A Brief Introduction to the Exciting World of Object Oriented Systems

Peer-Olaf Siebers

IMA Seminar 14/02/2012
Outline

• Object Oriented Systems
• Developing Object Oriented Systems
• Hands-On OOAD&P Example
Object Oriented Systems?
Object Oriented Systems!

• Object Orientation is about viewing and modelling the world/system as a set of interacting and interrelated objects.

• Objects are often used to model the real-world objects that you find in everyday life:
  – Tangible objects (e.g. people)
  – Intangible objects (e.g. orders)

• A class is a blueprint or prototype for objects.
Developing Object Oriented Systems

• Your first day at work
  – Develop the control software for a novel video tagging tool
  – Project budget: £50,000
  – Project time: 6 month
  – Project team: you and two other programmers
  – Produce software that is easy to maintain and extend

• How would you approach this project?
Developing Object Oriented Systems
Developing Object Oriented Systems

• Your task requires an object oriented (OO) approach!
  – OO Analysis
  – OO Design
  – OO Programming
Hands-On OOAD&P Example

• Analysis
  – The User Story
  – Use Case Diagrams

• Design
  – Class Diagrams
  – State Machine Diagrams
  – Sequence Diagrams

• Programming
Hands-On OOAD&P Example

• Analysis: The User Story
  – Stating the need
  – Collecting and prioritising high-level features
  – Should be written by project stakeholders and not the developers
  – Keep it simple!
Hands-On OOAD&P Example

• The User Story
  – Develop a simulation software that allows to simulate customer/staff interactions in a grocery store
    • Story: Customers come into a grocery store, pick up a few items, pay for them, and leaves the grocery store
    • Goals of simulation:
      – Help to improve customer experience
      – Optimise staffing
UML

• UML (Unified Modelling Language) is a family of graphical notations that help in describing, designing and organising object oriented software systems
• Latest version: 2.4
UML Software
Hands-On OOAD&P Example

- Analysis: Use Case Diagrams
Hands-On OOAD&P Example

- **Use Case Diagrams**
  - UML diagram(s) + Documentation + Prototype Screen(s)

  - Telling a story in a highly structured way
    - Define actors (can be roles or systems)
    - Define UseCases (procedures by which an actor might use a system)
    - Sometimes it is useful to sub-divide UseCases into lower level activities
    - Define environment (not always required)

  - Elements: Actors, use cases, system boundaries, connections
    - "x uses y": Task x has a subtask y & y will be completed at least once
    - "x extends y": x is a task for the same type as y but a more specific case
Hands-On OOAD&P Example

![Diagram of a shopping scenario with interactions between customer and clerk. The diagram depicts actions such as picking up items, paying for items, unloading items, passing money, taking change, stocking shelves, scanning items, and passing back change. The system flow is illustrated with arrows showing the sequence of events.]
Hands-On OOAD&P Example

System

Customer

Shopping at store

Picking up items

Paying items

Unloading items

Passing on money

Taking change
Hands-On OOAD&P Example

Diagram:

System

- Serving at store
  - Stocking
    - Stocking fridges
    - Stocking shelves
  - Cashing
    - Scanning items
    - Taking money
    - Passing back change

Clerk

东营大学 - CHINA - MALAYSIA

The University of Nottingham
Hands-On OOAD&P Example

Design: Class Diagrams

<table>
<thead>
<tr>
<th>Use Case Modeling</th>
<th>Structural Modeling</th>
<th>Behavioral Modeling</th>
<th>Architectural Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Diagram</td>
<td>Composite Structure Diagram</td>
<td>Sequence Diagram</td>
<td>Component Diagram</td>
</tr>
<tr>
<td>Use Case Grid</td>
<td>Object Diagram</td>
<td>Communication Diagram</td>
<td>Deployment Diagram</td>
</tr>
<tr>
<td>Actor Grid</td>
<td></td>
<td>Activity Diagram</td>
<td>Package Diagram</td>
</tr>
</tbody>
</table>
Hands-On OOAD&P Example

• Class Diagrams
  – Show a set of classes, interfaces and collaborations, and their relationships
  – Addresses static design view of a system
  – Classes
    • Blueprints (templates) for objects
    • Contain data/information and perform operations

```
Customer
- shoppingList : array[item][quantity]
- shoppingBasket : array[item][quantity]
- money : float
+ enterShop(shop)
+ selectGroceries()
+ checkOut()
```
Hands-On OOAD&P Example

• Your Task:
  – Create the remaining Classes
Hands-On OOAD&P Example

- Class Diagrams

![Class Diagrams](image_url)
Hands-On OOAD&P Example

- Design: State Machine Diagrams
Hands-On OOAD&P Example

• State Machine Diagrams
  – Addresses the dynamic view of a system and is important in modelling the behaviour of an interface, class or collaboration
  – Consist of states, transitions, events, and activities
Hands-On OOAD&P Example

- **Your Task:** Create the remaining State Machines

```
Customer
- shoppingList : array[item][quantity]
- shoppingBasket : array[item][quantity]
- money : float
+ enterShop(shop)
+ selectGroceries()
+ checkout()

Clerk
- availability : boolean
+ serveCustomer(customer)
+ stockShelves()

Item
- type : int

GroceryShop
- clerks : vector[clerk]
- customers : vector[customer]
- stockList : vector[items][quantity]
- numClerks : int
+ hireClerk(clerk)
```
Hands-On OOAD&P Example

- State Machine Diagrams

```
Customer

Clerk

Item

GroceryShop

Enter

InShop

InQueue

BeingServed

Leave

Checking Stock

Idle

Serving Customer

Open

Closed
```
Hands-On OOAD&P Example

- Design: Sequence Diagrams
Hands-On OOAD&P Example

• Sequence Diagrams
  – Shows potential interactions consisting of a set of objects and the messages sent and received by those objects
  
  – Address the dynamic behaviour of a system with special emphasis on the chronological ordering of messages
  
  – Consists of objects, messages, object lifelines, activation
Hands-On OOAD&P Example

• Sequence Diagram
From OOAD to OOP

• Programming

```cpp
class Customer {
    // -- Members variables
    float money;
    std::list<item> shoppingList;
    std::list<item> shoppingBasket;

    // -- Member functions
    void shop();
    void collectGroceries();
    void checkOut();
    void pay(float m);
};
```

```cpp
class Clerk {
    // -- Member variables
    bool availability;

    // -- Member functions
    bool isAvailable();
    void setAvailability(bool b);
    void serve();
    void stock();
};
```
From OOAD to OOP

• The result :)
Questions / Comments