APT 3040: OBJECT ORIENTED ANALYSIS AND DESIGN & PROGRAMMING

Prerequisite: APT 1030 Fundamentals of Programming Languages

3 Credit Units

Rationale

The overall goal of this course is to ingrain to the students the object approach to systems design and programming and to show why it is important and superior compared to traditional approaches like the conventional model, in which a program is seen as a list of tasks to perform. At the heart of object-oriented approach, instead of tasks we find objects – entities that have behaviors, that hold information, and that can interact with one another. Programming consists of designing a set of objects that model the problem at hand. Software objects in the program can represent real or abstract entities in the problem domain. This is supposed to make the design of the program more natural and hence easier to get right and easier to understand. Each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent "machine" with a distinct role or responsibility. The actions (or "methods") on these objects are closely associated with the object. OOP data structures tend to carry their own operators around with them or at least inherit them from a similar object or class.

Course Description

This course unit aims to describe what object-oriented (OO) software is all about. More specifically, to teach the concepts, tasks and notation (using UML). Introduction to the Design Process Improvement Model UML Structural Modeling Techniques. Conceptual model: the result of object-oriented analysis, captures concepts in the problem domain. Use case: description of sequences of events that, taken together, lead to a system doing something useful. Use case actors; Use case diagrams; System Sequence Diagrams (SSD); - picture for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events. User interface documentations: look and feel of the end product's user interface. Relational data model: abstract model that describes how data is represented and used.

Learning Outcomes

After successful completion of the course unit, a student will be able to:

1. Design software in an object-oriented manner
2. Use UML as a notation to support this design
3. Apply structural and behavioral modeling techniques.
4. Use model-based software development methodology.

5. Demonstrate usage of the methodology and the modeling techniques in a significant software design project.

6. Identify and describe design patterns and their application in a software design project.

7. Describe and illustrate usage Design and Testing Process Improvement Models.

Course Content

Review of object oriented concepts ;Objects ; Classes ; Inheritance; Object Oriented Type Systems. Software Development Methodology; Engineering or invention? ; Example Artifacts using UML ; CRC Cards. Requirements Capture ;Introduction ; Business Perspective; Developer Perspective. Analysis ;Introduction ; Static Analysis ; Dynamic Analysis. System Design ; Introduction ; Networked System Topologies ; Choosing Technologies ; Partitioning Software. Subsystem Design; Designing the Business Logic; Persistence using a Relational Database; Finalizing the User Interfaces; Designing the Business Services; Thread Safety . Code Specification; Background; Object-Oriented Specification; Design by Contract; Informal Specification in Java

Teaching Methodology

The primary teaching methods will be lectures and demonstrations. The student will attend lectures and demonstrations participate in discussion on assigned readings, complete assigned projects, and complete required tests and examinations.

Instructional material & equipment

Textbooks, whiteboard, handouts, electronic projector and laptop, Internet access, software and library.

Methods of evaluation

Laboratory Work	20%
Project	20%
Assignments	10%
Mid-semester	20%
Final semester exams	30%
Total	**100%**

Course Text

Object-oriented systems analysis and design using uml 3rd ed by Bennett, Simon 2006

Head first **object-oriented analysis** and **design** by Brett McLaughlin, Gary Pollice, David West – 2006

Recommended Reading

Object-oriented analysis and **design**: a pragmatic approach by John Deacon - 2005