CHITTAGONG UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

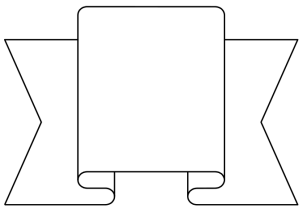
Report - 3

Object-Oriented Analysis and Design of Systems of IICT

COURSE CODE : CSE 354
COURSE NAME : SYSTEM ANALYSIS & DESIGN (SESSIONAL)

submitted by

SHAFIUL ALAM	(1904006)
MD AKIB HASAN	(1904015)
K.M. MAHABUB HOSSAIN	(1904017)
SADMAN RAHMAN ANANTA	(1904020)
SAWMIK KUMAR PAUL	(1904021)



REMARKS

supervised by

DR. KAUSHIK DEB
Professor
Department of CSE, CUET

SABIHA ANAN
Assistant Professor
Department of CSE, CUET

Introduction

Object-Oriented Analysis and Design (OOAD) is a software engineering methodology that involves using object-oriented concepts to design and implement software systems. It involves a number of techniques and practices, including object-oriented programming, design patterns, UML diagrams, and use cases.
IICT is a special institute of CUET which performs various tasks with its different sections, which are managed by different individuals. Those various sections play a significant role in terms of digitalization of CUET.
The dynamic behavior of a system such as IICT can be represented by a use case diagram. It incorporates use cases, actors, and their relationships to encapsulate the functionality of the system.

Object-Oriented Analysis

Object-Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises interacting objects.
The main difference between object-oriented analysis and other forms of analysis is that in object-oriented approach, requirements are organized around objects, which integrate both data and functions. They are modelled after real-world objects that the system interacts with. Such as,

- Identifying objects
- Organizing the objects by creating object model diagram
- Defining the internals of the objects, or object attributes
- Defining the behavior of the objects, i.e., object actions
- Describing how the objects interact

Object-Oriented Design

Object-Oriented Design (OOD) involves implementation of the conceptual model produced during object-oriented analysis. In OOD, concepts in the analysis model, which are technology-independent, are mapped onto implementing classes, constraints are identified and interfaces are designed, resulting in a model for the solution domain, i.e., a detailed description of how the system is to be built on concrete technologies.

- Restructuring the class data (if necessary),

- Implementation of methods, i.e., internal data structures and algorithms,
- Implementation of control, and
- Implementation of associations.

OOAD involves a number of techniques and practices, including object-oriented programming, design patterns, UML diagrams, and use cases. Some important aspects of OOAD are:

OOP: Object-oriented programming involves modeling real-world objects as software objects, with properties and methods that represent the behavior of those objects.

Design Patterns: Design patterns are reusable solutions to common problems in software design. OOAD uses design patterns to help developers create more maintainable and efficient software systems.

UML Diagrams: Unified Modeling Language (UML) is a standardized notation for creating diagrams that represent different aspects of a software system.

Use Cases: Use cases are a way of describing the different ways in which users interact with a software system.

UML Diagrams

The Unified Modeling Language (UML) is a graphical language for OOAD that provides a uniform way to describe the architecture of a software system. An object-oriented system's artifacts can be seen, described, built, and documented. It is used to represent the relationships and structures in a complex system. The Conceptual Model of UML encompasses three major elements –

- Basic building blocks
- Rules
- Common mechanisms

The three building blocks of UML are –

Things Structural things, behavioral things, grouping things, annotational things etc.

Relationships Dependency, association, generalization, realization etc.

Diagrams Class diagram, object diagram, use case diagram etc.

Furthermore, UML has a number of rules, so that the models are semantically self-consistent and related to other models in the system such as – names, scope, visibility, integrity etc. and mechanisms such as – specifications, adornments, common divisions.

Use Cases

Use case diagrams, also known as behavior diagrams, are used to illustrate a series of tasks that a system or systems (the subject) ought to or is capable of carrying out in coordination with one or more external users of the system (the actors). It simulates the duties, services, and operations needed by a system or application subsystem. It shows a system's high-level functionality and also describes how a user interacts with a system.

Use cases A use case describes a function that a system performs to achieve the user's goal. A use case must yield an observable result that is of value to the user of the system.

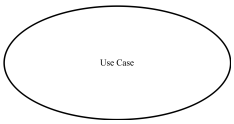


Figure 1: Use Case

Actors An actor represents a role of a user that interacts with the system that you are modeling. The user can be a human user, an organization, a machine, or another external system.

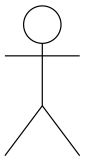


Figure 2: Actor

Subsystems In UML models, subsystems are a type of stereotyped component that represent independent, behavioral units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modeling.

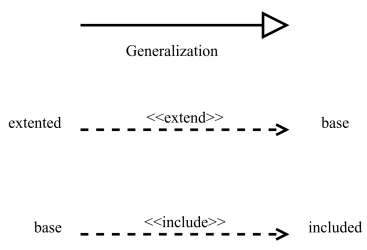


Figure 3: Relationships

Relationships In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between the model elements.

Use Case Diagrams

To achieve the aims and objectives, the institute is performing under two wings;

1. Academic Wing and
2. System and Support Wing.

Academic wing provides technical and training courses and other services for students and faculties, whereas the support wing provides networking and technical support. Most of the services performed by various sections are –

i. Course Registration
ii. Email Services
iii. Issuing ID Card
iv. Offered Courses by IICT
v. Internet Service
vi. Payment System
vii. Surveillance System

The use case diagrams of different sections of IICT are given below:

Academic Wing

Course Registration System

There are four actors in course registration system such as student, hall authority, advisor, and department head. Student applies for hall approval from hall authority after clearing due of that semester and the authority provides the approval. Then student also collects approval from their department head and respective advisors.

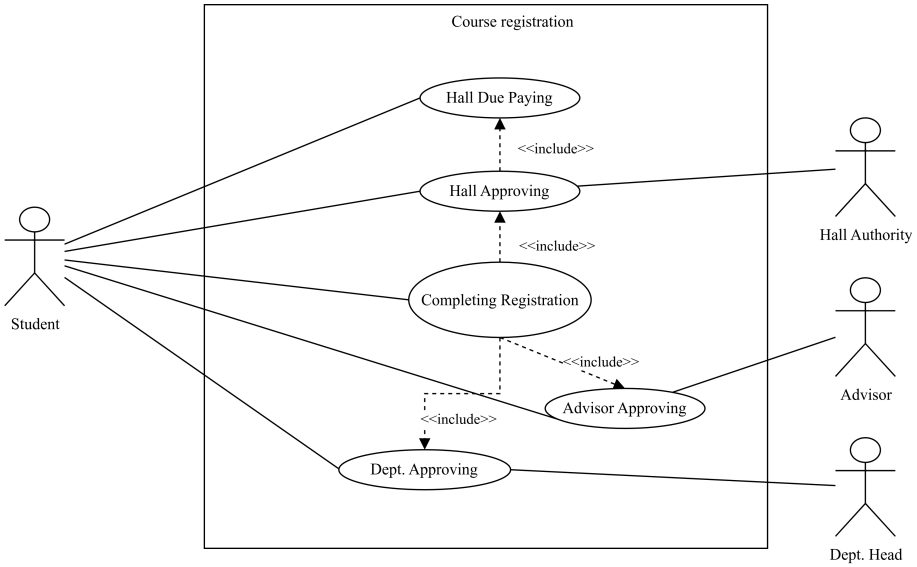


Figure 4: Use case diagram for course registration

With required approvals, the registration system in IICT completes the students registration and provides the filled forms and admit card for exam automatically.

Email Services

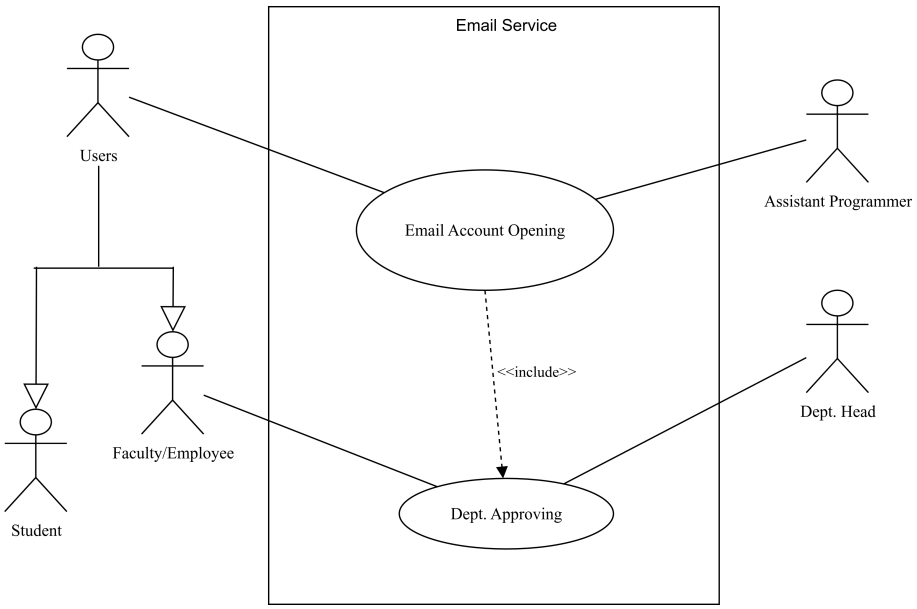


Figure 5: Use case diagram for email service

Students and other faculties are provided the facility of institutional email service including storage and others by IICT. There are three main actors where the user actor is specialized into students and faculties/officers. A bulk list of students are provided by the department to IICT after admission with department head's approval and the mail

ID are generated automatically and provided to the students. For new faculty or employee, they ask approval from their respective department and with necessary approval assistant programmer performs email account opening action.

Issuing ID Card

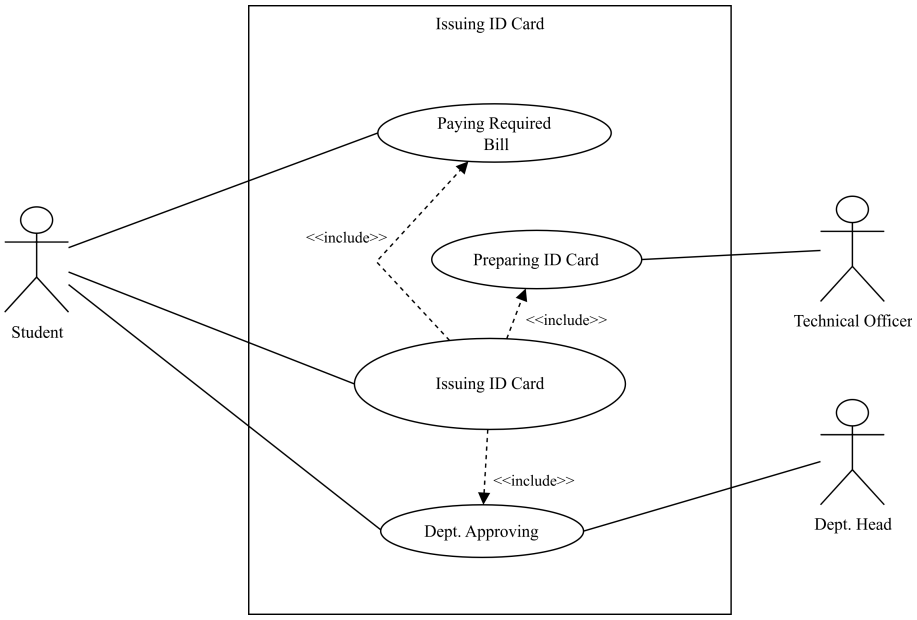


Figure 6: Use case diagram for issuing ID card

There are three actors in this section such as students, department head and technical officer. Student performs bill payment and takes department approval and then asks for ID card. Technical officer issues the ID card.

Offered Courses by IICT

Various courses are offered by IICT frequently for the students of the university and outsiders. There are three actors in this section who performs the required job such as applicant, system analyst and assistant programmer. First applicant applies and assistant programmer checks the eligibility of the applicant, and the system analyst enrolls him to the respective course.

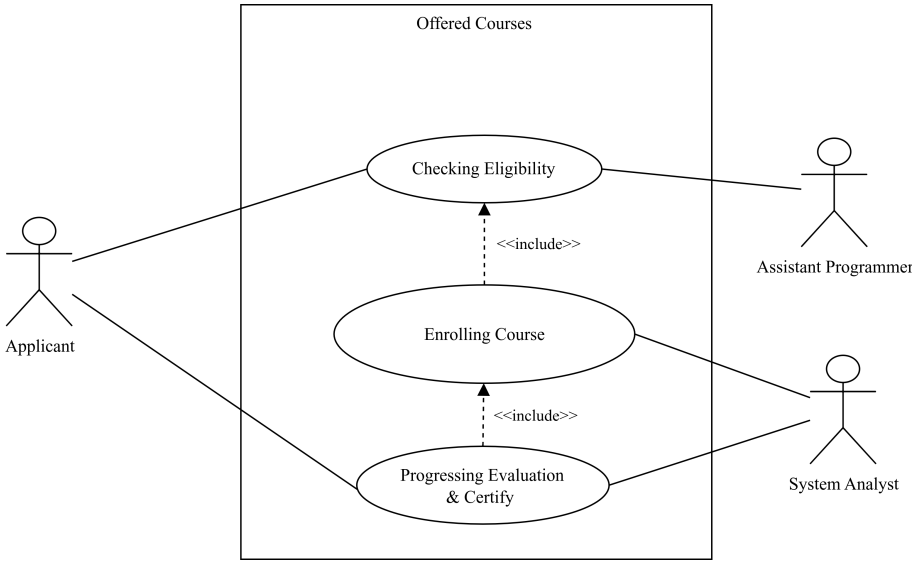


Figure 7: Use case diagram for offered courses

Finally, the applicant can collect the performance evaluation and certificate which is also controlled by system analyst.

System & Support

Providing Internet Service

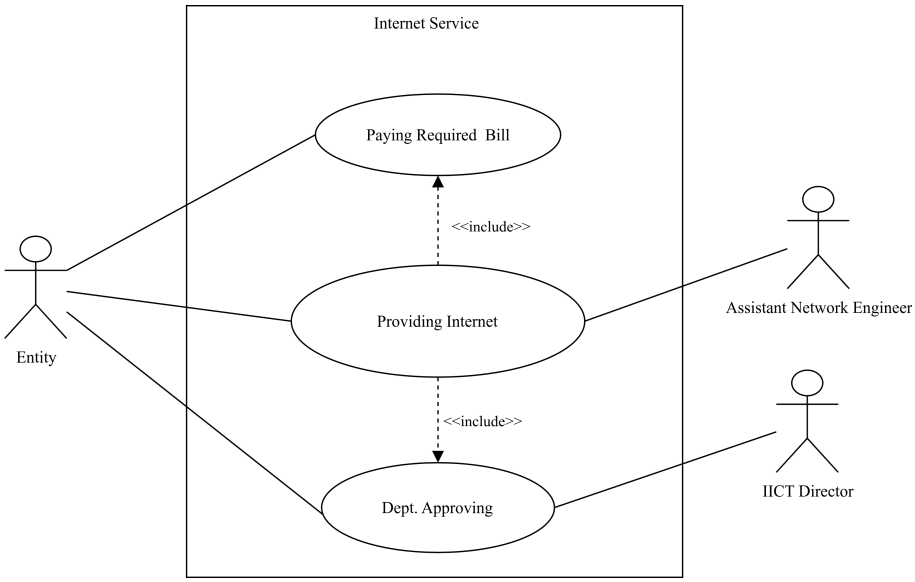


Figure 8: Use case diagram for internet service

Internet service is provided by IICT covers almost the whole university. For a new building or institution this service is extended immediately by them. In this section there are three main actors– entity, IICT director and network engineer. The authority of newly constructed institution pays the required bill and the IICT director provides

the necessary approval. Finally, the network engineer takes essential steps to extend the internet service.

Payment System

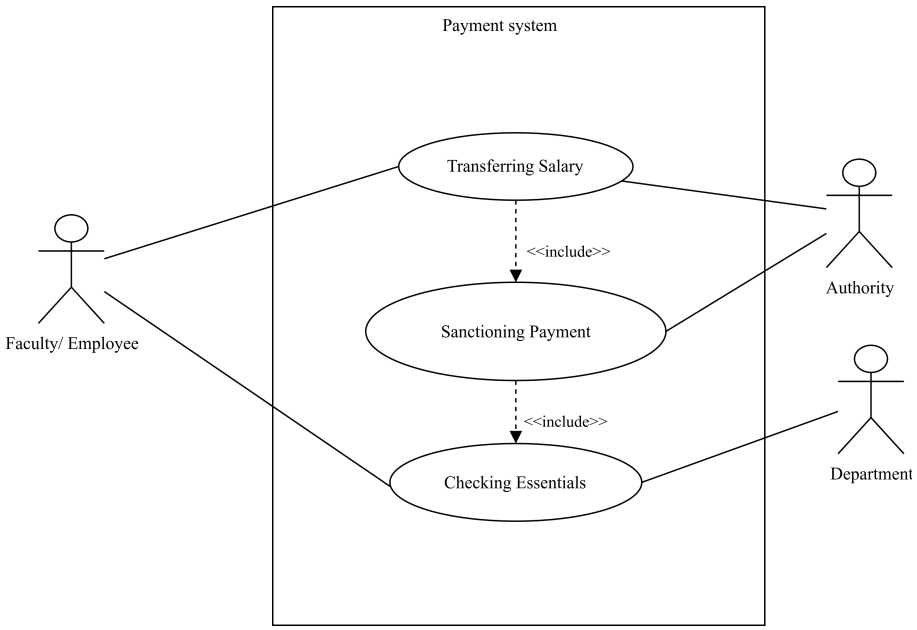


Figure 9: Use case diagram for payment system

The payment system for faculties and officials are semi-automated process conducted by IICT. There are three actors as usual and the employee requests forms with the departments approval and sanctioning payment system with salary transfer system by authority provides the payment.

Surveillance System

There are four actors in the surveillance system of IICT such as victim, CCTV, DSW and IICT director. The victim can apply for any footage covered by IICT CCTV with time stamp. With the approval of the DSW, CUET and the director the footage is supplied.

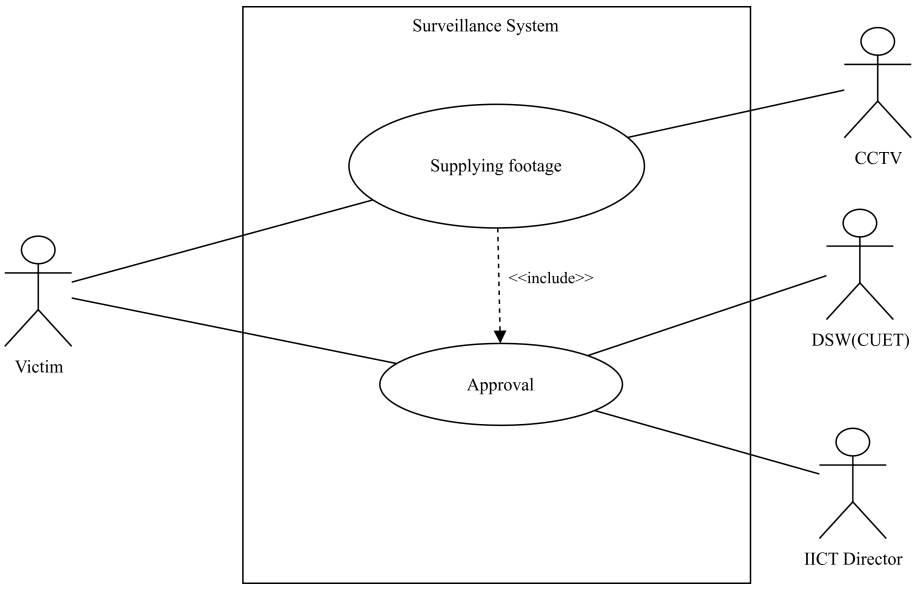


Figure 10: Use case diagram for surveillance system

Conclusion

A graphical language for visualizing, describing, building, and documenting software-intensive system artifacts is called Unified Modeling Language (UML) and a subset of UML diagrams is use case diagrams. We used use case diagrams to visualize various aspects of the academic and administrative tasks performed by IICT. Those section's workflow are mostly unique. And there is hardly any relations in between two sections. Finally, the functional behavior, event flow, and involved parties of IICT is illustrated simply using use case and UML diagrams.