Requirements Engineering (CSE491-602, Fall 2006)

Homework #3

Due October 12, 2006 (beginning of class)

Instructions:

The objective of this assignment is to give you experience with creating sequence and state diagrams from requirements and use cases. Please remember that all diagrams must be accompanied by explanatory text. (The grading criteria for the assignment will be posted on the web to offer you guidance as to the key elements of the assignment that must be included.)

This assignment extends the UML model of the ReMoDD system (from HW 2) with state and sequence diagrams. To help you get started, we are providing a class diagram, a sample sequence diagram, and preliminary state diagrams, provided in both jpeg and visio format. (There are two visio files with multiple pages per file.)

1. Include the use case diagram from your first homework assignment, making any corrections that were marked on your returned assignment.

2. Select three use cases from Homework assignment 2. Incorporate into your current assignment the descriptions for each of the use cases, again, making any corrections marked on the returned assignment. Figure 1a contains a class diagram depicting the key participants in the ReMoDD system. Figure 1b contains a data dictionary describing the elements of the class diagram.

3. Using a UML modeling tool, create a sequence diagram for each use case. For example, the Register use case is modeled and described in Figure 2. The corresponding sequence diagram, sequence diagram for Register, is given in Figure 3.

4. Augment/modify the state diagrams to encompass the behavior specified by all three sequence diagrams that were created for part 3. For example, the state diagrams in Figures 4-7, respectively, correspond to the User, UserInterface, Database, and RepositoryAdministrator classes and depict the behavior of the sample sequence diagram for the Register use case.
Figure 1a: High-level class diagram for ReMoDD system.

(www.cse.msu.edu/~chengb/RE-491/Homework/HW3/ReMoDD.vsd  page = Class – ReMoDD)

Figure 1b: Data Dictionary for class diagram (bolded text refers to class diagram elements)

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Person who can have identity</td>
</tr>
<tr>
<td>User</td>
<td>User of repository that has access to the ReMoDD via the UserInterface</td>
</tr>
<tr>
<td>RepositoryAdministrator</td>
<td>Person who has administrative privileges. Maintains the ReMoDD repository</td>
</tr>
<tr>
<td>ReMoDD</td>
<td>Repository containing MDD artifacts housed in the Database and accessed via the UserInterface</td>
</tr>
<tr>
<td>UserInterface</td>
<td>The graphical interface for interacting with the ReMoDD and provides access to the Database.</td>
</tr>
<tr>
<td>Database</td>
<td>As part of the ReMoDD, it is persistent storage for Artifacts; is accessible by the UserInterface.</td>
</tr>
<tr>
<td>Artifact</td>
<td>MDD artifact, such as UML models, source code, patterns. These artifacts are stored in the Database. Each Artifact has a specific User responsible for its existence.</td>
</tr>
</tbody>
</table>
**Use Case:** Register

**Actors:** Repository Administrator & User

**Description:** The goal of this use case is for a user to register with ReMoDD. First, the user attempts to register a user name and password. The user interface checks with the database to determine if the user has previously registered. If the user has previously registered, he/she is redirected to the login screen. If the user has not previously registered, then the user interface alerts the repository administrator and also notifies the user. The repository administrator adds the user to the database and then informs the user that his/her registration was successful.

**Type:** Primary

**Includes:**

**Extends:**

**Cross-refs:** 1, 2

**Use cases:**

---

Figure 2a: Simple use case diagram for ReMoDD

Figure 2b: Use case description for Register

(www.cse.msu.edu/~chengb/RE-491/Homework/HW3/ReMoDDState.vsd page = usecase)
When a user attempts to register a username and password, the user interface checks to see if he/she is in the database.

If so, the user is redirected to the login screen. If not, the user is told to wait for authorization.

If the user is not registered, the user interface alerts the repository administrator, who then adds the user to the database.

Once the user is added to the database, he/she receives an email from the repository administrator.

Figure 3: Sequence diagram for Register use case.
Figure 4: State diagram for **Database** class (covering **Register** use case behavior)
Figure 5: State diagram for User class (covering Register use case behavior)
Figure 6: State diagram for UserInterface class (covering Register use case behavior)
Figure 7: State diagram for RepositoryAdministrator class (covering Register use case behavior)