Object Oriented Software Engineering

Ali Kamandi
Sharif University of Technology
kamandi@ce.sharif.edu
Spring 2007
Outline

- Historical aspects
- Economic aspects
- The Object-Oriented Process Model
- Strengths of the Object-Oriented Paradigm
- Basic Principles of Object Orientation
- Basic Concepts of Object Orientation
- Strengths of Object Orientation
- OO Project Metrics
1. Historical Aspects

- OO approach was first proposed in the late 1960s.

- **Aim:** To solve the *software crisis*

- **Software is delivered**
  - Late
  - Over budget
  - With residual faults
Introduction to Object-Oriented

- We live in a world of objects.
- They can be categorized, described, organized, combined, manipulated, and created.
- Object-Oriented view is an abstraction that models the world in ways that help us to better understand and navigate it.
- As time passes, object technologies are replacing classical software development approaches. Why?
- The answer is not simple. Object technologies lead to reuse, OO software is easier to maintain, to adapt, and to scale.
Standish Group Data

- Data on 28,000 projects completed in 2000

- Canceled 23%
- Successful 28%
- Completed late, over budget, and/or with features missing 49%
Cutter Consortium Data

- 2002 survey of information technology organizations
  - 78% have been involved in disputes ending in litigation
Cutter Consortium Data

- For the organizations that entered into litigation:
  - In 67% of the disputes, the functionality of the information system as delivered did not meet up to the claims of the developers
  - In 56% of the disputes, the promised delivery date slipped several times
  - In 45% of the disputes, the defects were so severe that the information system was unusable
Mismanagement of a project leads to:

- Producing the wrong product.
- Producing a product of inferior quality.
- Being late.
- Working 80 hour weeks.
The Delivery Date is Frozen
The Spec is Never Frozen

The Spec V1
The Spec V2
The Spec V3
The Spec VN.1
The Spec VN.2
The Waterfall Model

- Managing the Development of Large Software Systems
  - Dr. Winston W. Royce — 1970

1 May
1 Jul
1 Sep
1 Nov

Analysis
Design
Implementation

DFD
ERD
DL
ST

2006
Code Waste

- In a study of 400 waterfall projects:
  - Only 5% - 15% of the code was ever used.
Long Projects Fail.

Project Success. 23,000 projects

Percent success

Months

0 5 10 15 20 25 30 35 40 45

6 9 12 18 24 36
Without data, all you can manage is:

The Date!
Without data managers can:

This project will be done on time!

Or HEADS will ROLL!
OR....

You are great.
I have faith in you.
I know you can do it!

I sure hope you can.
In 1995, the International Standards Organization and International Electrotechnical Commission defined maintenance *operationally*.

Maintenance is nowadays defined as the process that occurs when a software artifact is modified because of a problem or because of a need for improvement or adaptation.
Modern Maintenance Definition (contd)

- In terms of the ISO/IEC definition
  - Maintenance occurs whenever software is modified
  - Regardless of whether this takes place before or after installation of the software product

- The ISO/IEC definition has also been adopted by IEEE and EIA
Time (= Cost) of Postdelivery Maintenance

(a) Between 1976 and 1981
(b) Between 1992 and 1998
The Costs of the Classical Phases

- Surprisingly, the costs of the classical phases have hardly changed.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Various Projects between 1976 and 1981</th>
<th>132 More Recent Hewlett-Packard Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements and analysis</td>
<td>21%</td>
<td>18%</td>
</tr>
<tr>
<td>(specification) phases</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design phase</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>Implementation phase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding (including unit testing)</td>
<td>36</td>
<td>34</td>
</tr>
<tr>
<td>Integration</td>
<td>24</td>
<td>29</td>
</tr>
</tbody>
</table>
Consequence of Relative Costs of Phases

- Reducing the coding cost by 10% yields at most a 0.85% reduction in total costs
  - Consider the expenses and disruption incurred

- Reducing post-delivery maintenance cost by 10% yields a 7.5% reduction in overall costs
The Object-Oriented Paradigm

- The structured paradigm was successful initially
  - It started to fail with larger products (> 50,000 LOC)

- Postdelivery maintenance problems (today, 70 to 80 percent of total effort)

- Reason: Structured methods are
  - Action oriented (e.g., finite state machines, data flow diagrams); or
  - Data oriented (e.g., entity-relationship diagrams, Jackson’s method);
  - But not both
Conclusion

- The software crisis has not been solved
Informally, an object represents an entity, either physical, conceptual, or software.

- Physical entity
- Conceptual entity
- Software entity
Representing Objects

- An object is represented as rectangles with underlined names

: Professor

ProfessorClark

ProfessorClark : Professor

Class Name Only

Object Name Only

Professor Clark

Class and Object Name

a + b = 10
Object Definition

- An *abstraction* of something in problem domain, reflecting the capabilities of the *system* to keep information about it, interact with it, or both.

[Coad & Yourdon 1990]
Object Definition

- We define an object as a concept, abstraction, or thing with crisp boundaries and meaning for the problem in hand. Objects serve two purposes: They promote understanding of the real world and provide a practical basis for computer implementation.

[Rumbaugh 1991]
Object Definition

- Object has state, behavior and identity.

[Booch 1994]
Object Definition

- Objects are deliberately characterized as if each were a person, with a role in the system that is based on its answer to these questions:
  - What am I?
  - What can I do?
  - What do I know?

[Wirfs-Brock 1990]
What is a Class?

- A class is a description of a group of objects with common properties (attributes), behavior (operations), relationships, and semantics.
  - An object is an instance of a class.

- A class is an abstraction in that it:
  - Emphasizes relevant characteristics
  - Suppresses other characteristics

*OO Principle: Abstraction*
Sample Class

Class
Course

Properties
- Name
- Location
- Days offered
- Credit hours
- Start time
- End time

Behavior
- Add a student
- Delete a student
- Get course roster
- Determine if it is full

a + b = 10
Class Compartments

- A class is comprised of three sections
  - The first section contains the class name
  - The second section shows the structure (attributes)
  - The third section shows the behavior (operations)

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>emplID</td>
</tr>
<tr>
<td>Operations</td>
<td>create()</td>
</tr>
<tr>
<td></td>
<td>save()</td>
</tr>
<tr>
<td></td>
<td>delete()</td>
</tr>
<tr>
<td></td>
<td>change()</td>
</tr>
</tbody>
</table>
What is an Attribute?

Class: CourseOffering
- number
- startTime
- endTime

Object: CourseOffering
- number = 101
- startTime = 900
- endTime = 1100

Object: CourseOffering
- number = 104
- startTime = 1300
- endTime = 1500
What is an Operation?

<table>
<thead>
<tr>
<th>Class</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CourseOffering</td>
<td>addStudent</td>
</tr>
<tr>
<td></td>
<td>deleteStudent</td>
</tr>
<tr>
<td></td>
<td>getStartTime</td>
</tr>
<tr>
<td></td>
<td>getEndTime</td>
</tr>
</tbody>
</table>
Elements of UML Class Diagrams

- **Classes**
  - Three levels of detail
    - Name only
    - Name and Attributes
    - Name, Attributes and Operations

- **Objects**

<table>
<thead>
<tr>
<th>ClassName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute1 : type</td>
</tr>
<tr>
<td>Attribute2 : type</td>
</tr>
<tr>
<td>Operation1()</td>
</tr>
<tr>
<td>Operation2()</td>
</tr>
</tbody>
</table>

Object : ClassName
Associations

- Connection between objects
- Solid line
  - Direction optional
  - Arrowhead depends on meaning
- Name (optional)
- Multiplicities (optional)
- Visibility (optional)
Aggregation and Composition

- **Aggregation**
  - Aggregate end (i.e. whole part) has open diamond arrowhead
  - Usually translates into “pointer to object”

- **Composition**
  - Aggregation with limitations
    - Part can only exist in one object
    - Composite object owns part creation and destruction
  - Composite end has filled diamond arrowhead
Aggregation and Composition

Example
Generalization

- Models the concept of inheritance
- Subclasses inherit the attributes and operations of all their superclasses
Generalization Example

```
Shape
  +draw()

Rectangle
  +draw()

Triangle
  +draw()

Circle
  +draw()

Square
  +draw()
```
Putting the Pieces Together

Order
- dateReceived : Date
- isPrepaid : Boolean
- number : Integer
- total : Currency
+ close() : void
+ dispatch() : void

{ if Order.Customer.GetCreditRating() is "poor", then Order.isPrepaid must be true }

Customer
- name : String
- address : Address
- CreditRating : String
+ GetCreditRating() : String{query,sequential}

CorporateCustomer
- ContactName : String
- CreditLimit : Currency
+ Remind() : void
+ BillForMonth() : void

PersonalCustomer
- CreditCardNumber : String

Employee

OrderLine
- isSatisfied : Boolean
- price : Currency
- quantity : Integer

Product

line items

1

1

0..1 sales rep
Association Classes

- Simple associations have
  - Roles
  - Cardinality
  - Navigability
  - Constraints

- When the association between classes has attributes and behavior of its own, it can become a class
Association Class Example

- **Library Member**
  - 0..*
  - **Rental Contract**
    - due : Date
    - fee : Money

- **Rentable Item**
  - 1

- **Media Library**
  - 1

- **Book**
- **DVD**
- **CD**
- **Video**
2. Economic Aspects

- Coding method $CM_{\text{new}}$ is 10 percent faster than currently used method $CM_{\text{old}}$. Should it be used?

- Common sense answer
  - Of course!

- Software Engineering answer
  - Consider the cost of training
  - Consider the impact of introducing a new technology
  - Consider the effect of $CM_{\text{new}}$ on maintenance
For many years, the term OO was used to denote a software development approach that used one of a number of OO programming languages (e.g. Ada 95, C++, Eiffel, Smalltalk).

Today, the OO paradigm encompasses a complete view of software engineering.

Software engineers and their managers must consider OORA, OOD, OODA(domain anal.), OODBMS, OOCASE.

Why should OO be any different?
Although any one of process models, discussed in Chapter 2, could be adapted for use with OO, the best choice would be an evolutionary process model.

It would be exceedingly difficult to define all necessary classes for a major system or product in a single iteration.

As the OO analysis and design models evolve, the need for additional classes becomes apparent.
An Evolutionary (Spiral) Model
The Component-Based Process Model
The OO Process Model

- Customer Communication
- Planning
- Risk Analysis
- Engineering, Construction & Release

- identify candidate classes
- construct nth iteration of system
- put new classes in library
- extract classes if available
- engineer classes if unavailable

- look-up classes in library

- OO analysis
- OO design
- OO programming
- OO testing
The OO Process Model: RUP
Basic Principles of Object Orientation

Object Orientation

- Encapsulation
- Inheritance
- Polymorphism
What is Encapsulation?

- Hide implementation from clients
  - Clients depend on interface

How does an object encapsulate? What does it encapsulate?
Encapsulation/Hiding

The object encapsulates both data and the logical procedures required to manipulate the data.

Achieves “information hiding”
Example: Single Inheritance

- One class inherits from another

Account
- balance
- name
- number

Withdraw()
CreateStatement()

Superclass (parent)

Generalization Relationship

Descendents

Subclasses
Use multiple inheritance only when needed, and always with caution!
What Gets Inherited?

- A subclass inherits its parent’s attributes, operations, and relationships.

- A subclass may:
  - Add additional attributes, operations, relationships
  - Redefine inherited operations (use caution!)

- Common attributes, operations, and/or relationships are shown at the highest applicable level in the hierarchy.
Example: What Gets Inherited

Superclass (parent)

Subclass

GroundVehicle
  weight
  licenseNumber
  register()

Person
  owner
    0..*
      1

Car
  size

Truck
  tonnage
  getTax()

Trailer

generalization
What is Polymorphism?

- The ability to hide many different implementations behind a single interface

OO Principle: Encapsulation
Strengths of the Object-Oriented Technology

- It leads to reuse
- Reuse leads to faster software development
- Reuse leads to higher quality programs
- OO software is easier to maintain
- OO systems are easier to adapt and easier to scale
OO Project Metrics and Estimation

- Number of scenario scripts
- Number of key classes
- Number of support classes
- Average number of support classes per key class
  - 2-3 in GUI application
  - 1-2 in GUI application
- Number of subsytems
Questions?