OBJECT-ORIENTED SYSTEMS DEVELOPMENT: USING THE UNIFIED MODELING LANGUAGE

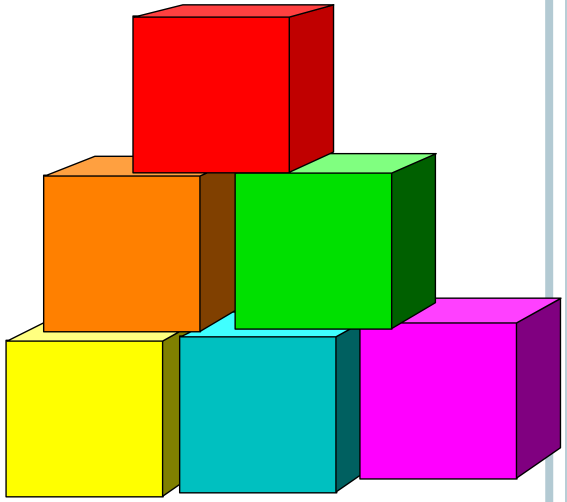
An Overview of Object-Oriented Systems Development

GOALS

- The object-oriented philosophy and why we need to study it.
- The unified approach.

INTRODUCTION

- Object-Oriented (OO) systems development is a way to develop software by building self-contained modules that can be more easily:
- Replaced
- Modified
- and Reused.



WHAT IS A SOFTWARE DEVELOPMENT METHODOLOGY?

- Practices, procedures, and rules used to develop software.

SYSTEMS DEVELOPMENT METHODOLOGIES

- Systems development methodology is a way to develop system.
- A comprehensive system development methodology utilizes sets of tools as well as the style in which they are to be used.

TRADITIONAL SYSTEMS DEVELOPMENT METHODOLOGY

- Traditional or Structured approach is based on the idea that a system can be thought of as a collection of modules or subsystems.
- It is much easier to work with a smaller cohesive module than a complex system.
- A s/w system can be thought of as collection of programs and isolated data OR Algorithms+Data Structures (a set of algorithms performing certain action(s) on certain data)

Two Orthogonal Views of the Software (contd….)

✓ Traditional Development Technique:

1. Views software as collection of programs or functions and isolated data.

   Algorithm + Data Structure = Program

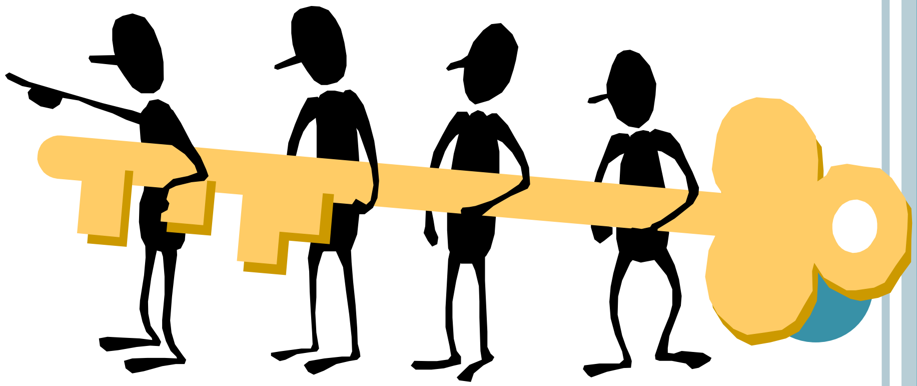
2. It focuses on the functions of the system – What is it doing ?
3. Primary focus is on function.
4. Data is free flowing.

OBJECT-ORIENTED SYSTEMS DEVELOPMENT METHODOLOGY

- In an O-O environment, software is a collection of discrete objects.
- These objects encapsulate their data and functionalities to model real world "objects."
- OOSD is a way to develop software by building self-contained modules or objects that can be easily replaced, modified and reused.

OBJECT-ORIENTED SYSTEMS DEVELOPMENT METHODOLOGY (CON'T)

- An object-oriented life cycle encourages a view of the world as a system of cooperative and collaborating agents.



BENEFITS OF OBJECT ORIENTATION

- Faster development,
- Re-usability,
- Increased quality,
- and easier maintenance.
- More robust & promote greater design

OO BENEFITS (CON'T)

- Reasons why object orientation works:
  i. Higher level of abstraction. At object level)
  ii. Seamless transition among different phases of software development.
  iii. O-O uses same language like UML(Unified Modeling Language) to talk about all phases of software development.
  iv. It reduces complexity, redundancy, & creating a robust system.
  v. Encouragement of good programming technique. &
  vi. Promotion of reusability.

UNIFIED APPROACH

- The *unified approach* (UA) is a methodology for software development that is used in this book.
- The UA, based on methodologies by Booch, Rumbaugh, Jacobson, and others, tries to combine the best practices, processes, and guidelines.

UNIFIED APPROACH (CON'T)

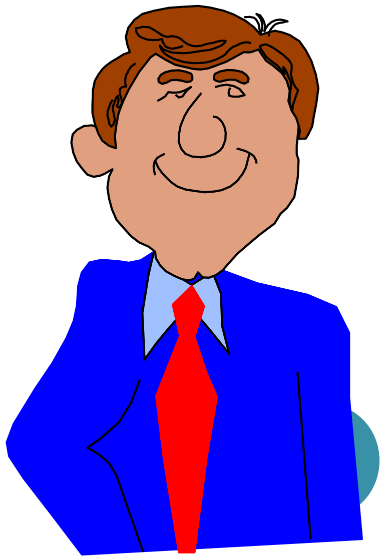
- UA utilizes the *unified modeling language* (UML) which is a set of notations and conventions used to describe and model an application.

LAYERED ARCHITECTURE

- UA also uses a layered architecture to develop applications.
- The layered approach consists of view or user interface, business, and access layers.

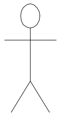
LAYERED ARCHITECTURE (CON'T)

- This approach reduces the interdependence of the user interface, database access, and business control.
- Therefore, it allows for a more robust and flexible system.



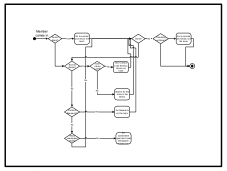
Analysis

1. Identify the users/actors (Chapter 6):
   Who is (or will be) using the system?



2. Develop a simple business process model

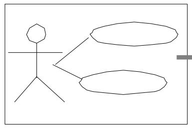
The advantage of developing a business process model is that it familiarizes you with the system and therefore the user requirements



Business process modeling using activity diagram

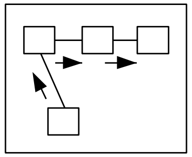
3. Develop the use case (Chapter 6):
What are (or will be) the users are doing with the system?
Use cases provide comprehensive documentation of the system under study

Use cases capture the goal of the users and the responsibility of the system to its users

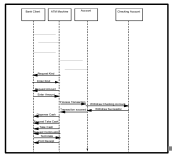


Use case diagrams

4. Interaction diagrams (Chapter 7)
   4.1 Develop sequence diagrams
   4.2 Develop collaboration diagrams.
   4.3 Iterate and refine

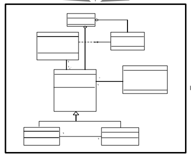


collaboration diagram



Sequence diagram

5. Classification (Chapter 8)
   5.1 Identify Classes
   5.2 Identify Relationships
   5.3 Identify Attributes
   5.4 Identify Methods
   5.5 Iterate and refine.

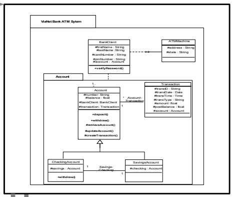


Class diagram

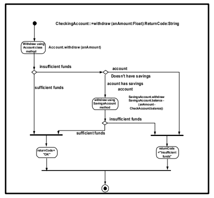
The process of creating sequence or collaboration diagrams is a systematic way to think about how a use case can take place, and by doing so, it forces you to think about objects involves in your application

Design

6. Apply design axioms to design classes, their attributes, methods, associations, structures, and protocols (Chapter 9)
   6.1. Refine and complete the static UML class diagram (object model) by adding details to the UML class diagram (Chapter 10)
      6.1.1 Refine attributes
      6.1.2 Design methods and protocols by utilizing UML activity diagram for representation of method's algorithm
      6.1.3 Refine (if required) associations between classes
      6.1.4 Refine (if required) class hierarchy and design with inheritance
   6.2 Iterate and refine (reapply Design axioms).

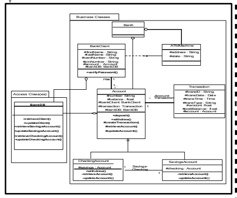


Refine UML Class diagram



Design methods by utilizing UML Activity Diagram

7.0 Design the access ayer (Chapter 11)
   7.1. Create maccess layer classes by mirroring the business classes
   7.2. Define relationships
   7.3. Simplify classes and structures
      7.3.1 Eliminate redundant classes
      7.3.2 Eliminate method classes
   7.4 Iterate and refine

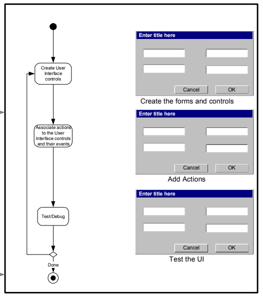


UML Class diagram with added access and view classes

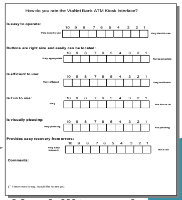
8. Designing view layer classes (Chapter 12)
   8.1 Macro-level UI design Process- Identifying View layer Objects
   8.2 Micro-level UI design activities:
      8.2.1 Designing the view layer objects by applying design axioms and corollaries
      8.2.2 Prototyping the view layer interface.
   8.3. Usability and user satisfaction testing (Chapter 14):
   8.4 Iterate and refine

9. Iterate and refine the design/analysis: If needed repeat the preceding steps

Prototyping and Testing


Create User Interface controls
Associate actions to the User Interface controls and their events
Test/Debug
Done
Enter title here
Cancel
OK
Create the forms and controls
Add Actions
Test the UI

Prototype user interface


How do you rate the ViaNet Bank ATM Kiosk Interface?

Usability and user satisfaction testing

SUMMARY

- In an object-oriented environment, software is a collection of discrete objects that encapsulate their data and the functionality to model real-world objects.
- An object orientation produces systems that are easier to evolve, more flexible, more robust, and more reusable than other traditional approaches.