Object-Oriented Systems Development: Using the Unified Modeling Language

Chapter 3:
Object-Oriented Systems Development Life Cycle
Goals

- The software development process
- Building high-quality software
- Object-oriented systems development
Goals (Con’t)

- Use-case driven systems development
- Prototyping
- Rapid application development
- Component-based development
- Continuous testing and reusability
Software Process

The essence of the software process is the transformation of

• Users’ needs to
• The application domain into
• A software solution.
Software Process (Con't)

What are the uses of the system?

Problem Statements

Transformation 1

Analysis

Transformation 2

Design

Implementation

Transformation 3

System

Software

Product

Transformation 3

Detail

Needs

What are the uses of the system?
Traditional Waterfall Approach to Systems Development

What

How

Do It

Test

Use
Software Quality

- There are two basic approaches to systems testing.
- We can test a system according to how it has been built.
- Alternatively, we can test the system with respect to what it should do.
Quality Measures

- Systems can be evaluated in terms of four quality measures:
  - Correspondence
  - Correctness
  - Verification
  - Validation
Quality Measures (Con’t)

• Correspondence measures how well the delivered system corresponds to the needs of the operational environment.
How would you determine Correspondence?

- It cannot be determined until the system is in place.
Quality Measures (Con't)

- **Correctness** measures the consistency of the product requirements with respect to the design specification.
Quality Measures (Con’t)

- **Verification** - "Am I building the product right?"
- **Validation** - "Am I building the right product?"
Quality Measures (Con’t)

• **Verification** is to predict the correctness.

• **Validation** is to predict the correspondence.
Object-Oriented Systems Development Approach

**O-O Analysis**
- Build Use-Cases
- Object analysis
- Validate/test

**Iteration and Reuse**

**O-O Design**
- Design classes, define attributes and methods
- Build object & dynamic model
- Build UI and prototype
- User satisfaction test, usability test, quality assurance test

**O-O Implementation**
- Using TOOLS CASE and/or OO programming languages
- User satisfaction, Usability & QA Tests
- Build UI and prototype
- User satisfaction test, usability test, quality assurance test

**Build Use-Cases**
- Object analysis
- Validate/test

**Object-Oriented Systems Development Approach**
Object-Oriented Systems Development activities

- Object-oriented analysis.
- Object-oriented design.
- Prototyping.
- Component-based development.
- Incremental testing.
Use case driven systems development

- **Use Case**, is a name for a scenario to describe the user–computer system interaction.
Object-Oriented Analysis

- OO analysis concerns with determining the system requirements and identifying classes and their relationships that make up an application.
Object-Oriented Design

- The goal of object-oriented design (OOD) is to design:
  - The classes identified during the analysis phase,
  - The user interface and
  - Data access.
Object-Oriented Design (Con’t)

• OOD activities include:

  – Design and refine classes.
    • Design and refine attributes.
    • Design and refine methods.
    • Design and refine structures.
    • Design and refine associations.

  – Design User Interface or View layer classes.

  – Design data Access Layer classes.
Prototyping

- A Prototype enables you to fully understand how easy or difficult it will be to implement some of the features of the system.
- It can also give users a chance to comment on the usability and usefulness of the design.
Types of Prototypes

- A *horizontal prototype* is a simulation of the interface.
- A *vertical prototype* is a subset of the system features with complete functionality.
Types of Prototypes (Con't)

• An *analysis prototype* is an aid for exploring the problem domain.
• A *domain prototype* is an aid for the incremental development of the ultimate software solution.
Component-based development (CBD)

- CBD is an industrialized approach to the software development process.
- Application development moves from custom development to assembly of pre-built, pre-tested, reusable software components that operate with each other.
Component-based development (CBD) Con’t

- Component wrapper
  - Legacy programs
- Component wrapper
  - Legacy data
- Component wrapper
  - Legacy screens
- Component wrapper
  - Legacy software packages
Rapid Application Development (RAD)

- RAD is a set of tools and techniques that can be used to build an application faster than typically possible with traditional methods.
• RAD does not replace SDLC but complements it, since it focuses more on process description and can be combined perfectly with the object-oriented approach.
Incremental Testing

• Software development and all of its activities including testing are an iterative process.

• If you wait until after development to test an application for bugs and performance, you could be wasting thousands of dollars and hours of time.
Reusability

- A major benefit of object-oriented systems development is reusability, and this is the most difficult promise to deliver on.
Reuse strategy

- Information hiding (encapsulation).
- Conformance to naming standards.
- Creation and administration of an object repository.
Reuse strategy (Con't)

- Encouragement by strategic management of reuse as opposed to constant redevelopment.
- Establishing targets for a percentage of the objects in the project to be reused (i.e., 50 percent reuse of objects).
Summary

• The essence of the software process is the transformation of users’ needs into a software solution.

• The O-O SDLC is an iterative process and is divided into analysis, design, prototyping/implementation, and testing.