Software Development Methodologies

Lecturer: Raman Ramsin

Lecture 3

Seminal Object-Oriented Methodologies:
A Feature-Focused Review (Part 1)
Coad-Yourdon

- Two-phase introduction: Object-Oriented Analysis (OOA) in 1989, and Object-Oriented Design (OOD) in 1991

- Process: The Baseball Model

[Coad and Yourdon 1991]
Coad-Yourdon: OOA

The analysis (OOA) phase of the methodology consists of five principal activities:

- Finding “Classes” (abstract classes) and “Class-&-Objects” (concrete classes)
- Identifying “Structures” (generalization-specialization and whole-part relationships between classes)
- Identifying “Subjects” (partitions/subsystems)
- Defining attributes, and “Instance-Connections” (association relationships between classes)
- Defining “Services” (class operations) and “Message-Connections” (invocations of operations)
Coad-Yourdon: Main Model

- OOA/OOD Results are reflected in a special *Class-&-Object Diagram* (later complemented by behavioural models); the diagram consists of five layers:

  - **Subject** layer: showing the overall partitions of the system
  
  - **Class-&-Object** layer: showing the abstract and concrete classes of the system
  
  - **Structure** layer: showing the generalization-specification and whole-part relationships between the classes
  
  - **Attribute** layer: showing the attributes of the classes and the association relationships between classes.
  
  - **Service** layer: showing the operations of the classes and the potential message-passing between the objects
Coad-Yourdon: Class-&-Object Diagram
Coad-Yourdon: OOD

During the design phase of the methodology (OOD) the system is designed in four components:

- **Problem Domain Component (PDC):** initially contains the results of the analysis phase. During OOD, it is improved and enriched with implementation detail.

- **Human Interaction Component (HIC):** handles sending and receiving messages to and from the user.

- **Task Management Component (TMC):** for systems needing to implement multiple threads of control; defines the multi-task processing classes, coordinates the tasks (processes) and provides means for inter-task communication.

- **Data Management Component (DMC):** provides the infrastructure to store and retrieve objects; may be a simple file system, a relational database management system, or even an object-oriented database management system.
Coad-Yourdon: Design Components

PDC

Sensor
Model
InitSequence
Conversion
Interval
Address
Threshold
Value
Initialise
Sample
MonitorForAlarmCondition

CriticalSensor
Tolerance
Sample

Building
StreetAddress
EmergencyNumber

AlarmEvent
DateTime
Severity
TimeToRepair
Status

AlarmDevice
Duration
Status
Activate

HIC

SensorWindow
Coordinates

SensorAlarmWindow
DisplayMenu
InvokeMenuAction

SensorAlarmItem
invokeItem

SensorStatusWindow
DisplayMenu
InvokeMenuAction

SensorGraphicItem
invokeItem

Task
ID
Name
Description
Priority
ServiceIncluded
CoordinatesBy
CommunicatesVia
Initialise
Start
Standby
terminate

TMC

TaskCoordinator
Coordinate

DMC

Customer
customerId
date
creditLevel
store()
retrive()
Responsibility-Driven Design (RDD)

- Introduced in 1990; a UML-based and use-case driven version was released in 2002.

- The process starts when a detailed requirements specification of the system has already been provided.

- The system is modeled as a collection of objects that collaborate to fulfill their responsibilities.

- Responsibilities include two key items:
  - The knowledge an object maintains.
  - The actions an object can perform.
RDD: Process

- The process is divided into two phases:
  - **Exploratory Phase**: classes and their responsibilities and collaborations are identified.
  - **Analysis Phase**: subsystems and class details are specified.

[Wirfs-Brock et al. 1990]
RDD: Exploratory Phase

- The major tasks to be iteratively performed in this phase are:
  - Discovering the *classes* required to model the application
  - Determining what behavior the system is responsible for and assign these *responsibilities* to specific classes
  - Determining what *collaborations* must occur between classes of objects to fulfill the responsibilities.

- Results are modeled in CRC (Class-Responsibility-Collaborator) cards

**CRC Card**

<table>
<thead>
<tr>
<th>Class: name of class</th>
<th>(Abstract or Concrete)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Superclasses:</strong> list of superclasses</td>
<td></td>
</tr>
<tr>
<td><strong>Subclasses:</strong> list of subclasses</td>
<td></td>
</tr>
<tr>
<td><strong>Responsibilities:</strong></td>
<td><strong>Collaborations:</strong></td>
</tr>
</tbody>
</table>

[Wirfs-Brock et al. 1990]
RDD: Analysis Phase

- Major tasks to be performed in this phase:
  - Factoring the responsibilities into inheritance hierarchies modeled in *Inheritance Graphs*, and identifying class contracts
  - Identifying possible subsystems of objects and modeling them and their client-server relationships in *Collaboration Graphs*
  - Determining *Class Protocols* (method signatures) and completing a specification of classes, subsystems of classes, and client-server contracts
RDD: Analysis Phase

Collaboration Graph

Spreadsheet Program

Spreadsheet Editor

Spreadsheet Subsystem

Spreadsheet

File

Spreadsheet Subsystem

Value

Numeric Value
Computed Value
Textual Value

Expression

Constant Expression
Call Reference Expression
Binary Expression

Class name
A CRC class

Contract with contract number

Name of subsystem
A CRC subsystem

Collaboration

[Wirfs-Brock et al. 1990]
Booch

- Introduced, purely as a design method, in 1991 [Boo91]; extended version, which also covered analysis, was introduced in 1994.

- Process designed as a repeating process (referred to as “The Micro Process”) within a lifecycle-level repeating process (referred to as “The Macro Process”).

- The macro process serves as a controlling framework for the micro process.
Booch: Macro Process

- Represents the activities of the development team on the scale of weeks to months.
- Many parts of this process are basic software management practices such as quality assurance, code walkthroughs, and documentation.
- The focus at this level is more upon the customers and their desires for things such as quality, completeness, and scheduling.

[Booch 1994]
Booch: Micro Process

- Driven by scenarios and architectural specifications that emerge from the macro process
- Represents the daily activities of the individual or small group of developers

[Booch 1994]
Object Modeling Technique (OMT)

- Introduced by Rumbaugh et al. in 1991

- Categorized as *combinative* [MP92], since it is based on three different models and defines a method for integrating them.

- **Object Model (OM)**: depicts object classes in the system and their relationships, as well as their attributes and operations, in a *Class Diagram*

- **Dynamic Model (DM)**: indicates the dynamics of the objects, their changes in state and the flow of events; captured in *Event-Trace Diagrams* and *State Transition Diagrams (State Charts)*

- **Functional Model (FM)**: a hierarchical set of *Data Flow Diagrams (DFDs)* of the system
OMT Process

Knowledge about application domain
Problem statement

Init. OM
Init. DM
Init. FM

Analysis

System Design

Basic architecture
OM, DM, FM

Class libraries

Object Design

Detailed OM, DM, FM

Object source code

Coding

Source code

Testing

Test scenarios

Reuse Database of OMT Specifications

[OMT] Rumbaugh et al. 1991
OMT Process: Analysis Phase

- The goal is to build a correct and comprehensible model of the real world.

- Once the initial problem is defined, the following tasks are carried out:
  - Building the object model, including a *Class Diagram* and a *Data Dictionary*.
  - Developing the dynamic model, including *State Transition Diagrams* and global *Event-Trace Diagrams*.
  - Constructing the functional model including *Data Flow Diagrams* and constraints.
  - Verifying, iterating, and refining the three models.
OMT Process: Analysis Phase

Class Diagram

Person
- name
- address
- social security no.
- charge-time
- earn-salary

Worker
- Works-on

Manager
- Manages

Project
- product name
- budget
- priority

Company
- name
- address
- phone number
- primary product
- hire
- fire
- dept name

Department
- Manufactures

Product
- product name
- budget
- priority

[Rumbaugh et al. 1991]
OMT Process: Analysis Phase

Event-Trace Diagram

- Caller
  - caller lifts receiver
  - dial tone begins
  - dials (4)
  - dial tone ends
  - dials digit (5)
  - dials digit (3)
  - dials digit (7)
  - ringing tone
- Phone Line
  - phone rings
  - answers phone
  - ringing stops
  - tone stops
  - phones connected
- Callee
  - phones connected
  - callee hangs up
  - connection broken
  - connection broken
  - caller hangs up

[Rumbaugh et al. 1991]
OMT Process: Analysis Phase

State Transition Diagram

[Diagram showing state transitions between Idle, Dial tone, Dialing, Time-out, Recorded message, Connecting, Ringing, Connected, and Disconnected states with transitions labeled as on-hook, off-hook, digit(n), time-out, number busy, valid number, invalid number, trunk busy, routed, called phone answers, and called phone hangs up.]

[Rumbaugh et al. 1991]
Data Flow Diagram

- Icon definitions
- Window
  - size
  - location
- expand into vectors
  - application vector list
- clip vectors
  - window vector list
- offset vectors
  - screen vector list
- Screen buffer
  - pixel operations
  - convert to pixels

[Rumbaugh et al. 1991]
OMT Process: System Design

- High-level structure of the system is defined

- Tasks include:
  - Organizing the system into subsystems
  - Identifying concurrency
  - Allocating subsystems to processors and tasks
  - Choosing the strategy for implementing data stores in terms of data structures, files, and databases
  - Identifying global resources and determining mechanisms for controlling access to them
  - Choosing an approach to implementing software control
  - Considering boundary conditions
  - Establishing trade-off priorities
OMT Process: Object Design

- Concerned with fully specifying the existing and remaining classes, associations, attributes, and operations necessary for implementing the system.

- Operations and data structures are fully defined along with any internal objects needed for implementation.

- All details for fully determining how the system will be implemented are specified.
References


