Object Oriented System Design

MODULE 4

8.00

Design Issues -

8.30

- UML as a model cannot work in isolation

9.00

- Large scale system design or development involves

9.30

  - team-oriented efforts
  - software architectural design.
  - system design, implementation, and integration

10.00

10.30

11.00

- The unified process by rational is

11.30

  - iterative & incremental
  - use-case driven
  - architecture-centric.

12.00

12.30

13.00

Unified approach to design -

14.00

The unified process is a design framework which guides the tasks, people and products of the design process. It provides the inputs & output of each activity, but does not restrict how each activity must be performed. Different activities can be used in different situations, some left out, some modified or replaced.

14.30

15.00

15.30

16.00

16.30

17.00

There are 4 key elements behind unified process -

17.30

1) Iterative & Incremental : the design process is based on iterations which either address different aspects of the design process or move the design forward in some way.

18.00

Evening

2) Use case driven : it helps identify the primary requirement of the system. it ensure that the evolving design is always relavant to what the user required. Use cases act as consistent thread throughout development process.

Meetings

Things To Do ☐ ☐ ☐

Important Calls ✓ ☐ ☐

8.00 3) Architecture-centric : Unified process prescribes the successive refinement
8.30 of executable architecture, thereby attempting to ensure that the
9.00 architecture remains relavant.
9.30
10.00 4) Acknowledge Risk : the unified process tries to force the riskiest aspects
10.30 of the system to be designed and implemented early on, hence
11.00 ensuring that the risk in the system is addressed & managed in
11.30 professional way.
12.00
12.30 # Partitioning of Analysis Model -
13.00 The purpose of system partitioning is to separate the different types of
13.30 problems that you will need to address in order to create a successful
14.00 software system.
14.30 Its of 2 types -
15.00
15.30 1) Domain Partitioning : it means to identify the different types of function
16.00 that the system must support. Functions should be grouped
16.30 using the principles of cohesion & coupling so that the related functions
17.00 appear together. these functions are defined in use case model
17.30
18.00 2) Technological Partitioning : it identifies and seperates the diff software
Evening responsibilities. Each partition will typically address one type of
functionality such as user interface design, application logic,
and data access. In turn, the nature of each type of
technological functionality will usually imply a diff. type
of programming and use of diff. technology.

Meetings

✓ Things To Do
☐
☐
☐

✓ Important Calls
☐
☐
☐

✓
☐
☐
☐

2016
March
(069-297) Wednesday 9

Concurrency and Subsystem Allocation -

The dynamic aspect of the object-behavior model provides an indication of concurrency among classes. If classes (or subsystems) are not active at the same time, there is no need for concurrent processing. This means that the classes (or subsystems) can be implemented on the same processor hardware.

If classes (or subsystems) must act on events asynchronously and at the same time, they are viewed as concurrent. When subsystems are concurrent, two allocation option exists:

i) Allocate each subsystem to an independent processor.

ii) Allocate the subsystems to the same processor & provide concurrency support through operating system features.

If the flow of events and transition indicates that only a single object is active at any one time, a thread of control has been established and tasks in object oriented system are designed by isolating threads of control.

Task Management Component -

Coad and Yourdon suggest the foll strategy for the design of objects that manage concurrent tasks:

- the characteristics of the task are determined by understanding how the task is initiated.
- A coordinator task and associated objects are defined.
- The co-ordinator and other tasks are integrated.

Event driven & clock driven tasks are encountered & activated by an interrupt. The priority and criticality of the task must also be determined. High criticality tasks must continue to operate even if resource availability is reduced. Once the characteristics of tasks are determined, object attributes and operations required to achieve coordination & communication with other tasks are defined.

Meetings | Things To Do | Important Calls

8.00 # User Interface Component -

8.30 The object oriented analysis model contains usage scenarios (use-cases)
9.00 and a description of the roles that users play (actors) as they interact
9.30 with the system. They serve as input to the user interface design process.
10.00 Once the actor and its usage is defined, a command hierarchy is
10.30 identified. The command hierarchy is refined iteratively until every
11.00 use-case can be implemented by navigating the hierarchy of functions.
11.30 The implementer need only instantiate objects that have appropriate
12.00 characteristics for the problem domain.
12.30

13.00 # The Data Management Component -
13.30 Data management encompasses two distinct areas of concern:
14.00 i) The management of data that is critical to the application itself.
14.30 ii) The creation of an infrastructure for storage & retrieval of objects.
15.00

15.30 In general, data management is designed in layered fashion. the idea is to
16.00 isolate the low-level requirements for manipulating data structures from higher
16.30 level requirements for handling system attributes. the objects required to
17.00 manipulate the database are members of reusable classes. The design
17.30 for data management component includes the design of the attributes &
18.00 operations required to manage objects.
Evening

The Resource Management Component -

A variety of different resources are available to an object oriented system or product; and in many instances, subsystems compete for these resources at the same time.

Regardless of the nature of the resource, the software engineer

Meetings | ✓ Things To Do | ✓ Important Calls | ✓

8.00 should design a control mechanism for it.
8.30 Rambaugh & his colleagues suggest that each resource should be owned
9.00 by a "guardian object." The guardian object is the gatekeeper for the resource,
9.30 controlling access to it and moderating conflicting requests for it.
10.00

10 # Inter-Subsystem Communication -

11.00 Once each subsystem has been specified, it is necessary to define the collaboration
11.30 that exist between the subsystems. The model that we use for object-to-object
12.00 collaboration can be extended to subsystems as whole. Communication can occur
12.30 by establishing a client/server link. We must specify the contract that
13.00 exists between subsystem. The collaboration graph is similar in form to the
13.30 event flow diagram. Each subsystem is represented along with its interactions
14.00 with other subsystems.
14.30

15.00 # Object Description -

15.30 A design description of an object can take two forms:
16.00

16.30 1) A protocol description that establishes the interfaces of an object by defining
17.00 each message that the object can recieve. It is nothing more than a set
17.30 of msges & a corresponding comment for each msg.
18.00

Evening 2) An implementation description that shows implementation details for each
operation implied by a msg that is passed to an object. Implementation
details include information about the object's private part.

Meetings	✓	Things To Do	✓	Important Calls	✓
	☐		☐		☐
	☐		☐		☐
	☐		☐		☐

15 Week 11
March
Tuesday (075-291)

Week	March				
	9	10	11	12	13
Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	31
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

8.00
8.30
9.00
9.30
10.00
10.30
11.00
11.30
12.00
12.30
13.00
13.30
14.00
14.30
15.00
15.30
16.00
16.30
17.00
17.30
18.00
Evening

Data Structures –

Data structures are designed concurrently with algorithms. Since operations invariably manipulate the attributes of a class, the design of data structures that best reflect the attributes will have a strong bearing on the algorithmic design of corresponding operation.

There are 3 types of operations –

1) operations that manipulate data in some way.
2) operations that perform a computation.
3) operations that monitor an object for occurence of controlling event.

Program Components & Interfaces –

An important aspect of software design quality is modularity ; ie. the specification of program components (modules) that are combined to form a complete program.

The object - oriented approach defines the object as a program componene that is itself linked to other components. But defining objects & operations is not enough. we must also identify the interfaces b/w ~~objects~~ objects & overall structure of the objects. Although a program component is a design abstraction, it should be represented in the context of programming language used for implementation. Data objects & corresponding operations are specified for each of the program components.

Meetings	✓	Things To Do	✓	Important Calls	✓
	☐		☐		☐
	☐		☐		☐
	☐		☐		☐

2016
March
(076-290) Wednesday 16

8.00 # Design Patterns –

8.30 They are recurring patterns of classes and communicating objects in
9.00 many object oriented systems. These patterns solve specific design
9.30 problems and make object-oriented design more flexible, elegant, &
10.00 ultimately reusable. They help designers reuse successful designs by
10.30 basing new designs on prior experience.
11.00
11.30 → Characteristics –
12.00 • Generic • well-proven • Simple • Reusable • Object-oriented.
12.30
13.00 → Types –
13.30 1. Creational Patterns – it seperate the operation of an application
14.00 from how its objects are created.
14.30 2. Structural Pattern – it describes how classes & objects can be
15.00 combined to form large structures.
15.30 3. Behavioral Pattern – they are those patterns that are most specifically
16.00 concerned with communication between objects.
16.30
17.00 # Reuse –
17.30 Most of object-oriented design & programming centres around reuse and
18.00 reusable code. Some type of reuse found in software design are –
Evening • integration of off-the-shelf components.
• use of standard and custom class libraries.
• use of design patterns to solve common problems
• use of application framework.

Meetings
✓ Things To Do ✓ Important Calls ✓
☐ ☐ ☐
☐ ☐ ☐
☐ ☐ ☐

APR MAY JUN

17 Week 11
March
Thursday (077-289)

Week	March						
	9	10	11	12	13		
Monday		7	14	21	28		
Tuesday		8	15	22	29		
Wednesday	1	9	16	23	30		
Thursday	2	10	17	24	31		
Friday	3	11	18	25			
Saturday	4	12	19	26			
Sunday	5	13	20	27			
	6						

8.00 # ~~Interaction Diagrams~~ State chart diagrams

8.30

9.00 ~~objects~~ # State Transition Diagrams -

9.30 the basic idea is to define a machine that has a no. of states. The

10.00 machine recieves events from outside world, and each event can cause

10.30 the machine to transition from one state to another. It describe all

11.00 of the states that an object can have, the events under which

11.30 an object changes state, the conditions that must be fulfilled before

12.00 the transitions will occur (guards), and the activities undertaken

12.30 during the life of an object (actions).

13.00

13.30

14.00

14.30

15.00

15.30

16.00

16.30

17.00

17.30

18.00

Evening

Meetings	✓	Things To Do	✓	Important Calls	✓
	☐		☐		☐
	☐		☐		☐
	☐		☐		☐
			☐		