10 References


CLEMENTS, P.; KAZMAN, R.; KLEIN, M. Evaluating Software Architectures: Methods and Case Studies, USA: Addison-Wesley, 2002


PINTO, M.; GAMEZ, N.; FUENTES, L. Towards the Architectural Definition of the Health Watcher System with AO-ADL. *Proceedings of the Early Aspects at


SILVA, L.; BATISTA, T.; GARCIA, A.; MEDEIROS, A.; MINORCA, L. On the Symbiosis of Aspect-Oriented Requirements and Architectural Descriptions,


The appendix presents the component-and-connector view of the eight releases of Mobile Media AO and non-AO architectures. Figure 63 to Figure 68 show the non-AO architectures. The graphical description of releases 1 and 2 are identical (Figure 63). They differ in the interfaces details. Releases 3 and 4 also have the same representation (Figure 64).
Figure 64: Non-AO architecture of the Mobile Media product line – Releases 3 and 4
Figure 65: Non-AO architecture of the Mobile Media product line – Release 5
Figure 66: Non-AO architecture of the Mobile Media product line – Release 6
Figure 67: Non-AO architecture of the Mobile Media product line – Release 7
Figure 68: Non-AO architecture of the Mobile Media product line – Release 8
Figure 69 to Figure 75 show the AO architectures of the Mobile Media releases. In fact, there is no AO version for the first release of Mobile Media. Aspectual components were included as the change scenarios were implemented, i.e., from the second version on. Therefore, we present here the AO architecture of releases two to eight.

Figure 69: AO architecture of the Mobile Media product line – Release 2
Figure 70: AO architecture of the Mobile Media product line – Release 3

Figure 71: AO architecture of the Mobile Media product line – Release 4
Figure 72: AO architecture of the Mobile Media product line – Release 5
Figure 73: AO architecture of the Mobile Media product line – Release 6
Figure 75: AO architecture of the Mobile Media product line – Release 8
Appendix B –
Detailed Design Metrics Study: Form and Metrics Values

This appendix presents the form provided to the students involved in the
detailed design metrics study which contains information about the target design
as well as the guidelines they should follow in order to conduct the study. In
addition, we also present here the values of the metrics used by the students. Table
30 and Table 31 present these results.

<table>
<thead>
<tr>
<th>Class</th>
<th>Lack of Cohesion in Methods</th>
<th>Coupling Between Object Classes</th>
<th>Number of Attributes</th>
<th>Number of Operations</th>
<th>Weighted Methods per Class</th>
<th>Lack of Concern-based Cohesion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complaint</td>
<td>228</td>
<td>19</td>
<td>14</td>
<td>25</td>
<td>46</td>
<td>1</td>
</tr>
<tr>
<td>ComplaintRecord</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>6</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>ComplaintRepositoryRDB</td>
<td>0</td>
<td>23</td>
<td>7</td>
<td>20</td>
<td>43</td>
<td>2</td>
</tr>
<tr>
<td>Employee</td>
<td>4</td>
<td>17</td>
<td>3</td>
<td>8</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>EmployeeRecord</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>EmployeeRepositoryRDB</td>
<td>0</td>
<td>10</td>
<td>2</td>
<td>6</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>HealthUnit</td>
<td>16</td>
<td>12</td>
<td>3</td>
<td>9</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>HealthUnitRecord</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>7</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>HealthUnitRepositoryRDB</td>
<td>0</td>
<td>15</td>
<td>3</td>
<td>10</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>HealthWatcherFacade</td>
<td>0</td>
<td>21</td>
<td>4</td>
<td>23</td>
<td>36</td>
<td>4</td>
</tr>
<tr>
<td>HealthWatcherFacadeInit</td>
<td>71</td>
<td>29</td>
<td>5</td>
<td>18</td>
<td>29</td>
<td>3</td>
</tr>
<tr>
<td>HWServlet</td>
<td>1</td>
<td>21</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>iComplaintRepository</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>iEmployeeRepository</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>IFacade</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>16</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>IHealthUnitRepository</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>IPersistenceMechanism</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>PersistenceMechanism</td>
<td>63</td>
<td>8</td>
<td>11</td>
<td>14</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>ServletInsertEmployee</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>ServletSearchComplaintData</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>ServletUpdateComplaintData</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>ServletUpdateHealthUnitData</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 30: Results of the metrics for the Health Watcher design used in the study about
detailed design metrics (Section 8.2) - values obtained per component.
<table>
<thead>
<tr>
<th>Concern</th>
<th>Concern Diffusion over Classes</th>
<th>Concern Diffusion over Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrency</td>
<td>8</td>
<td>42</td>
</tr>
<tr>
<td>Distribution</td>
<td>49</td>
<td>76</td>
</tr>
<tr>
<td>Exception Handling</td>
<td>73</td>
<td>294</td>
</tr>
<tr>
<td>Persistence</td>
<td>41</td>
<td>158</td>
</tr>
<tr>
<td>Business</td>
<td>37</td>
<td>222</td>
</tr>
<tr>
<td>View (GUI)</td>
<td>21</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 31: Results of the metrics for the Health Watcher design used in the study about detailed design metrics (Section 8.2) - values obtained per concern.

In the following we present the form provided to the students with the instructions to conduct the study.

**Group Members**
1. ________________________________________________________
2. ________________________________________________________

**Experimental Steps**

**Step 1)** The group start to follow the guidelines and answer the questions. The answers are provided by the group (just one form per group).

**Step 2)** Start to read the description of the Health Watcher system provided below.

**IMPORTANT:** before you start to read the description, indicate here what time it is now: _:_ _pm

**Health Watcher System – Design**

The Health Watcher (HW) system allows a citizen to register complaints to the health public system. Figure 1 shows a diagram representing some of the classes and interfaces of the Health Watcher system. This diagram is actually a simplification of the HW design. In the HW system, complaints are registered, updated and queried through a Web client. The system is structured with the goal of decoupling different parts of the system in order to make it easy to change them separately. Therefore, the user interface part of the system is decoupled from the part which implements the business logic which, in turn, is also decoupled from the database management.

The user interface is represented by the Servlet classes, such as ServletInsertEmployee and ServletSearchComplaintData (Figure 1). Accesses to the HW services are made through the IFacade interface, which is implemented by the HealthWatcherFacade. One of the Health Watcher’s requirements is to allow several customers to access the system at the same time. Therefore, a client-server approach is used to distribute part of the execution, which is implemented by the classes HealthWatcherFacade and the
HealthWatcherFacadeInit. The HealthWatcherFacadeInit works as a portal to access the business collections, such as ComplaintRecord and EmployeeRecord. Records access the data also using interfaces, like IComplaintRepository, which decouple the business logic from the specific type of data management in use. Figure 1 shows the classes that serve the purpose of implementing a repository for a relational database: HealthUnitRepositoryRDB, EmployeeRepositoryRDB and ComplainRepositoryRDB. Table 1 summarizes the main responsibilities associated with each class in the HW system. Table 2 describes the main design concerns.

Table 2. Main Design Concerns

<table>
<thead>
<tr>
<th>Concern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>It defines the business elements and rules.</td>
</tr>
<tr>
<td>Concurrency</td>
<td>It provides a control for avoiding inconsistency in the information manipulated by the system.</td>
</tr>
<tr>
<td>Distribution</td>
<td>It is responsible for externalizing the system services at the server side and supporting their distribution to the clients.</td>
</tr>
<tr>
<td>Exception Handling</td>
<td>It supports error recovery.</td>
</tr>
<tr>
<td>Persistence</td>
<td>It is responsible for storing the information manipulated by the system.</td>
</tr>
<tr>
<td>View (GUI)</td>
<td>It provides a Web interface for the system.</td>
</tr>
</tbody>
</table>
Table 1. Class Responsibilities

<table>
<thead>
<tr>
<th>Classes</th>
<th>What they do</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWServlet, ServletInsertEmployee, ServletSearchComplaintData, ServletUpdateComplaintData, ServletUpdateHealthUnitData</td>
<td>These classes implement the user interface of the system.</td>
</tr>
<tr>
<td>IFacade, HealthWatcherFacade, HealthWatcherFacadeInit</td>
<td>These classes provide a simple interface to all services of the system.</td>
</tr>
<tr>
<td>HeathUnit, Employee, Complaint</td>
<td>These classes represent business basic concepts of the Health Watcher system domain.</td>
</tr>
<tr>
<td>HealthUnitRecord, EmployeeRecord, ComplainRecord</td>
<td>These classes represent a grouping of objects from a basic class. For instance, EmployeeRecord groups objects of Employee.</td>
</tr>
<tr>
<td>HealthUnitRepositoryRDB, EmployeeRepositoryRDB, ComplainRepositoryRDB</td>
<td>These classes contain methods for manipulating persistent objects of the business basic classes. The code of these classes depends on a specific API for accessing some persistence platform.</td>
</tr>
<tr>
<td>IHealthUnitRepository, IEmployeeRepository, IComplainRepository</td>
<td>These interfaces establish a decoupled relationship between the “Record” classes and “RepositoryRDB” classes. The goal is to make the business code more independent from the data access code and from a specific persistence platform.</td>
</tr>
<tr>
<td>PersistenceMechanism</td>
<td>This class implements services such as connecting to and disconnecting from the database, transaction mechanism, concurrency mechanism.</td>
</tr>
</tbody>
</table>

Step 3) The designers of the Health Watcher system need your help to identify which classes contain the bad smell “Divergent Change”. Start to read the description of the bad smell and the measures. Remember that it is important to reason about the metrics and identify which of them (one, some, or all) are relevant indicators based on the bad smell description.

IMPORTANT: before you start to read the text and measures, indicate here what time it is now: _:_ _pm

Step 4) Your group should now answer the following questions:

a) Which are the classes with the highest probability of having the bad smell “Divergent Change”? You should rank your list of classes; the ones with highest probability should come first.

IMPORTANT: when you finish to answer this question, indicate here what time it is now: _:_ _pm
b) Explain which metrics you have used for detecting the bad smell, and how they were useful to identify the classes mentioned in the previous question. Which ones were not useful at all?

**Step 5** The goal now is to detect the presence of “Shotgun Surgery” bad smells. So identify **which classes** when changed are more likely to propagate changes to other classes. Start to read the description of the bad smell and the measures. Remember that it is important to reason about the metrics and identify which of them (one, some, or all) are relevant indicators based on the bad smell description.

**IMPORTANT**: before you start to read the text and measures, indicate here what time it is now: _ _ _pm

**Step 6** Your group should now answer the following questions:

a) Which are the classes (when changed) that have the highest probability of propagating changes to other classes? You should rank your list of classes; the ones with highest probability should come first.

**IMPORTANT**: when you finish to answer this question, indicate here what time it is now: _ _ _pm

b) Explain which metrics you have used for detecting the bad smell, and how they were useful to identify the classes mentioned in the previous question. Which ones were not useful at all?

**Step 7** Answer the questionnaire about the experiment design issues.