Topics

- Finding classes from use case requirements
- Architectural elaboration of class design
- Class instantiation
- Interactions
  - Sequence diagrams
  - Communication (collaboration) diagrams
- Examples of interactions for EM

About class and interaction design

- **Class design** is the process of ensuring that the classes deliver the behavior specified in the use case model while conforming to the architectural framework chosen for the system.
  - Is inseparable from interaction design
  - Concerned with the design classes
  - Includes the design of interfaces

- **Interaction design**
  - Serves the purpose of verifying the existing class design and augmenting it with further details
  - Signatures (the argument list) of class operations (methods) can be specified
  - Uses
    - Sequence diagrams
    - Communication diagrams (known as collaboration diagrams prior to the UML 2.0)

Finding classes from use case requirements

<table>
<thead>
<tr>
<th>Req. No.</th>
<th>Requirement Definition</th>
<th>Responsible Package and Class Name</th>
<th>Responsible Operation Name</th>
<th>Collaborating Package and Class or Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>The system displays an informational message and requests that the Employee provides a username and password.</td>
<td>presentation PConsole displayLogin</td>
<td>presentation PConsole</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>control CActioner getUserInput</td>
<td>control CActioner</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>The system attempts to connect the Employee to the EM database.</td>
<td>foundation FConnection</td>
<td>foundation FConnection</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>presentation IAConstants</td>
<td>presentation IAConstants, foundation FReader</td>
<td></td>
</tr>
</tbody>
</table>

Elaborating classes from architectural requirements

<table>
<thead>
<tr>
<th>Req. No.</th>
<th>Responsible Class and Operation</th>
<th>Collaborating Package and Class or Interface</th>
<th>Architectural Principle and/or Pattern and/or Other Reason for Change</th>
<th>New/Updated Responsible Class and Operation</th>
<th>New/Updated Collaborating Package and Class or Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PConsole displayLogin</td>
<td>presentation PConsole</td>
<td>Direct Instance</td>
<td>[New/Updated] Responsible class DisplayLogin</td>
<td>[New/Updated] Collaborating package and class interface PConsole</td>
</tr>
<tr>
<td>R2</td>
<td>FConnection getEmployee</td>
<td>foundation FReader</td>
<td>Displacement of Responsibility</td>
<td>[New/Updated] Responsible class FConnection</td>
<td>[New/Updated] Collaborating package and class interface FReader</td>
</tr>
</tbody>
</table>

Presenting classes and dependencies

- PConsole
  - displayLogin()
getUserInput()

- IAConstants

- DATABASE CONNECTION

- CAConstants
  - login()

- FConnection
  - getConnection()

- FReader
  - readEmployee()
### Elaborated classes and dependencies

- **IAConstants**
  - `DATABASE_CONNECTION`

- **FConnection**
  - `getConnection()`

- **FReader**
  - `readEmployee()`

- **CActioner**
  - `login()`

- **MBroker**
  - `login()`
  - `createEmployee()`

- **EEmployee**

### Class instantiation

- **PMain**
  - `main(args : String[]) : void`

- **CActioner**
  - `login()`

- **MBroker**
  - `login()`

- **FConnection**
  - `getConnection()`

- **FReader**

### Instantiation diagram for EM

- **PMain**
  - `<<instantiate>>`

- **CActioner**
  - `<<instantiate>>`

- **MBroker**
  - `<<instantiate>>`

- **FConnection**
  - `<<instantiate>>`

- **FReader**
  - `<<instantiate>>`

- **FWriter**
  - `<<instantiate>>`

- **EEmployee**

- **EContact**

- **EOutMessage**

### Interactions

- **Interaction** - “a unit of behavior that focuses on the observable exchange of information between parts”
- “A part represents a set of instances that are owned by a containing classifier instance”
- The existence of an object (a part) at a particular time is called the *lifeline*
- An interaction is realized as a sequence of *messages* between lifelines:
  - synchronous or
  - asynchronous

### Sequence diagram - notation

- **InteractionName**
  - **Client**
  - **S1 : Supplier**

- **1. method1(arg1 : String)**
  - **1.1. method2( )**

### Messages in sequence diagram
Messages in communication (collaboration) diagram

- Class1
- Class2
- Class3

1. m1()
2. m2()
2.1. m3()
2.1.1. m4()
2.1.1.1. m5()

EM – “Login” interaction

EM – “Exit” interaction

Summary

- Class design and interaction design are two sides of the same coin
- Finding classes from use case requirements involves extracting requirements from the use case document and conceiving of classes and collaborations between classes
- Architectural constraints introduce a need for the elaboration of the initial class design
- Interactions are modeled in sequence diagrams and communication (collaboration) diagrams
- Interactions focus on sequences of messages, not on the data that the messages pass around