Topics

- Package Design
  - Use Case Packages
  - Class Packages
- Component Design
- Deployment Design
- Collaboration Design

From analysis - Use Case Diagram
From analysis - Class Diagram

<table>
<thead>
<tr>
<th>ConfiguredComputer</th>
<th>configured_price : Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>StandardComputer</td>
<td>standard_price : Currency</td>
</tr>
</tbody>
</table>

Customer
- customer_name : String
- customer_address : String
- phone_number : String
- email_address : String

(from Use Case View)

Payment
- payment_method : String
- date_received : Date
- amount_received : Currency

Invoice
- invoice_number : String
- invoice_date : Date
- invoice_total : Currency

ConfigurationItem
- item_type : String
- item_descr : String

Computer
- computer_name : String

Order
- order_number : String
- order_date : Date
- ship_address : String
- order_total : Currency
- order_status : String
- salesperson_name : String

Each package will eventually have more use cases than shown (in particular, «extend» and «include» use cases)
Boundary class packages

- Most classes that we defined in analysis represented **persistent database objects** ("business objects")
- **BCED application program classes** need to be considered as well

The functions of configuring computers and entering orders require Boundary Packages

```
<<boundary>>
Configuration
GUI
```

```
<<boundary>>
Order
GUI
```

Entity class packages

- **Persistent database classes** correspond to Entity Classes in the application program
- **Entity Packages** represent in-memory run-time structure for persistent database classes

```
<<entity>>
Customers
```

```
<<entity>>
Computers
```

```
<<entity>>
Orders
```

Includes classes Invoice and Payment

Control class packages

- **Control classes** → **Control Packages**
  - represent application logic
  - "glue" boundary and entity classes

```
<<control>>
Configure
Process
```

```
<<control>>
Order
Placement
```
**DB interface class packages**

- To mediate between entity classes and the database
- To handle connections, authorizations, transactions
- To hold “meta-information” about DB schema

```
<db interface>
  CRUD
  Connection
  Schema
```

**Component design**

- **Components** – physical parts of the system
- **Component design** refers to the implementation platform for the system
- **OnLine Shopping** – Web application with database server

**Web application**

- “…Web system that allows its users to execute business logic with a web browser.”
- Business logic can reside on the server and/or on the client
- Client/Server system with a Web site

**Implementing Web applications**

- **Web pages**
  - Rendered in Internet client browser
  - Delivered by Web server

- **Web page document**
  - can be static (unmodifiable) or dynamic
  - can be a form that a user fills in

- **Frames**
  - divide the screen’s “real estate” so that the user can view multiple Web pages at the same time

- **Application server**
  - to manage the application logic
  - to monitor the application state
    - By storing cookies in the browser
    - Session timeouts
Implementing Web applications

- **Dynamic client pages**
  - Script – program interpreted by the browser
  - Applet - compiled component that executes in the browser’s context

- **Server pages** - Web pages with **scripts** executed by the server
  - Have access to DB server
  - Manage client sessions
  - Place cookies on the browser
  - Build client pages

Component diagram

- **Component**
  - Cohesive functional unit with clear interfaces
  - Replaceable part of the system
  - Can correspond to implementation of one or more Web pages
  - Can parallel Use Case Packages

Deployment design

- **Assignment of objects to computing nodes**
- **Difficulties related to Web applications**
  - Connectionless nature of Internet
  - Session management
    - Cookies
    - Distributed objects (CORBA, DCOM, EJB)
  - Web server as the routing point between all client browsers and the database
  - Security
  - Network loads, backups, etc.
Deploying Web applications

- Four tiers of computing nodes
  - Client with browser
    - Static and dynamic pages
    - Scripted pages and applets downloaded and run within the browser
  - Web server
    - Page requests from the browser
    - Generation of pages and code for execution on the client
  - Application server
    - Necessary with distributed objects
  - Database server
    - Data storage
    - Multi-user access

Deployment Diagram

Collaboration design

- Architectural design = packages, components
- Detailed design = collaboration design
- Collaborations define the **realization** of
  - Use cases
  - Operations
- Collaboration design is conducted in parallel with the **elaboration** of
  - Use case models
  - Class models
  - Most other models
Elaborating use cases

Use Case Specification: Order Configured Computer

1. [UC15 Order Configured Computer]

1.1 Brief Description
A customer visits an online shopping site and selects a purchase order form. The system verifies the details and confirms or rejects the order.

2. Flow of Events

2.1 Start

2.1.1 [UC15.1] The user clicks the Order Configured Computer link on the home page. The form shows the following:

- [UC15.1.1] The title of the form is "Order Configured Computer".
- [UC15.1.2] The help text for the input fields is "Enter the details of your computer configuration." The form contains the following input fields:
  - Brand
  - Model
  - Processor type
  - Memory size
  - Storage capacity
  - Operating system

2.1.2 After filling in the details, the user clicks the "Submit" button. The system validates the input and displays the confirmation message.

2.1.3 [UC15.2] If the input is valid, the system saves the order details and redirects the user to the order confirmation page. Otherwise, the user is prompted to correct the errors.

Requirements management

Structure of collaboration

Structure of collaboration = collaboration

Class Diagram extended with application program classes (BCED classes)

Adheres to the enabling technology chosen for the application

Difficulty:
- The enabling technology may not be OO

OnLine Shopping
- Boundary classes – client pages, forms
- Control classes – server pages
Using BCED approach

- **Recommended practice** - prefixing the class names with letters
  - b (Boundary), e.g. b_OrderClientPage
  - c (Control)
  - e (Entity)
  - d (Database Interface)

- **Association and aggregation relationships** to link BCED classes

- **Instantiation relationships** to signify messages that instantiate objects
  - User events leading to object instantiation can be named

---

**Boundary classes**

- **For use case “Order Configured Computer”**

  ![Diagram of boundary classes]

---

**Control and entity classes**

- **For use case “Order Configured Computer”**

  ![Diagram of control and entity classes]
Behavior of collaboration

For use case "Order Configured Computer"

Boundary and control objects

Entity and DB Interface objects

Note from structural collaboration that b_OrderClientPage contains b_ConfigurationClientPage.

newOrderClientPage

incompleteOrder

getConfOID(out ConfOID)

newOrder

commitTransaction

beginTransaction

rollbackTransaction

deleteOrder

newPayment

newCustomer

link(in ConfOID)

c_Order

e_Configuration

e_Payment

e_Customer

[on Purchase] newOrderClientPage

[on Submit] incompleteOrder

[on Cancel] refresh

[on Submit] newOrder

[on Purchase] getConfOID(out ConfOID)