9. Object Oriented Design

IT 3106– Object Oriented Analysis and Design

Level II - Semester 3

Overview

In this section students will learn

- the activities involved in object-oriented design.

Intended Learning Outcomes

At the end of this lesson students will be able to

- understand the verification and validation of the analysis models,
- understand the transition from analysis to design,
- create package diagrams, verify and validate them.

List of sub topics

9.1 Introduction to Object Oriented Design [Ref 1: Pg. 240-241]

9.2 Verifying and Validating the Analysis Models [Ref 1: Pg.242-257]

- 9.2.1 Balancing Functional and Structural Models
- 9.2.2 Balancing Functional and Behavioral Models
- 9.2.3 Balancing Structural and Behavioral Models

9.3 Evolving Analysis Models into Design Models [Ref 1: Pg. 257-262]

9.4 Package Diagrams [Ref 1: Pg. 262-268]

- 9.4.1 Guidelines for Creating Package Diagrams
- 9.4.2 Creating Package Diagrams
- 9.4.3 Verifying and Validating Package Diagrams

Ref 1: Alan Dennis, Barbara Haley, David Tegarden, Systems analysis design, An Object Oriented Approach with UML: an object oriented approach, 5th edition, John Wiley & Sons, 2015, ISBN 978-1-118-80467-4

9.1 Introduction to Object Oriented Design

- The purpose of analysis is to figure out the business needs.
- The purpose of design is to decide how to build the system.
- The major activity that takes place during design is evolving the set of analysis representations into design representations.
- An important initial part of design is to examine several design strategies and decide which will be used to build the system.
  - e. g. Systems can be built from scratch, purchased and customized, outsourced to others etc.
- The project team needs to investigate the viability of each alternative. This decision influences the tasks that are to be accomplished during design.

9.1 Introduction to Object Oriented Design cont…

- Design includes activities such as designing the user interface, system inputs, and system outputs, which involve the ways that the user interacts with the system.
- Physical architecture decisions are made regarding the hardware and software that will be purchased to support the new system and the way that the processing of the system will be organized.

  e.g. The system can be organized so that its processing is centralized/distributed.

9.2 Verifying and Validating the Analysis Models

Before we evolve our analysis representations into design representations:

- We need to verify and validate the current set of analysis models to ensure that they faithfully represent the problem domain under consideration.
- We need to give attention on ensuring that the different models are consistent.

  e.g.

  - We must be sure that the activity diagram(s), use-case descriptions, and use case diagrams all describe the same functional requirements.
  - Transitions on a behavioral state machine are associated with operations contained in a class diagram.

9.2 Verifying and Validating the Analysis Models

Balancing Functional and Structural Models:

- We must ensure that the two sets of models are consistent with each other.
- The activity diagrams, use-case descriptions, and use-case diagrams must agree with the CRC cards and class diagrams that represent the evolving model of the problem domain.

E.g.

- Every class on a class diagram and every CRC card must be associated with at least one, use case and vice versa.
- Every activity or action contained in an activity diagram and every event contained in a use-case description should be related to one or more responsibilities on a CRC card and one or more operations in a class on a class diagram and vice versa.
- Every object node on an activity diagram must be associated with an instance of a class on a class diagram and a CRC card or an attribute contained in a class and on a CRC card.
- Every attribute and association/aggregation relationships contained on a CRC card (and connected to a class on a class diagram) should be related to the subject or object of an event in a use-case description.

9.2 Verifying and Validating the Analysis Models

Balancing Functional and Behavioral Models

- We must ensure that the two sets of models are consistent with each other.
- The activity diagrams, use-case descriptions, and use-case diagrams must agree with the sequence diagrams, communication diagrams, behavioral state machines, and CRUDE matrix.

Eg.

- The sequence and communication diagrams must be associated with a use-case on the use-case diagram and the use-case description.
- Actors on sequence diagrams, communication diagrams, and/or CRUDE matrices must be associated with actors on the use-case diagram or referenced in the use case description, and vice versa.
- Messages on sequence and communication diagrams, transitions on behavioral state machines, and entries in a CRUDE matrix must be related to activities and actions on an activity diagram and events listed in a use-case description, and vice versa.
- All complex objects represented by an object node in an activity diagram must have a behavioral state machine that represents the object's lifecycle, and vice versa.

9.2 Verifying and Validating the Analysis Models

Balancing Structural and Behavioral Models

- We must ensure that the two sets of models are consistent with each other.
  - Objects that appear in a CRUDE matrix must be associated with classes that are represented by CRC cards and appear on the class diagram, and vice versa.
  - Behavioral state machines must be associated with instances (objects) of classes on a class diagram and with a CRC card that represents the class of the instance.
  - Communication and sequence diagrams contain objects that must be an instantiation of a class that is represented by a CRC card and is located on a class diagram.
  - Messages contained on the sequence and communication diagrams, transitions on behavioral state machines, and cell entries on a CRUDE matrix must be associated with responsibilities and associations on CRC cards and operations in classes and associations connected to the classes on class diagrams.
  - The states in a behavioral state machine must be associated with different values of an attribute or set of attributes that describe an object.

9.3 Evolving Analysis Models into Design Models

- After successfully verifying and validating the analysis models, we need to begin evolving them into appropriate design models.
- In the analysis we defined the functional requirements whereas in systems design, we address both the functional and nonfunctional requirements.
- When evolving the analysis model into the design model, you should first carefully review the use cases and the current set of classes (their operations and attributes and the relationships).
  - ✓Are all the classes necessary? Are there any missing classes?
  - ✓Are the classes fully defined?
  - ✓Are any attributes or methods missing?
  - ✓ Do the classes have any unnecessary attributes and methods?
  - ✓Is the current representation of the evolving system optimal?

9.3 Evolving Analysis Models into Design Models

- Object-oriented systems development is both incremental and iterative. Hence, we must review the analysis models again.
- Start looking at the models of the problem domain through a design lens.
- Factoring, partitions and collaborations, and layers can be used to evolve problem domain-oriented analysis models into optimal solution domain-oriented design models.

Factoring: Factoring is the process of separating out a module into a stand-alone module (new class or new method)

Eg. When reviewing a set of classes, it may be discovered that they have a similar set of attributes and methods. The new class can be related to the existing classes through a generalization (a-kind-of) or possibly through an aggregation (has-parts) relationship.

Partitions and Collaborations: in the evolution of the system, it might make sense to split the representation into a set of *partitions*. A partition is the object-oriented equivalent of a subsystem. An easy approach to model partitions and collaborations is the use of packages and package diagrams.

A good place to look for potential partitions is the collaborations modeled in UML's communication diagrams.

9.3 Evolving Analysis Models into Design Models

Layers: To successfully evolve the analysis model of the system into a design model of the system, we must add the system environment information. One useful way to do this, without overloading the developer, is to use layers.

A layer represents an element of the software architecture of the evolving system.

We have focused only on one layer in the evolving soft ware architecture: the problem domain layer.

There should be a layer for each of the different elements of the system environment (e.g., data management, user interface, physical architecture).

Like partitions and collaborations, layers also can be portrayed using packages and package diagrams

Eg. Problem Domain Layers – Employee, Customer

Data Management Layers - DataInputStream, FileInputStream

Human–Computer Interaction Layers- Button, Panel

Refer Recommended text for other examples.

9.4 Package Diagrams

- In UML, collaborations, partitions, and layers can be represented by a higher-level construct: a package . A package serves the same purpose as a folder on your computer.
- A *package* is a general construct that can be applied to any of the elements in UML models.

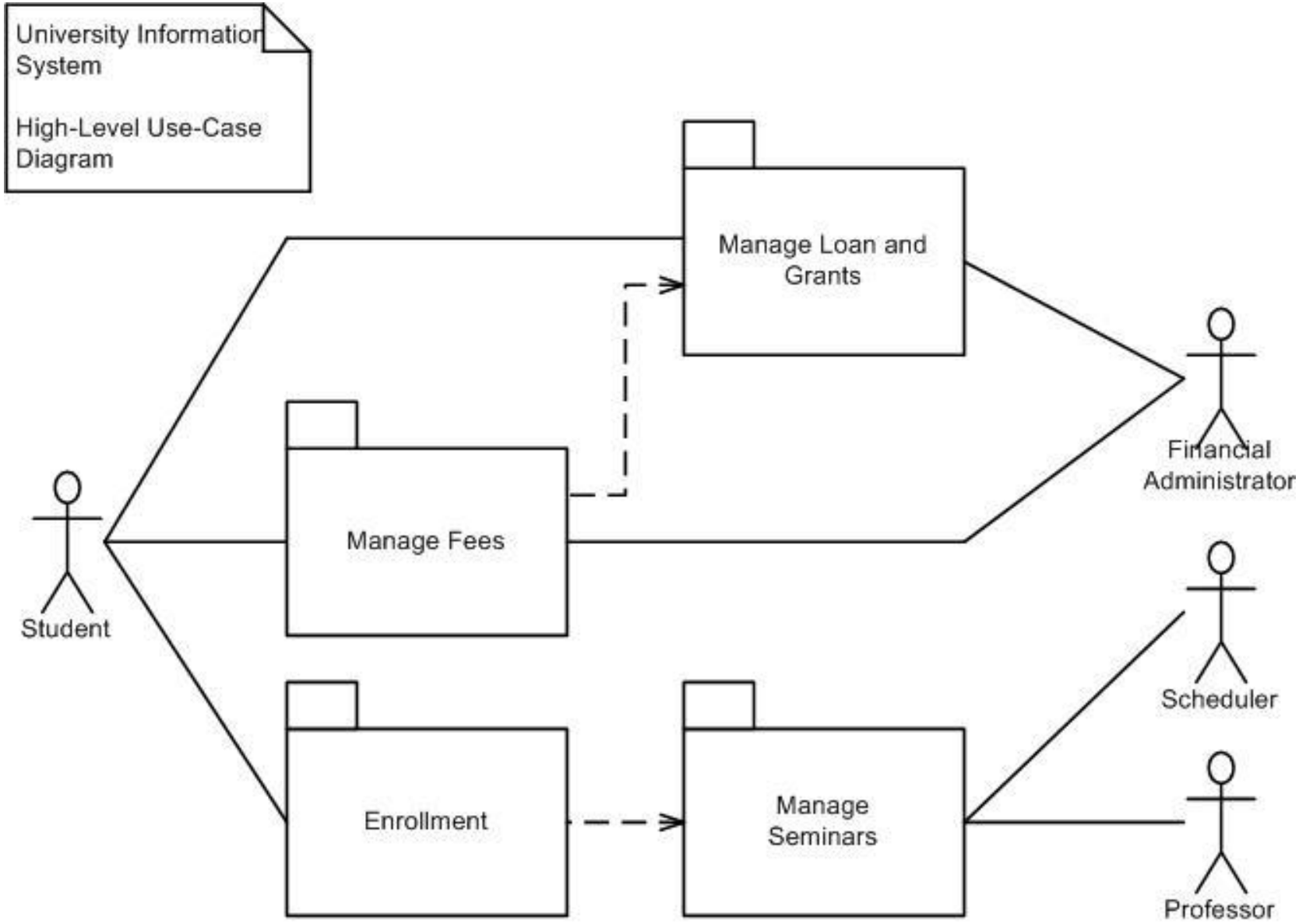
Symbols used in a Package diagram:

Package: Is a logical grouping of UML elements. It is used to simplify UML diagrams by grouping related elements into a single higher-level element.

Dependency Relationship:

Represents a dependency between packages: If a package is changed, the dependent package also could have to be modified.

Has an arrow drawn from the dependent package toward the package on which it is dependent.



9.4 Package Diagrams

Guidelines for Creating Package Diagrams

1. **Set Context** : example- problem domain layer
2. **Cluster Classes** : cluster the classes together into partitions based on the relationships that the classes share. The relationships include generalization, aggregation, the various associations, and the message sending that takes place between the objects in the system.
3. **Create Packages :** example -place the clustered classes together in a partition and model the partitions as packages. In an Appointment System, Appointment package , Patient package etc.
4. **Identify Dependencies :** identify the dependency relationships among the packages.
5. **Draw Diagram :** To increase the understandability of the dependency relationships among the different packages, a pure package diagram that shows only the dependency relationships among the packages can be created.

9.4 Package Diagrams

Verifying and Validating Package Diagrams

- Like all the other models, package diagrams need to be verified and validated.
- The identified packages must make sense from a problem domain point of view.
- All dependency relationships must be based on message-sending relationships on the communications diagram, cell entries in the CRUDE matrix, and associations on the class diagram.

Summary

- The major activity that takes place during design is evolving the set of analysis representations into design representations.
- Design also includes activities such as designing the user interface, system inputs, and system outputs, which involve the ways that the user interacts with the system.
- Physical architecture decisions are made regarding the hardware and soft ware that will be purchased to support the new system and the way that the processing of the system will be organized.
- Design also includes activities such as designing the user interface, system inputs, and system outputs, which involve the ways that the user interacts with the system.

Summary cont..

- Before we evolve our analysis representations into design representations, we need to verify and validate the current set of analysis models to ensure that they faithfully represent the problem domain under consideration.
- We must ensure that the Functional and Structural models are consistent with each other. Eg. The activity diagrams, use-case descriptions, and use-case diagrams must agree with the CRC cards and class diagrams that represent the evolving model of the problem domain.
- We must ensure that the Functional and Behavioral models are consistent with each other. Eg. The activity diagrams, use-case descriptions, and use-case diagrams must agree with the sequence diagrams, communication diagrams, behavioral state machines, and CRUDE matrix.

Summary cont..

- We must also ensure that the two Structural and Behavioral models are consistent with each other.
- After successfully verifying and validating the analysis models, we need to begin evolving them into appropriate design models.
- Factoring, partitions and collaborations, and layers can be used to evolve problem domain-oriented analysis models into optimal solution domain-oriented design models.
- In UML, collaborations, partitions, and layers can be represented by a higher-level construct: a package.
- A package serves the same purpose as a folder on your computer. A *package* is a general construct that can be applied to any of the elements in UML models.

Summary cont..

- Like all the other models, package diagrams need to be verified and validated.
- The identified packages must make sense from a problem domain point of view.
- All dependency relationships must be based on message-sending relationships on the communications diagram, cell entries in the CRUDE matrix, and associations on the class diagram.

*****