Unit-4 5

Page No.
Date

Class Design

→ During analysis phase the class design is created.

→ The purpose of class design is to complete the definitions of classes, and associations and choose the algorithm for operations.

→ choosing appropriate algorithm for the operation of the classes and breaking the complex operations into simple ones.

→ At each level new operations, attributes and classes are added, relationship is added.

Steps

→ Bridge the gap b/w high level requirement to low level services.

→ Analyze use case

→ for each operation design an algorithm

→ Recurse downwards to design operations.

→ Refactor the model into simpler class design

→ Optimize access path to data.

→ Reify behaviour that must be manipulated

→ Adjust class structure to increase inheritance

→ Arrange classes and associations appropriately.

* Bridging the Gap.

→ The system can offer various services when features are properly utilized by the resources available.

Page No.
Date

But there exists a gap between these desired feature and available resources.

Resources :-> OS, class library, application.

* finding the intermediate element in a framework or class library, select them and fit them together for providing particular services.
* The intermediate can be operations, classes or some other UML construct.
* it required high level operations.

* Realizing Use Cases.

→ uses cases meant for defining behavior of the system.

→ following step for realize the use cases.

1. list the responsibilities of use case or operation
2. Each operation can have various responsibilities
3. Define the operation for each responsibility cluster
4. Assign new lower level operation to classes.

* Designing Algorithms

for operation we have to create Algorithm steps.

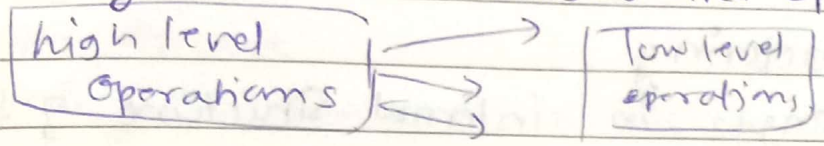
1) Select low cost Algorithm.
2) Select appropriate data structure
3) Define new internal classes & operation if required
4) Assign operation to appropriate class.

Page No.
Date

① Selecting Algorithm
→ Pseudo code is used
→ choose simple & understanble
→ Computational complexity
→ flexibility → (extended further)

② Selecting Data Structure → Array, Stack, Queue priority queue etc.

③ Selecting internal classes and operations.



④ Assigning operation to classes.

* Recursing downward
* top down approach
* The operation at high level layer invoked the operation at low layer.

high level layer
↓
low level layer

→ Recursing downward can be done by

1) functionality layers.
→ high level functions broken into ~~few~~ small function having fewer operation.
+ Danger → if high level functionality broken randomly then the lower level operation can not be related to the classes.

2) Mechanism layer
Building the system to support various much needed Mechanism.

Page No.
Date

* they provide the support to perform high level responsibility.

* In large system functionality layer are mixed with mechanism layer.

eg. updating student data is mechanism.

* Refactoring
- changes in internal structure of software to improve the design without altering the external functionality.
→ rework on classes & operation.
→ time consuming but imp step.

* Design optimization.
→ first get logic correct then optimize
→ focus on critical area
→ less likely to reusable.

(1) Adding Associations for efficient Access.

* we use redundant association for efficient access.

(2) Rearranging Execution order
→ eliminate deadth path
→ narrow the search
→ Invert the execution order for optimization

Page No.
Date

(3) Saving desired value

→ we avoid the recomputation.

- Explicit update
- Periodic - recomputation
- Active value.

* Reification of behavior.

Reification is a Mechanism in which some entity which is not an object is promoted to an object

→ making behavior as a object

eg.

Algorithm → object

So, we can use object in another algorithm, and Modify it.

* Adjustment of inheritance.

(1) Rearrange the classes and operation to increase inheritance.

(a) operations having Special cases.

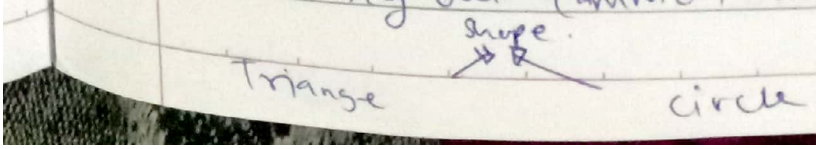
eg Shape → rectang, circle, ellipse.

(b) inconsistent names.

eg. similar attribute but have diff name in diff class, give same name and move to common ancestor.

(c) define only significant operation.

(2) Abstracting out common behaviour.



Page No.
Date

resusability, Extensibility.

(3) using delegation.
Delegation is a Mechanism in which only the useful operation from one object are called and send them to another object.
→ So, there is no danger of inheriting meaningless operations.

* Organizing a class design.

① Information hiding.
→ Internal specification is hidden from outside world
(a) don't access foregin attribute
(b) Define interface
(c) hide External object

(2) Coherence of entities
entities - (class, operation or package)
→ entity should not be a collection of unrelated parts.

(3) fine-tunning package

- class model is partitioned into package.
- Arrange closely related classes in one package and unrelated in diff. package.

* Implementation Modelling.

-> Implementation is a final stage of development
-> programming language is used.
-> design → source code.

Steps
① fine tuning of the classes
② fine tuning generalizations
③ Realizing associations
④ Preparing for testing.

* fine tuning classes.
-> Partition a class: eg Address to Homeaddres and company address
-> Merge class: eg: EmployeeHome + employeeComp → Employee.
-> partition or merge attributes
-> Promote an attribute or demote a class.
we can represent entity to a attributes, classes or several related classes.

* fine-tuning Generalizations.
-> Adding or removing Generalization.

* Realizing association.
-> It provide acess path b/w 2 Objects.

(1) one way association. } uses pointer
(2) two way association.
(3) Implementing association object

* Testing.
  * Identify the bug, and quality assurance
  * Every stage testing should perform.

① Unit testing → modules are checked
② Integration → Merged module are tested
③ System testing → complete integrated System testing
   (a) functional testing
   (b) Non functional testing

* Legacy System

→ older, large complex computer based System.
→ business critical System
→ used for long period. coz too risky to replace.
eg Air traffic control.

Components.
① System hardware
② Supporting software - older OS
③ Application Software
④ Application data.
⑤ Business process.
⑥ Business policies and rules.

Page No.
Date

lagecy System - layered Technology

Business process
App. Software
Support Software
Hardware

* If one layer is changed then other layer also demand change

Technique used to deal with legacy system.

1. Reverse engineering
2. wrapping
3. Maintenance

1. Reverse Engineering.

→ design recovery.
→ data & architectural information is Extracted from source code.
→ from application requirements are extracted
→ less time required.
→ Can be imperfect

inputs to reverse Engineering

1. programming code
2. Data
3. forms and reports
4. Database structure
5. Documentation
6. Application Understanding
7. Test cases.

Page No.
Date

o/p from reverse engineering.

① Models — Class model.

② Mapping — State Mode or interaction Model

③ Logs — records the observation.

* Building the Class model. (reverse Engineering)

① Implementation recovery.

* from application create an initial class Model

② Design recovery

* recover association

③ Analysis recovery

* redundant information is eliminated
* aggregation & Composition is identified
* Packages are Created.

* Building interaction Model.

* necessary to understand behavior of the system

* activity diagram build.
* Sequence diagram build.

* Building State Model.

- from Sequence diagram State Model is Created.
- All States are identified by studying class.

Page No.
Date

Reverse Engineering tips.

① you have to Change the decision Some time
② use flexible process.
③ If more info. present then less judgement can be made by reverse engineer.
④ can't get accurate result
⑤ use consistent style.

2. Wrapping

→ wrapper is collection of interface that control access to the system.
→ It consists of set of boundary class to provide the interface
→ And that Boundary class call the existing System's operation.

3. Maintenance

(1) Initial development - developer create software
(2) Evolution → refactoring, updates
(3) Servicing →
(4) Phaseout → thinking to devise the software
(5) closedown → removed from Market.