and customer service. It can help to streamline the ordering and payment process, improve inventory management, and provide valuable data and insights to the restaurant management. The system should be easy to use, reliable, and should provide real-time data and insights, so that the restaurant management can quickly respond to changing conditions. The system should also provide secure storage for customer and employee data to protect their privacy. By implementing a POS billing system for restaurant management, a restaurant can improve its operations, enhance customer service and ultimately increase their revenue.

Section B:

Lab Assignment - Demonstrated by students

Assignment-1: Write a report on **problem definition**, **information gathering**, and **requirement specifications** of your selected project.

Lab #2: Feasibility Analysis of the Proposed System

Section A: Tutorial - Demonstrated by Instructor

(30-60 minutes prior to lab Experiments)

Objectives:

The primary objective of this lab session is to introduce students to the concept of feasibility analysis in system development. Students will learn to evaluate the feasibility of proposed systems based on various factors including technical, economic, operational, and schedule feasibility.

Activities:

Introduction to Feasibility Analysis:

- Lecture: Understand the concept of feasibility analysis and its importance in system development.
- Discuss the different types of feasibility: technical, economic, operational, and schedule feasibility.
- Explain the role of feasibility analysis in the system development life cycle (SDLC).

Technical Feasibility:

- Discuss the technical requirements of the proposed system.
- Evaluate the availability of technology and expertise required for system development.
- Identify potential technical constraints and challenges.
- Group activity: Analyze the technical feasibility of a sample system and identify potential solutions to technical challenges.

Economic Feasibility:

- Discuss the cost-benefit analysis of the proposed system.
- Identify and quantify costs associated with system development, implementation, and maintenance.

- Estimate potential benefits and returns on investment (ROI) from the proposed system.
- Calculate key financial metrics such as net present value (NPV), return on investment (ROI), and payback period.
- Case study: Perform a cost-benefit analysis for a proposed system and make recommendations based on the findings.

Operational Feasibility:

- Understand the impact of the proposed system on existing business operations and processes.
- Evaluate the readiness of stakeholders to accept and adopt the new system.
- Identify potential organizational and cultural barriers to system implementation.
- Conduct surveys or interviews to gather feedback from end-users and stakeholders.
- Group discussion: Analyze the operational feasibility of a proposed system and propose strategies to mitigate potential challenges.

Schedule Feasibility:

- Define project timelines and milestones for system development and implementation.
- Identify potential risks and uncertainties that may impact project schedules.
- Develop a project schedule and identify critical path activities.
- Conduct a feasibility analysis to assess the likelihood of meeting project deadlines.
- Group exercise: Develop a project schedule for a proposed system and identify strategies to manage schedule risks.

Feasibility Report:

- Each group prepares a concise feasibility report summarizing their analysis. The report should include:

- ✓ Introduction: Briefly describe the proposed system.
- ✓ Feasibility Analysis: Discuss findings and insights for each feasibility aspect (operational, technical, economic, schedule).
- ✓ Recommendations: Based on the analysis, recommend whether the project should proceed, be modified, or be rejected, justifying the recommendation.

Conclusion:

The feasibility analysis of a proposed system is a critical step in the system development process. By evaluating technical, economic, operational, and schedule feasibility factors, stakeholders can make informed decisions about whether to proceed with the development and implementation of a system. This lab session equips students with the skills and knowledge to conduct comprehensive feasibility analyses and make recommendations based on their findings.

Section B:

Lab Assignment - Demonstrated by students

Assignment-2: Write a report on the ***Feasibility Analysis*** of your assigned project including:

i) Technical Feasibility
ii) Economic Feasibility
iii) Operational Feasibility

Lab #3: Process Modeling with Data Flow Diagrams

Section A: Tutorial - Demonstrated by Instructor

(30-60 minutes prior to lab Experiments)

Objectives:

- Understand the use of Data Flow Diagrams (DFDs) to depict system processes.
- Develop context and level 0 DFDs for a chosen system.

Activities:

- Analyze a system's processes and identify data flows and data stores.
- Create a context DFD, level 0 DFD and level 1 DFD for the chosen system using a drawing tool.

Contents:

- Introduction
- Data Flow Diagrams
  - ✓ Context Diagram
  - ✓ Level 0 and Level 1 DFDs
- Validating the Data Flow Diagrams
- Process Descriptions
- Conclusion

Data Flaw Diagram (DFD)

Scenario:

ExpressoCoffee processes a customer order in the following sequence:

First, customers Place an online order, which includes verifying the customer's identity, verifying the items' availability and verifying customer credit standing. To verify the customer credit standing and authorization request for the order total is sent to the credit card company. After receiving the customer credit confirmation, sales order information is stored in the sales order and sales order table. If the customer does not have good credit standing or any items in the customer's order are not available, then the customer order is rejected and customer is notified.

Second, the shipment is processed and sent to warehouse, which includes backing the ordered items, labeling them and then ship them to customer.

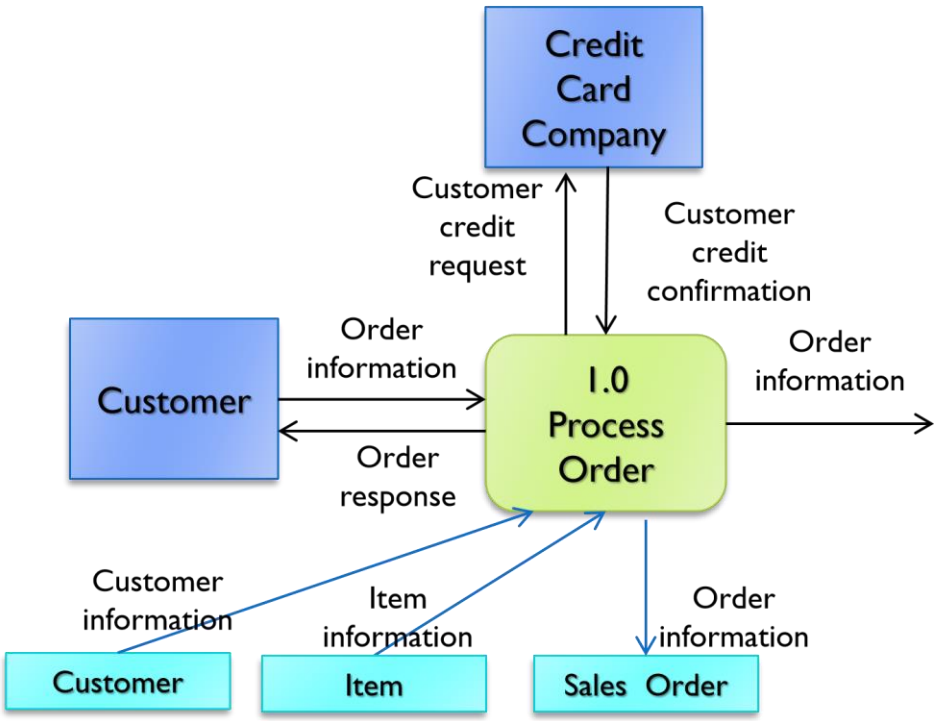
Third, an invoice is generated from the sales information and emailed to the customer. The corresponding database tables are updated.

1. Identify Entities, Process, Data Stores & Data Flow

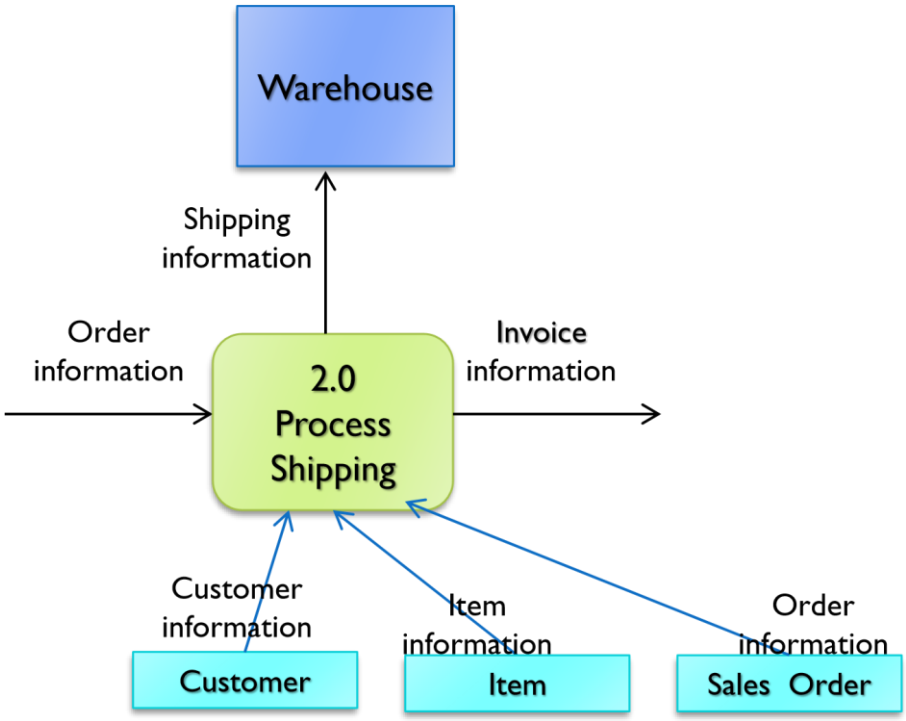
Process No	Process	Agent	Data Store
1	Process Customer Order	- Customer - Credit Card Company	- Customer - Sales Order - Items
2	Process Shipping	- Warehouse	- Customer - Sales Order - Items
3	Generate Invoice	- Customer	- Customer - Sales Order - Items

2. Draw a Level 0 Data Flow Diagram (DFD)

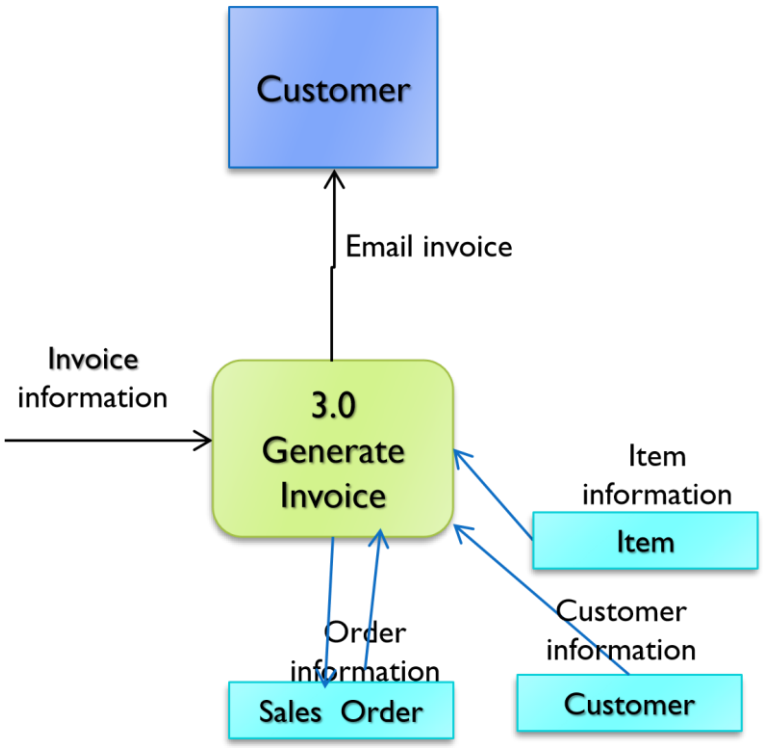
Process 1:



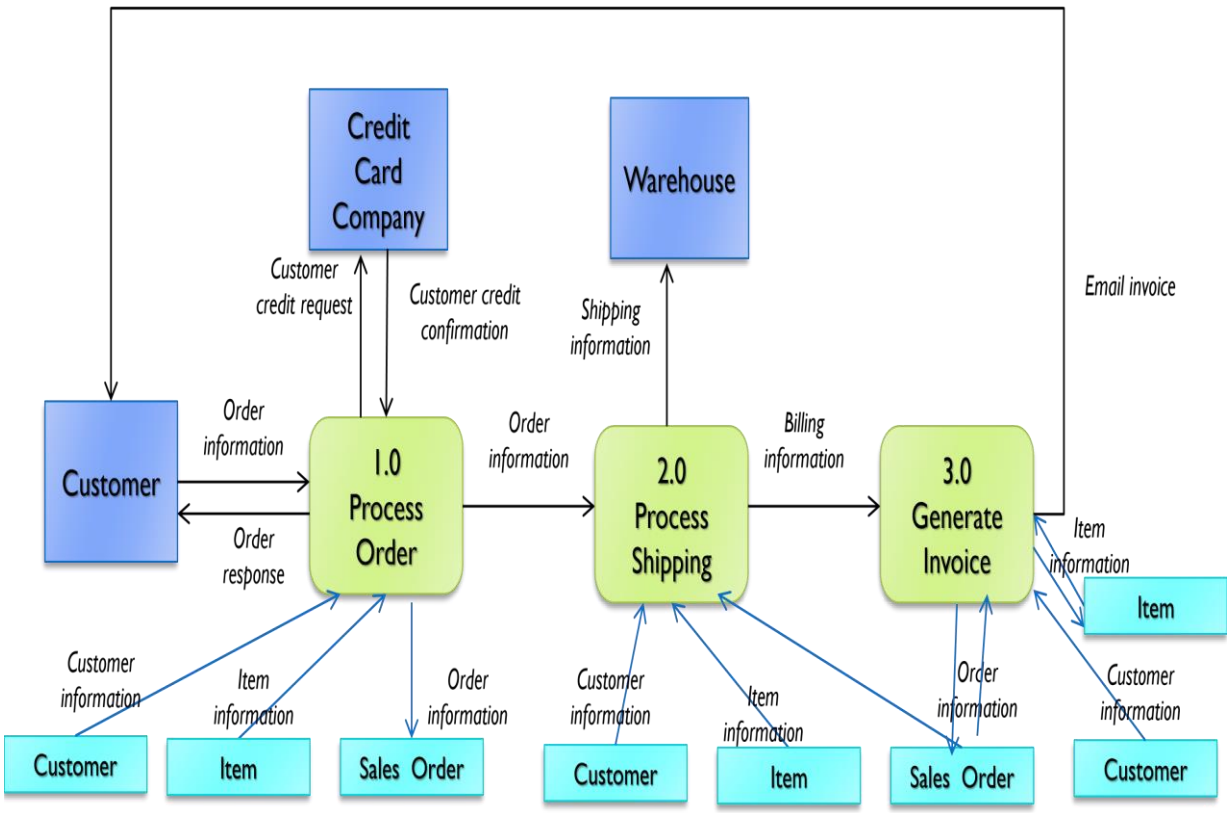
Process 2:



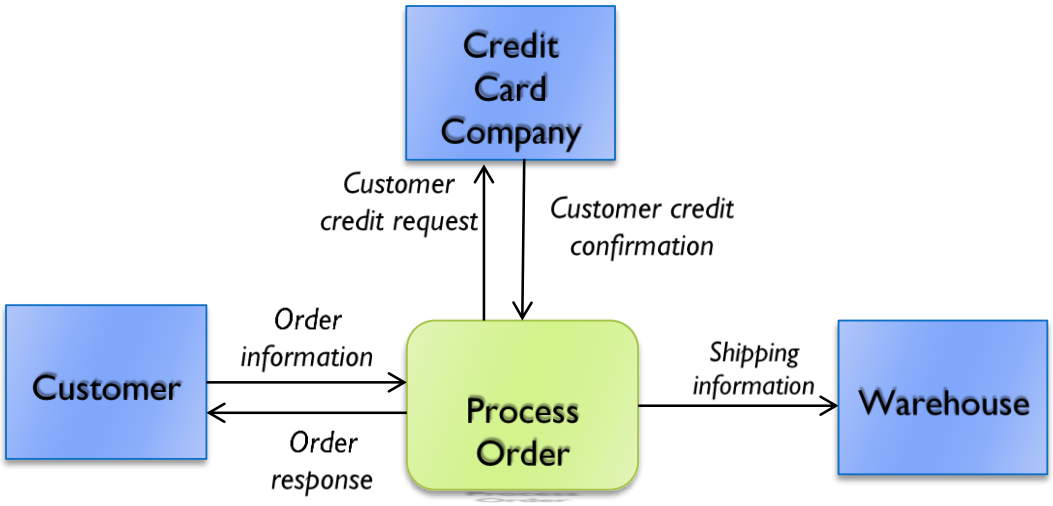
Process 3:



Level 0 Data Flow Diagram:



3. Draw a context diagram for the order system



4. Identify Entities, Process, Data Stores & Data Flow in LEVEL 1.

ExpressoCoffee processes a customer order in the following sequence:

First, customers *Place* an online order, which includes *verifying* the customer's identity, *verifying* the items' availability and *verifying* customer credit standing. To verify the customer credit standing and authorization request for the order total is sent to the credit card company. After receiving the customer credit confirmation, sales order information is *stored* in the sales order and sales order table. If the customer does not have good credit standing or any items in the customer's order are not available, then the customer order is rejected and customer is notified.

Second, the shipment is processed and sent to warehouse, which includes *backing* the ordered items, *labeling* them and then *ship* them to customer.

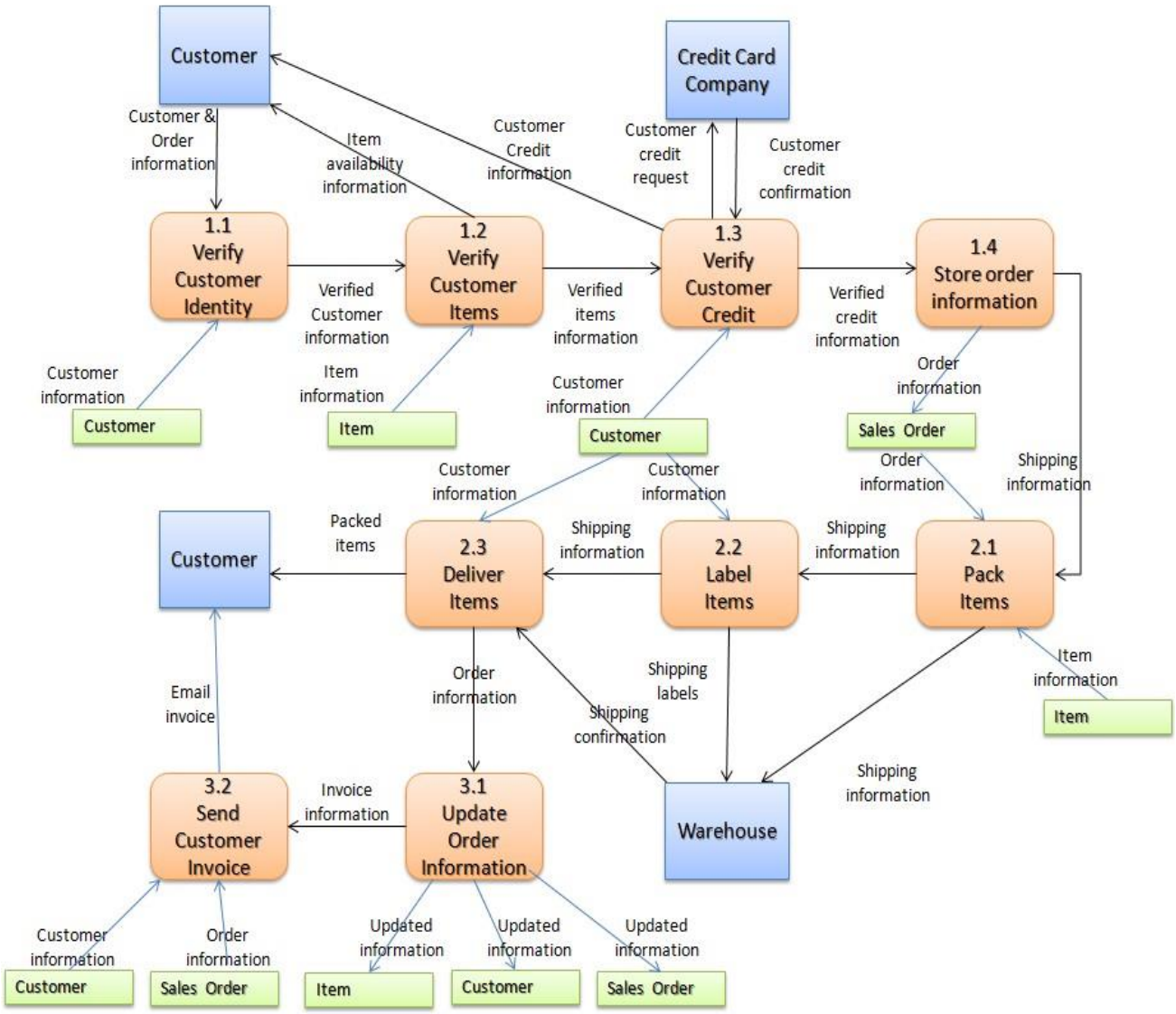
Third, an invoice is generated from the sales information and *emailed* to the customer. The corresponding database tables are *updated*.

5. Identify Entities, Process, Data Stores & Data Flow in LEVEL 1.

Process No	Process	Agent	Data Store
1	*Process Customer Order* *1. verify customer identity* *2. verify order item* *3. verify customer credit*	*- Customer* *- Credit Card Company*	*- Customer* *-Sales Order* *-Items*

	4. *Store order*		
2	*Process Shipping* 1. pack items 2. label items 3. ship items	*-Warehouse*	*- Customer* *-Sales Order* *-Items*
3	*Generate Invoice* 1. Send invoice 2. update customer order information	*- Customer*	*- Customer* *-Sales Order* *-Items*

6. Draw a Level 1 Data Flow Diagram



Section B:
Lab Assignment - Demonstrated by students

Assignment-3: Draw Data Flow Diagrams (DFDs) including

(i) Context Diagram, and

(ii) Level 0 and Level 1 DFDs of your selected project work.

Lab #4: Object Oriented Analysis and Design of the Proposed System

Section A: Tutorial - Demonstrated by Instructor

(30-60 minutes prior to lab Experiments)

Objectives:

- Introduce the Unified Modeling Language (UML).
- Understand different UML diagrams and their purposes.
- Practice creating UML diagrams for system modeling.

Activities:

- Lecture: Introduction to UML and its diagrams.
- Workshop: Creating Use Case, Class, Sequence, and Activity diagrams using UML.
- Project: Model a system using UML based on provided requirements.

Contents:

- Introduction
- UML Diagrams
    - ✓ Use Case Diagrams
    - ✓ Sequence Diagrams
    - ✓ Activity Diagrams
    - ✓ State Transition Diagrams
    - ✓ Class Diagrams
- Conclusion

OBJECT-ORIENTED ANALYSIS AND DESIGN

INTRODUCTION

Previously, all lessons discussed the analysis and design for an information system using structured techniques. We have learned how to analyze the requirements, modeled the data and process using various diagrams such as context diagram, data flow diagrams, and entity-relationship diagram. All this diagrams are part of the traditional structured system analysis and design. In this lesson, we will introduce the concept of object-oriented analysis and design. This lesson consists of four sections:

- Overview of object-oriented analysis and design
- The unified modeling language
- Object modeling with the unified modeling language
- Object-oriented development life cycle

LEARNING OUTCOMES

At the end of this lesson, students should be able to:

- define object-oriented approach in system development life cycle
- explain unified modeling language
- construct the object modeling tools with unified modeling language
- explain tasks involved in object-oriented development life cycle

TERMINOLOGY

No	Word	Definition
1	Activity diagram	A diagram that can be used to graphically depict the flow of a business process, the steps of a use case, or the logic of an object behavior (method).
2	Class diagram	A graphical depiction of a system's static object structure showing object classes that the system is composed of as well as the relationship between those object classes

3	Sequence diagram	A UML diagram that models the logic use of a use case by depicting the interaction of message between objects in time sequence
4	State transition diagram	A tool used to depict the sequence and variation of screens that can occur during a user session
5	Unified modeling language	is a widely used method of visualizing and documenting software system design
6	Use case diagram	A visual summary of several related use cases within a system or subsystem

4.1 OVERVIEW OF OBJECT-ORIENTED ANALYSIS AND DESIGN

In Lesson Five, we have introduced useful tools for system development such as case diagrams and class diagrams. These two types of tools are for object-oriented analysis and design (OOAD). In this lesson, we will go further discussing on OOAD and design. Nowadays, OOAD become popular because its ability to represent complex relationships, represent data with a consistent notation. All this techniques are associated with object-oriented language called Unified Modeling Language (UML).

4.2 UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a widely used method of visualizing and documenting software system design. UML uses object-oriented concepts, but it's independent of any specific programming language and can be used to describe business processes and requirements generally. UML provides various graphical tools such as use case diagram, sequence diagram, state transition diagrams and others. In the next subtopic, we will explain more on each technique.

The UML notation is useful for graphically depicting object-oriented analysis and design models. It is not only allows you to specify the requirements of a system and capture the system decisions, but also promotes communication among key person involved in the development effort. UML allows us to represent multiple views of a system using a variety of graphical diagrams such as use case, class, state, sequence, and collaboration diagrams.

4.3 OBJECT MODELING WITH THE UNIFIED MODELING LANGUAGE

In structured analysis, DFD and ERD is used to model data and processes, meanwhile in object-oriented, the UML is used to describe the object-oriented systems. The UML uses a set of symbols to represent graphically the various components and relationship within a system. There a several tools can be used for object-oriented modeling, and all this tools are associated with UML. These tools are use-case modeling, use case DIAGRAMS, class diagrams, sequence diagrams, state transition diagrams and activity diagrams. We will discuss further on this tools.

4.3.1 Use Case Modeling

In Lesson Five, we have discussed on use case. Use case represents a step 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. For example, STUDENT is an actor, and REGISTER A COURSE is a use case, as shown in Figure 4-1. UML use an oval with a label for a use case symbol that describes the action or event, while the actor is shown as a stick figure. The line associated between an actor and a use case called association. A use case can interact with another use cases. The UML indicates the relationship with a hollow-headed arrow that points that point at the use case being used. We can create use case by reviewing the information gathered during the requirement modeling in the system analysis phase.

When identifying the use cases, try to group all the related transactions into a single use-case. For example, when student register a course, the system will check the student status, check the student's credit limit, check the number of available in the course, update the student status, and send the confirmation to student as one use case called REGISTER A COURSE.

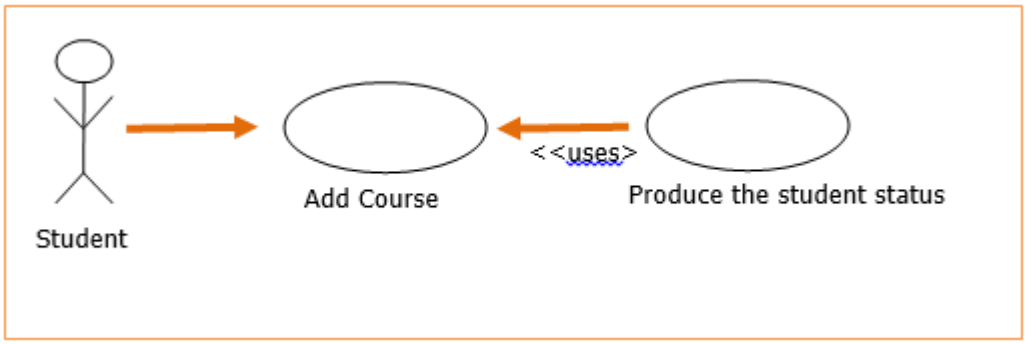


Figure 4-1: Use Case Modeling

4.3.2 Use Case Diagrams

A use case diagram is a visual summary of several related use cases within a system or subsystem. When creating a use case diagram, the first step is to identify the system boundary, which represent by a rectangle. The system boundary shows what is included in the system. After that, add the actors and show the relationship. Figure 4-2 shows an example of use case diagram. Use case diagram represent the interaction between users and the information system.

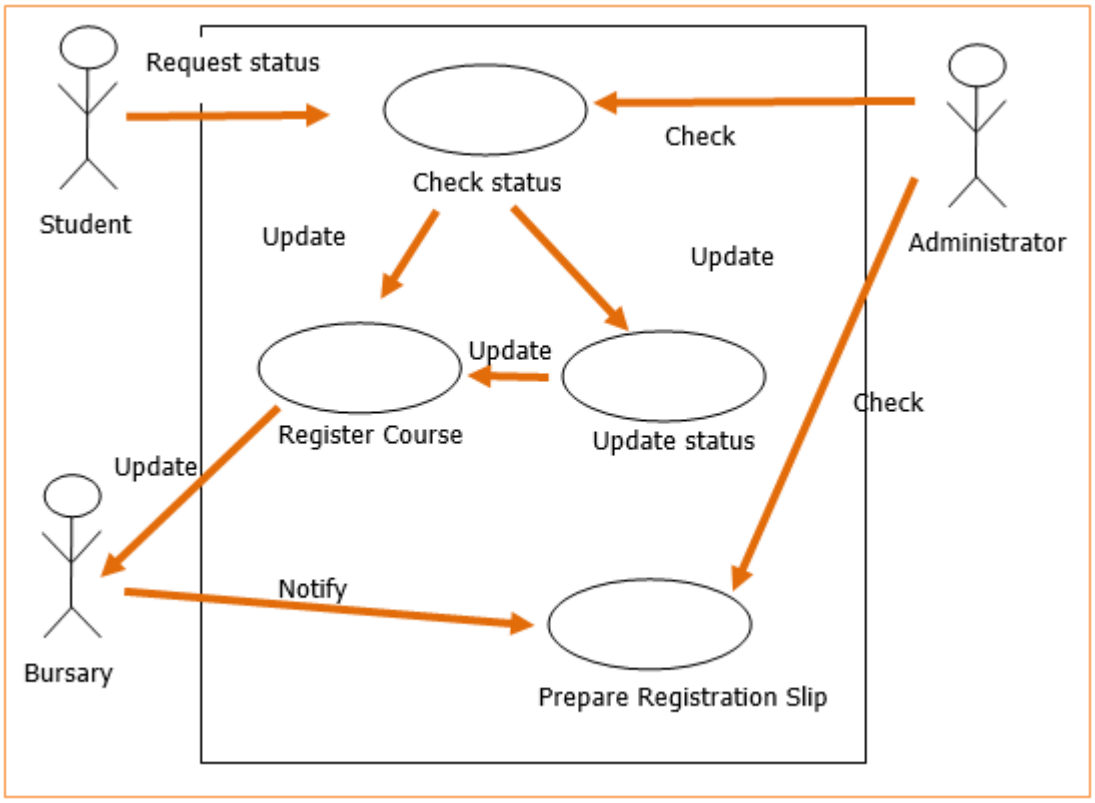


Figure 4-2: Use Case Diagram

4.3.3 Class Diagrams

A class diagram is a graphical depiction of a system's static object structure showing object classes that the system is composed of as well as the relationship between those object classes (Bentley *et. al.*, 2007). A class diagram represent a detailed view of a single use case, shows the classes that participate in the use case, and document the relationship among the classes. Class diagram is a logical model, then transformed to physical model and finally become functioning information system. Class diagram evolves into code modules, data objects, and other system components.

In class diagram, each class appears as a rectangle, with the class name at the top, followed by the class's attributes and methods. A line shows the relationship between classes and has labels identifying the action that relates the two classes. When constructing the diagram, the first step is to review the use case and identify the classes that participate in the underlying business transaction.

The class diagram also includes a concept called cardinality, which describe how instances of one class relate to instance of another class. Figure 4-3 shows an example of class diagram for student registration course.

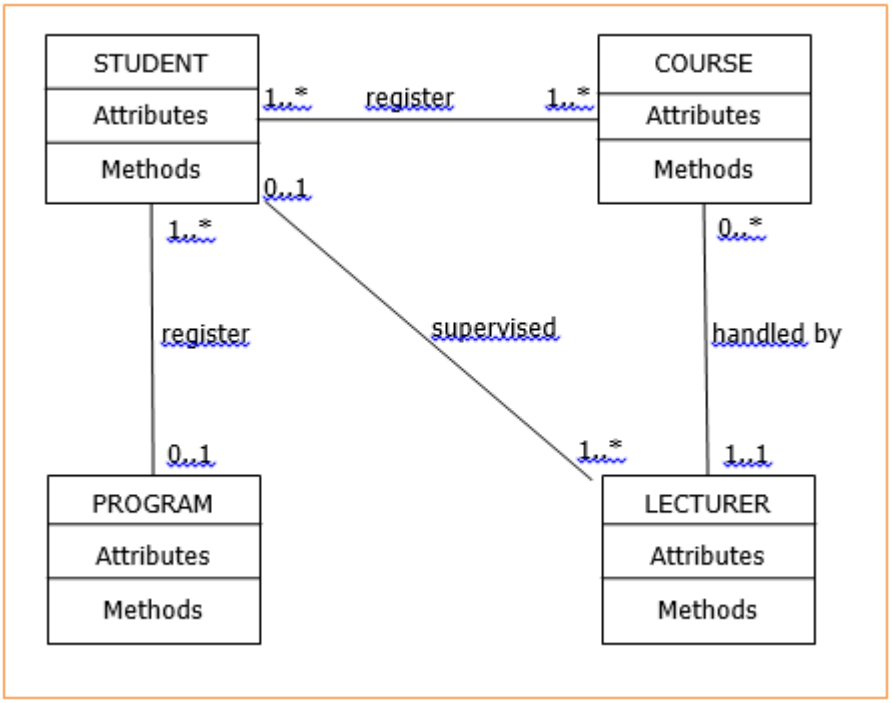


Figure 4-3: Class Diagram

4.3.4 Sequence Diagrams

A sequence diagram is a UML diagram that models the logic use of a use case by depicting the interaction of message between objects in time sequence (Bentley *et. al.*, 2007). A full sequence diagram depicts the interaction between all the object classes involved in the scenario. A sequence diagram models the logic of a use case by depicting the interaction between objects in the time sequence. The messages are arranged in time sequence from top to bottom. A sequence diagram can be seen as a way to integrate the steps of a use case with the objects of a class diagram. A sequence diagram graphically documents the

use case by showing the classes the messages, and the timing of the messages. Figure 4-4 shows an example of sequence diagram.

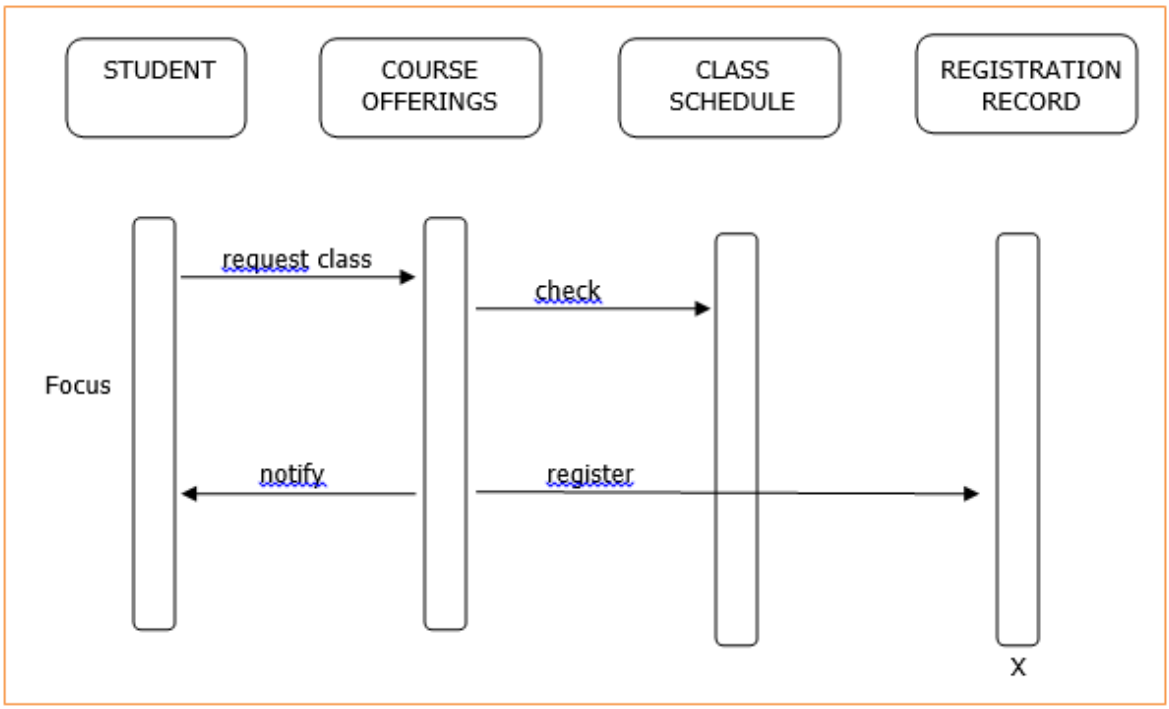


Figure 4-4: Sequence Diagram

Sequence diagram include symbols that represent classes, lifelines, messages and focuses.

- Classes – a class is identified by a rectangle with the name inside. Classes that send or receive messages are shown at the top of the sequence diagram
- Lifelines – A lifeline is identified by a dashed line. The lifeline represent the time during which the object above it 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. The label shows the name of the message and can include additional information about the contents
- Focuses – a focus is identified by a narrow vertical shape that covers the lifeline. The focus indicates when an object sends or receives a message.

4.3.5 State Transition Diagrams

State transition diagram is a tool used to depict the sequence and variation of screens that can occur during a user session (Bentley *et. al.*, 2007). State transition diagrams shows how an object changes from one state to another, depending on events that affect the object. All possible state must be documented in the state transition diagram. In a state transition diagram, the states appear as a rounded rectangle with the state names inside. The small circle to the left is the initial state, or the point where the object first interacts with the system. Reading from left to right, the lines show direction and describe the action or event that causes a transition from one state to another state. The circle at the right with a hollow border is the final state. Figure 4-5 shows an example of state transition diagram.

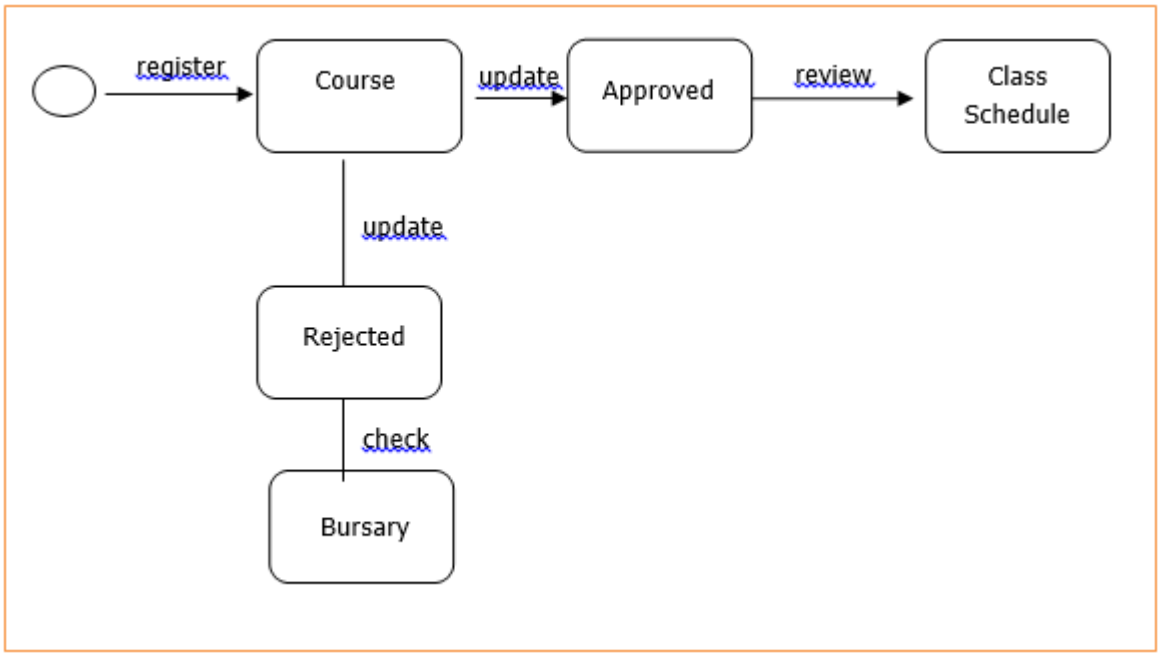


Figure 4-5: State Transition Diagram

4.3.6 Activity Diagram

An activity diagram is a diagram that can be used to graphically depict the flow of a business process, the steps of a use case, or the logic of an object behavior (method). It's similar with flowcharts; graphically depicting the sequential flow of activities of either a business process or a use case. But, it's different with flowchart because we can depict the activities that occur in parallel. Activity diagram is flexible, so that it can be used during both analysis and design phase. At least one activity diagram is needed for each use case. Normally, system analyst use activity diagram to better understand the flow and

sequencing of the use-case steps. Figure 4-6 shows an example of activity diagram. The solid circle representing the start of the process and the solid circle with the hollow inside represent the end of the process. The rounded rectangle represents individual steps, while the arrows indicate the progression through the actions. The diamond shape with one flow coming in and two or more flows going out used to represent the decision and the flow coming out are marked to indicate the conditions. The diamond shape with two or more flows coming in and one flow going out represent the merge.

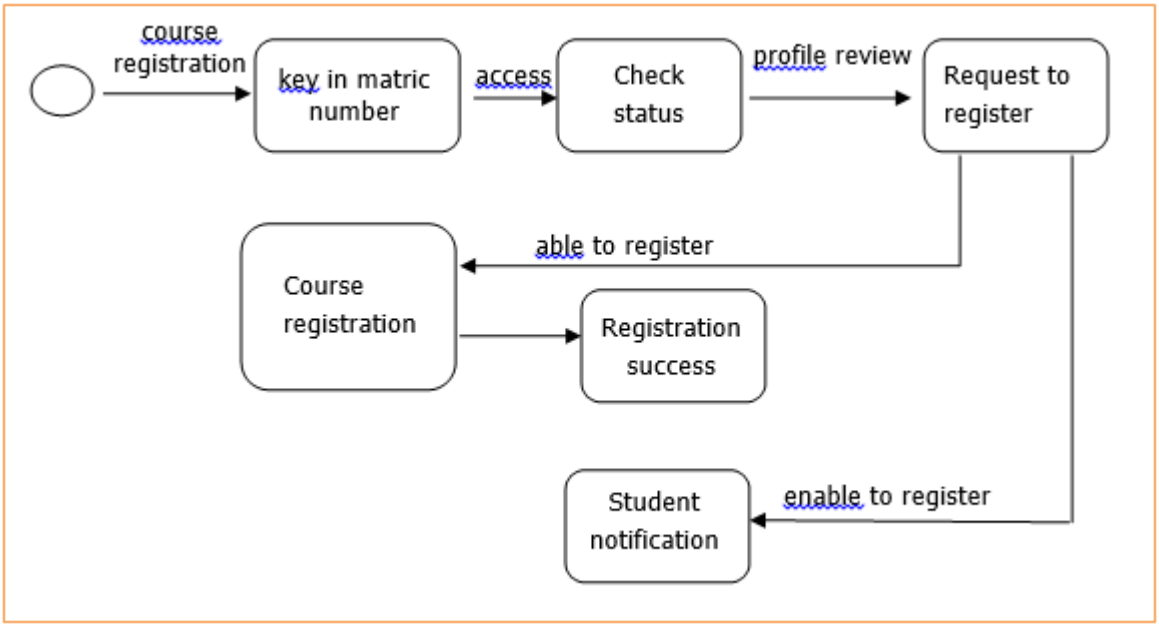


Figure 4-6: Activity Diagram

4.4 THE OBJECT-ORIENTED DEVELOPMENT LIFE CYCLE

Object-oriented development life cycle, as in Figure 4-7 consists of developing an object representation through three main phases – analysis, design and implementation. These three phases are quite similar with three main phases in structured development life cycle. The different is, in the early phases, the model that we develop is abstract, focusing on external qualities of the information system. Then, it focuses more how to build the system and how the system will function. It's related with system architecture, data structures and algorithms.

After completed the system requirements, it'll continued with the programming, and deciding the database access.

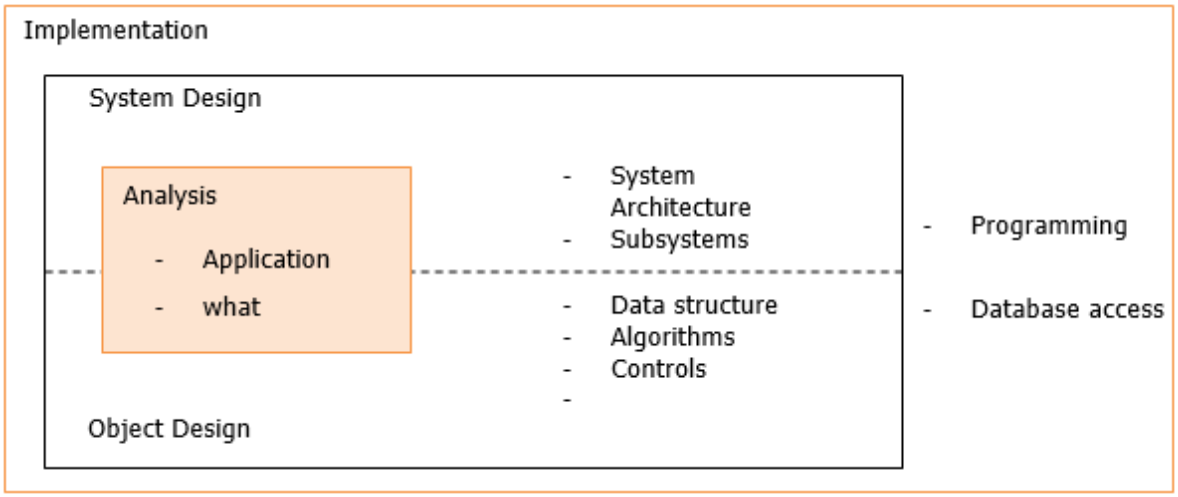


Figure 4-7: Phases of Object-Oriented System Development Cycle adapted from (Hoffer *et. al.*, 2005)

SUMMARY

This is the end of lesson. In this lesson, we have learned:

- overview of object-oriented analysis and design
- the unified modeling language
- object modeling with the unified modeling language
- object-oriented development life cycle

Section B:

Lab Assignment - Demonstrated by students

Assignment-4: Draw the following ***UML Diagrams*** for ***Object Oriented Analysis and Design*** of your selected project works:

i) Use Case Diagrams

ii) Sequence Diagrams

iii) State Transition Diagrams

iv) Activity Diagrams

v) Class Diagrams

Lab #5: Software Testing and Quality Assurance

Section A: Tutorial - Demonstrated by Instructor

(30-60 minutes prior to lab Experiments)

Objectives:

- Understand the importance of software testing in the SDLC.
- Learn various testing techniques and methodologies.
- Practice creating test cases and conducting software testing.

Activities:

- Lecture: Introduction to software testing and QA.
- Hands-on Exercise: Writing test cases for a given system.
- Group Project: Conducting system testing and reporting defects.

Contents:

- Introduction
- Test Case Design
- White-Box Testing
- Black-Box Testing
- Conclusion

Practical Example

Designing Test Cases for Login Function

Chosen System: Library Management System (LMS)

Module: Login Function

Functionality: The login function allows authorized users (library members and staff) to access the LMS by entering their username and password.

Test Case Design:

Test Case ID	Description	User name	Password	Expected Outcome	Pass /Fail
TC-01	Valid Login - Existing User	JohnDoe123	password123	User successfully logged in and directed to the LMS home page.	Pass
TC-02	Valid Login - Staff User	Staff100	staff_pwd	User successfully logged in and directed to the staff dashboard.	Pass
TC-03	Invalid Username - Non-existent User	JaneSmith456	password123	Error message displayed: "Invalid username or password."	Pass
TC-04	Invalid Password - Incorrect Password	JohnDoe123	wrong_password	Error message displayed: "Invalid username or password."	Pass
TC-05	Blank Username	""	password123	Error message displayed: "Username cannot be empty."	Pass
TC-06	Blank Password	JohnDoe123	""	Error message displayed: "Password cannot be empty."	Pass

TC-07	Excessively Long Username	ThisIsAReallyLongUsernameThatExceedsTheCharacterLimit	password123	Error message displayed: "Username exceeds the maximum character limit."	Pass
TC-08	Case-Sensitive Password	JohnDoe123	PASSWORD123	Error message displayed: "Invalid username or password."	Pass (Note: This assumes passwords are case-sensitive)

Explanation: These test cases cover various scenarios to ensure the login function operates as intended:

- **Valid Logins:** Test with existing user credentials and staff user credentials to check successful logins and redirection to appropriate sections.
- **Invalid Logins:** Test with non-existent usernames, incorrect passwords, blank fields, excessively long usernames, and case-sensitive passwords (if applicable) to verify error handling.

Note: This is a simplified example. The number and complexity of test cases can be expanded based on the specific system and its login functionality.

Section B:

Lab Assignment - Demonstrated by students

Assignment-5: Write a report on ***Testing*** of the system of your assigned project including:

i) Test Case Design

ii) White-Box Testing

iii) Black-Box Testing

Lab #6: System Implementation and Deployment

Section A: Tutorial - Demonstrated by Instructor

(30-60 minutes prior to lab Experiments)

Learning Outcomes:

- Understand the key phases and considerations involved in system implementation and deployment.
- Gain practical knowledge of different deployment strategies (e.g., phased, big bang).
- Identify various activities involved in data migration and user training.
- Develop a high-level implementation and deployment plan for a chosen system.

Pre-Lab Activities:

- Review the chapter on system implementation and deployment in your textbook.
- Watch a video on different deployment strategies:

  https://m.youtube.com/watch?v=OKM0tTYxQbk.

Lab Activities:

1. **Introduction:**
   - Briefly discuss the transition from system development to implementation and deployment.
   - Explain the different phases involved in system implementation and deployment, such as:
     - ✓ **Preparation:** Finalizing system testing, documentation, and training materials.
     - ✓ **Installation:** Setting up hardware and software infrastructure.
     - ✓ **Configuration:** Customizing the system for the specific environment.
     - ✓ **Data Migration:** Transferring data from the old system to the new system.
     - ✓ **User Training:** Educating users on how to use the new system.

- ✓ **Go-Live:** Launching the new system for production use.
- ✓ **Post-Deployment Support:** Providing ongoing support to users and addressing any issues.

2. **Deployment Strategies:**
   - Discuss different deployment strategies with their advantages and disadvantages:
     - ✓ **Phased Deployment:** Gradual rollout to different user groups or locations.
     - ✓ **Big Bang Deployment:** Full system implementation at once.
     - ✓ **Pilot Deployment:** Initial rollout to a limited group for testing and feedback before wider deployment.
   - Consider factors like system complexity, user impact, and risk tolerance when choosing a deployment strategy.
3. **Data Migration and User Training:**
   - Discuss the importance of data migration and its challenges (e.g., data cleansing, data conversion).
   - Highlight the significance of user training and different training methods (e.g., online training, classroom sessions).
   - Students can brainstorm strategies for data migration and user training considering a chosen system scenario.
4. **Implementation Plan Development:**
   - Divide students into groups (2-3 students per group).
   - Each group develops a high-level implementation and deployment plan for a chosen system (provided by the instructor or chosen based on specific criteria). The plan should include:
     - ✓ **Deployment Strategy:** Chosen strategy and justification.
     - ✓ **Timeline:** Estimated timeline for each implementation and deployment phase.
     - ✓ **Data Migration Plan:** Outline of data migration process, tools, and resources.
     - ✓ **User Training Plan:** Description of training methods, content, and schedule.
5. **Plan Presentation and Discussion:**
   - Each group presents their implementation and deployment plan to the class, highlighting key points and addressing any questions.

- Class discussion: Facilitate a discussion on the challenges and best practices associated with system implementation and deployment. Students can share their insights and learnings.

Post-Lab Activities:

- Reflect on the importance of careful planning and execution in successful system implementation and deployment.
- Research and discuss emerging trends in system deployment, such as cloud-based deployment or containerization.

Resources:

- System Deployment Guide:

  https://www.techtarget.com/searchcustomerexperience/definition/implementation

- User Training Plan Template:

  https://create.microsoft.com/en-us/templates/employee-training

Section B:

Lab Assignment - Demonstrated by students

Assignment-6: Write a report on ***Implementation***, ***Documentation*** and ***Deployment*** of the system of your assigned project.