Nadar Saraswathi College of Engineering & Technology

NSCET E-LEARNING PRESENTATION

LISTEN … LEARN… LEAD…

Nadar Saraswathi College of Engineering & Technology

COMPUTER SCIENCE AND ENGINEERING

III YEAR / V SEMESTER

CS8592 OBJECT ORIENTED ANALYSIS AND DESIGN

A.DURAIMURUGAN , M.E, (Ph.d)

ASSISTANT PROFESSOR

Nadar Saraswathi College of Engineering & Technology,

Vadapudupatti, Annanji (po), Theni – 625531.

Nadar Saraswathi College of Engineering & Technology

UNIT V

TESTING

Nadar Saraswathi College of
Engineering & Technology

OBJECT ORIENTED MEHODOLOGIES

- The Essentials Many methodologies are available to choose from for the system development. Each methodology is based on modeling the business problem and implementation in an object oriented fashion. The Rumbaugh et al method has a strong method for producing object models. Jacobson et al have a strong method for producing user-driven requirement and object oriented analysis model. Booch has a strong method for producing detailed object oriented design models

GRASP

- Rumbaugh's Object Modeling Technique: • Describes the dynamic behavior of objects in a system using the OMT dynamic model. • Four phases. ◦ Analysis – results are objects, dynamic and functional models. System ◦ design – gives a structure of the basic architecture. ◦ Object design – produces a design document. ◦ Implementation – produces reusable code. • OMT separates modeling in to three different parts • Object Model – presented by object model and the data dictionary. • Dynamic model - presented by the state diagrams and event

GRASP

- Object Model:- Object model describes the structure of objects in a system, their identity and relationships to other objects, attributes, and operations. The object model is represented graphically with an object diagram.
- The OMT Dynamic Model: • OMT provides a detailed and comprehensive dynamic model • The OMT state transition diagram is a network of states and events. ◦ Each state receives one or more events, at which it makes the transition to the next state.

Sequence Diagram Notations

State transition diagrams for the bank application user interface. • Round boxes represents states • Arrows represents transitions

The OMT Functional Model • The OMT data flow diagram (DFD) shows the flow of data between different processing in a business. • Data Flow Diagrams use four primary symbols: ◦ The process is any function being performed ◦ The data flow shows the direction of data element movement.

The Booch Methodology :

• Booch methodology is a widely used object-oriented

method that helps to design your system using the object paradigm.

• Is criticized for his large set of symbols.

• It consists of the following diagrams: ◦ Class diagrams. Object diagrams. ◦ State transition diagrams. ◦ Module diagrams. Process diagrams. ◦ Interaction diagrams.

• Two processes: ◦ Macro development

process ◦ Micro development process.

The Booch Methodology :

- Booch methodology is a widely used object-oriented

method that helps to design your system using the object paradigm.

- Is criticized for his large set of symbols.

- Macro development process. Primary concern – technical management of the system. Steps involved: • Conceptualization. Establish the core requirements and develop a prototype. • Analysis and development of the model Use the class diagram to describe the roles and responsibilities of objects. Use the object diagram to describe the desired behavior of the system.

• Design or create the system architecture. Use the class diagram to decide what classes exist and how they relate to each other, the object diagram to decide what mechanisms are used, the module diagram to map out where each class and object should be declared, and the process diagram to determine to which processor to allocate a process. • Evolution or implementation. Refine the system through much iteration.

- Micro development process. • The micro development process is a description of the day-to-day activities. • Steps involved: ◦ Identify classes and objects. ◦ Identify classes and object semantics. ◦ Identify classes and object relationships. ◦ Identify classes and object interfaces and implementation.

Jacobson Methodology: • Object-oriented business engineering (OOBE) • Object-oriented software engineering (OOSE) • It covers the entire life cycle • Stress traceability(enables reuse of analysis and design work) both forward and backward • Use cases • Scenarios for understanding system requirements. • Non formal text with no clear flow of events. • Text easy to read. • Formal style using pseudo code. • Can be viewed as concrete or abstract (not initiated by actors). • Understanding system requirements

REFERENCE ATTRIBUTES

Object oriented software Engineering: Objectory

- OOSE is also called Objectory
- Development process is also called as use case driven development
- The system development method based on OOSE is a process for the industrialized development of the s/w

- Use case model:- The use-case model defines the outside and inside of the system behaviour • Domain Object Model:- The objects of the real worls are mapped in to the domain object model • Analysis Object Model:- The analysis object model presents how the source code should be carried out written • Implementation model: The implementation model represents the implementation of the system • Test model:- The test model constitutes the test plans, specification and reports.

Object oriented business Engineering • Analysis phase:- ◦ The analysis phase defines the system to be built in terms of the problem-domain object model the requirements model analysis model • Design and Implementation phase:- ◦ The implementation environment must be identified for the design model.

ROLE NAMES

- Patterns and the Various Pattern Templates: • Pattern – ◦ Identifies a common structure that makes it useful for design. ◦ provide common vocabulary ◦ provide“shorthand” for effectively communicating complex principles ◦ help document software architecture ◦ capture essential parts of a design in compact form ◦ show more than one solution ◦ describe software abstractions • Patterns do not... ◦ provide an exact solution ◦ solve all design problems ◦ only apply for object-oriented design

Properties of a good pattern: ◦ It solves a problem:- Patterns capture solutions not just abstract principles or strategies ◦ It is a proven concept:- Patterns capture solutions with a track record, not theory or speculation ◦ Solution is not obvious:- The best patterns generate a solution to a problem indirectly ◦ Describes a relationship:- Patterns do not just describes modules but describes deeper system structures and mechanisms ◦ Has significant human component:- All software serves human comfort or quality of life

Non generative patterns – ◦ ogenerate systems or pats of systems ◦ operspective and active.Patterns templates. ◦ Name:- ⍰ A meaningful name allows us to use a single word or short phrase to refer to the pattern and the knowledge and structure it describes. ⍰ Some pattern forms also provide a classification of the pattern in addition to its name. ◦ Problem:- ⍰ A statement of the problem that describes its intent: the goals and objectives it wants to reach within the given context and forces. ⍰ The forces oppose these objectives as well as each other.

Why Collaboration Diagram?

- Context:- ▯ The preconditions under which the problem and its solution seem to recur and for which the solution is desirable. ▯ It can be thought of as the initial configuration of the system before the pattern is applied to it. ◦ Forces:- ▯ A description of the relevant forces and constraints and how they interact or with one another and with the goals that the user wish to achieve. ▯ A concrete scenario that serves as the motivation for the pattern frequently is employed. ◦ Solution:- ▯ Static relationships and dynamic rules describing how to realize the desired outcome. ▯ It describes how to construct the necessary products.

- Related patterns:- ◦ The static and dynamic relationships between this pattern and others within the same pattern language or system. ◦ Related pattern often share common forces. ◦ They also frequently have an initial or resulting context that is compatible with the resulting or initial context of another pattern. • Known uses:- ◦ The known occurrences of the pattern and its application within existing systems. This helps to validate a pattern by verifying that it indeed is a proven solution to the recurring problem.

- FRAMEWORKS: Frameworks are a way of delivering application development patterns to support best practice sharing during application development . A frame work is a way of presenting a generic solution to a problem that can be applied to all levels in a development. Several design patterns in fact a framework can be viewed as the implementation of a system of design patterns. The major differences between design patterns and frameworks as follows

- The Unified Approach: • Establishes a unifying and unitary framework by utilizing UML. • The processes are: ◦ Use case driven development. ◦ Object oriented analysis. ◦ Object oriented design. ◦ Incremental development and prototyping ◦ Continuous testing.

- UML:- Unified Modeling approach is used for modeling ◦ Layered approach:- ◦ Repository:- Repository for object-oriented system development patterns and frameworks ◦ CBD:- Component based development The Unified Approach allows iterative development by allowing to go back and forth between the design and the modeling or analysis phase. It makes backtracking very easy and departs from the linear waterfall process, which allows no form of backtracking

QUALITY ASSURANCE TESTS

- Debugging is a process of finding out where something went wrong and correcting the code to eliminate the errors or bugs that cause unexpected results. • Kinds of errors: In general, a software has three types of errors such as below 1) Language (syntax) errors are result of incorrectly constructed code, such as an incorrectly typed keyword or punctuations. They are easiest error to be detected on simple running system 2) Run–time errors are detected on running, when a statement attempts an operation that is impossible to carry out.

Testing Strategies

- The objective of s/w testing is to uncover errors. The various testing strategies constitutes – ◦ Unit Testing – Black Box testing, White black testing ◦ Integration Testing – Top–down testing, Bottom–up testing, Regression testing

- **Integration Testing**: It is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. The object is to take unit tested modules and build a program structure that has been dictated by design. It has again 3 testing patterns: Top–down testing, Bottom – up testing and regression testing
- **Top–down testing**: It assumes that main logic or object interactions and system messages of application need more testing than an individual object's method or supporting logic. This strategy can detect serious flaws early in implementation • Bottom–Up testing: It starts with details of system and proceeds to higher levels by a progressive aggregation of details until they collectively fit requirements for system

- **Regression testing**: It is activity that helps ensure that changes (due to testing or other reasons) do not introduce unintended behavior or additional errors. The regression test suite contain three different classes of test cases: ◦ A representative sample of tests that will exercise all s/w functions ◦ Additional tests that focus on s/w functions that are likely to be affected by change ◦ Tests that focus on s/w components that have been changed
- **Validation Testing**: After integration testing, s/w is completely assembled as a package: interfacing errors have been uncovered and corrected and final series of s/w tests – validating testing – begins. It is nothing but continuous execution on s/w by number of times & users. It is again implemented either by one of two methods: Alpha testing or Beta testing

- **Alpha test** is conducted in controlled environment at developer's site by a customer. The s/w is used in setting with developer "looking over shoulder" of user, recording errors & usage problems
- **Beta test** is conducted at one or more customer sites by end user(s) of the s/w. Its live application of s/w where customer records all problems that are encountered during beta testing and reports
- **System Testing**: It is a series of different tests whose primary purpose is to fully exercise the computer – based system. Although each test has a different purpose, all work to verify that.

- **Recovery testing** is a system test that forces s/w to fail in a variety of ways and verifies that recovery is properly performed. If recovery is automatic, re – initialization, checkpoint mechanisms, data recovery and restart are each evaluated for correctness
- **Security testing** attempts to verify that protection mechanisms built into a system will in fact protect it from improper penetration. During this test, tester plays the role(s) of individual who desires to penetrate system. The role of system designer is to make penetration cost greater than value of information that will be obtained
- **Stress testing** executes a system in a manner that demands resources in abnormal quantity, frequency or volume. A variation of stress testing is a technique called sensitivity testing.