bSecure – Computerized building security system

Assignment 3: Low Level Design & Work+Test Plan

Introduction to Software Engineering
Department of Computer Science
Technion – Israel Institute of Technology

Winter 2009-2010

1. Goals
In this phase we will simplify and define a subset of all the requirements as build 1. The details of build 1 will be discussed later on. You will produce a low level design and a test plan for this build. You will be required to prepare a document containing these two sections:

1. Low Level Design: will include the Low Level Class Model and the Sequence Model.
2. Test Plan: will include a description of the system tests you intend to run along with their expected results.
3. Work Plan: will include a description of your plans to see the successful delivery of build 1

You should prepare the sections as shown in the RoboLib example project.

2. Responsibility Distribution
The tasks should be distributed uniformly among the members of the group. You are required to describe the responsibility of each member in executing the phase. Define the role and actual work of each member, in the level of functional responsibility: each member is responsible for some modules and/or functionality and perhaps some development task, such as document integration, internal reviews, etc of the system.

3. Submission
In addition to the Low Level Design, the Test and Work Plan described in the project goals you should also provide the previous submissions which include the updated requirements document and the high level design document from assignment 2. These will not be graded but will be used to check consistency. If you need to make any changes in these documents please make sure you also provide a change log document describing your changes. The document (including previous phases) should be printed and submitted as one coherent document containing multiple sections, and also submitted electronically in pdf format via the course website. The submission should also include a cover page including the group name, the names and ids of all group members, and also the number of cell to return the checked exercise to. You should use the RoboLib documents as a reference for the format of the submission (diagrams, tables etc.).
Please inform Roy (royl@cs) and Pavel (pg@cs) of any problem, and in particular Miluim, as soon as such issues arise.
3.1 Low Level Design
The section should include the following components:

- One or more Class Diagrams. It should be clear from your design how your system is organized. When designing your classes you should also think of design solutions that will allow you to progress to build 2 (the next version of the software that will include more of the original requirements) with minimal changes to your code. Consider using relevant design patterns for this purpose.

- Each Class Diagram should include a class description table that contains:
  - The class name
  - A description of the class (including its role and responsibilities)
  - Class members (attributes and methods) including a detailed description. The methods should also include return values and parameters.

- Sequence diagrams that cover all the use cases that are relevant to build 1.

3.2 Test Plan
The test plan should include the following:

1. *The Acceptance Test* - this is basically the black box system tests that you intend to perform on the system, to show its correctness. The tests should be thorough and convincing that in fact your system is coherent with the requirements. The document should be formatted in the same manner as the test procedure document from the black box testing tutorial. The table should contain the following 4 columns:
   - Requirement numbers
   - Verify that
   - Test description
   - Expected result

2. *Unit tests* – for this part you should pick at least 5 of your major classes (classes that have meaningful, non-trivial methods) and specify a test plan for them. You should describe the following:
   - Stub (mock) objects you intend to implement so that you can test these classes in isolation.
   - Sequence of method calls and description of the class’s expected state (the described state should be testable – keep in mind that in the next assignment you will write the actual unit tests that will realize this description).

3.3 Work Plan
The work plan should include the following:

1. Define how you plan to perform your integration tests. A possible solution is to perform the integration tests by creating the system before all its components are complete and to begin running the acceptance tests that you expect to be successful.

2. Define when do you plan to begin the integration tests (you should specify a deadline). Keep in mind that a good plan will not delay the integration phase until a few days before delivery.

3. Define who will be in charge of running the acceptance tests during integration and as the final integration phase. You do not need to specify an individual’s name, just say if it will be one person in charge of QA or each
team member should run the acceptance tests relevant to his code. You may decide on different options as well.

- For this section you should note that the implementation phase of **build 1** should be submitted until the 20th of January 2010

### 4. bSecure – build 1

bSecure is a fully computerized building security system, comprising a large number of secure sectors, with different security levels. Each security sector is equipped with a camera, powerful speakers, an entrance or exit request button respectively and automatic rotating doors (capable of containing one person) deployed at its entrance and exit points. – (1) a single entrance door equipped with a speakers, an entrance request button and an enter with guest request button, and (2) a single exit door equipped with speakers and an exit request button.

Whenever a person arrives at an entrance to a security sector he may express his desire to enter the sector by pressing the request entrance button. The button then turns on and the request is then sent to the central computer which then responds by sending a recorded message to the entrance speakers directing the person to face the camera and wait for the recognition process to end. The camera begins recording and sends the images to another machine which is in charge of recognizing the face and retrieving. If the recognition was successful the person’s unique identification number, hereafter the person's id, is retrieved. The face recognition software is a commercial off-the-shelf product. The central computer then accesses the database to retrieve the person's security restrictions. If the person is allowed access he is instructed, via the speakers, to enter the door and the request entrance button turns off. The rotating door then performs a single rotation which takes 10 seconds. Similarly, when a person requests to exit a security level he reaches the exit doors, presses the request exit button, the button then turns on, he faces the camera and if no problems are detected he is allowed to exit and the door rotates (the rotation duration is also 10 seconds).

When a person exits a security sector the system verifies via its database that in fact the person is listed as having entered this security sector. The information regarding the individual's stay in the sector is recorded by the system.

There are four levels of restrictions. Each restriction level allows access to all security sectors below or equal to it. In other words, personnel with security restriction 3 may access sectors of level 1, 2, or 3 but cannot access sectors of security level 4. A person requesting to enter a security sector with a higher level of security than that which he has can only do so for a limited time, provided he is accompanied by a person with a proper security level.

Security restriction levels are assigned by a system administrator. The administrator has a console which allows him to enter new people into the system and assign security restrictions (levels 1-4) to them. The administrator can access the system by entering his username and password is constantly logged on to the system. After the administrator logs on, a welcome screen is displayed with the administrator's options presented. The administrator may add a new user, change an existing user's security restrictions, or delete a user from the system. When deleting a user, the administrator may also select to add this user to the black list. When entering a person into the black list, the administrator must also provide a reason for doing so in a special comment.
Personnel that are blacklisted cannot be entered into the system again in the future.

A person’s details consist of his full name, address, id number, image, and a list containing the security restrictions he has been granted since he was registered to the system. The list also contains the starting and ending dates each of these security restrictions have been granted and if revoked, a reason field also exists.

Once a month, the system produces a report containing detailed information about access to each security sector. The report contains the list of people that have entered the areas, the time they spent in each area, and who accompanied them (if relevant). Note that the report contains all the data and not just in the last month.

Whenever a security violation occurs, the administrator is notified and until he responds, no personnel are allowed to enter or exit any security sector (a situation called “lockdown”). The system also announces to each sector’s loudspeakers that the building is undergoing a security lockdown due to a breach in security. The administrator is the only one that can turn off the security lockdown and return the system to its standard state.

**build 1** is a simulation of the real system we wish to iteratively produce in multiple phases (**build 1** is the only phase you will be required to implement in this course). The simulation system will receive via the standard input **xml** content that describes (1) the events that occur in the building, (2) the actions that are performed by the single administrator and (3) system tester commands. In cases where the system is required to respond, the simulation will generate **xml** content to the standard output. Finally, the system’s data (such as the description of the sectors) will be provided via a static **xml** file. We will now describe the data contained in each of these **xml** documents. We leave the definition of the exact format of these documents to the next assignment (assignment 4):

- **Input xml**:
  - Sector event – a sector event will contain the sector number and the event type which can be any of the following:
    - Entrance/Exit request button pressed
    - Entrance with visitor request button pressed
    - Person in front of entrance/exit door identified as id number x
    - Violation in entrance/exit door (possibly two people inside door, or someone attempting to force the door open, etc …)
  - Admin operations – this simulates the operations performed by the system administrator in his dedicated console. The operations can be as follows:
    - Enter a new user into the system
    - Modify an existing user’s security restrictions
    - Delete an existing user
    - Delete a user and add him to the black list
    - Display person’s details
      - In this case the user details are sent to the standard output in **xml** format.
    - Produce report now
      - In this case the data is sent to the standard output in **xml** format.
- Cancel system lockdown
  - System tester commands – this allows the state of the system to be queried for coherency. The following commands should be supported:
    - Display sector data, a sector number is provided.
      - The output should include:
        - The sector’s number
        - The state of the entrance/exit doors (rotating/idle)
        - The message being broadcasted by the sector entrance/exit doors speaker system
        - A list of people currently known to be inside the sector, where guests are linked to their hosts. You may assume a person is in a sector if he requested to enter and the doors have rotated for him.
    - Display lockdown status
      - The output should say if there is currently a system lockdown or not

Please note that you should provide an xml response to any command that the system receives. In the cases where a specific response has been specified (e.g. when the tester wishes to display a sector) then you should respond as stated, otherwise you should provide a general response, saying if the command has been processed successfully or the command caused an error. E.g. if the admin attempts to add a user that is black listed the simulation should respond with an error, otherwise the system should respond by stating that the command has been executed successfully.

Good luck!