Professional Software Development 3
Group Exercise 2:
Undergraduate Teaching Session Booking System

Tim Storer and Jeremy Singer

Revision 368, made 29/01/2014 by tws

Background

The main aim of this group exercise is to provide group members with the opportunity to practice object-oriented design, development, testing and deployment of software, in particular using the UML, Java, and the OSGi middleware specification.

At the end of this group exercise, you will be able to:

- Produce UML-based designs for a system using component and class diagrams, as appropriate;
- Create an automated test harness comprising both unit and acceptance tests.
- Manage your development effort using version control and a continuous integration environment.
- Produce a Java implementation of your design as a collection of one or more OSGi-like bundles.
- Make a change to an existing design.

Task

During the consortium workshop you discussed the overall architecture of the proposed system and its partition into components. You also discussed which user stories were highest priority and should be implemented during the first sprint of a development project. The remaining project time will now be divided into three sprints, as summarised below.

Sprint	**Start**	**Due**	**Finish**	**% Weight (of GE2)**
Implementation	Mon 3rd Feb	Thu 13th Feb	Fri 14th Feb	20%
Test harness	Mon 17th Feb	Thu 27th Feb	Fri 28th Feb	40%
Modification	Non 3rd Mar	Thu 13th Mar	Fri 14th Mar	40%

All deadlines are at 4.30pm. There is credit in each sprint for both software product delivered and for evidence of a well managed development effort (evidence for this will be taken from your ticket management and version control systems.

Note that each sprint formally finishes a day after your submission. You are encouraged as team to use this time to conduct a retrospective of the completed sprint to (a) help you improve your processes for the subsequent sprint and (b) provide you with useful material for your individual journals.

Sprint 1 Implementation

The customer has now agreed that user stories 1,2,4,8,11,12,14 should be implemented in the first sprint. In addition, the system must implement *all* of the non-functional requirements.

Specific tasks for the sprint are as follows:

1. Each team is now required to develop and implement their own system design to realise the required user stories. The system must comprise *one or more* components implemented as OSGi bundles. The system must expose sufficient interfaces to permit acceptance testing.

2. You must document your component architecture using a UML component diagram on your project wiki under a page called *ComponentDiagram*. A photo of a (reasonably neat) whiteboard diagram or similar hand drawing is sufficient for this. You may add any furhter diagrams that help to explain your design to this page as well.

3. You must provide stub components for any part of the system or dependencies needed to test the actual components you build. For example, you need to provide a stub component to authenticate users via MyCampus.

4. You must provide ant and ivy build scripts to download any dependencies for your system and compile your code.

5. The system must be managed in a version control repository and built using the provided Jenkins continuous integration environment. You must configure exactly one job for your team called *TeamXMainBuild* (where X is your team's label) that is triggered whenever a commit is made to your version control system.

6. The system *does not* require a user interface. Rather, the behaviour of the system will be demonstrated through the use of an acceptance test harness in Sprint 2.

A summary of marks available is shown below.

Criteria	**Marks available**
Re-organised tickets on Trac (or similar) to reflect revised requirements	2
Evidence of use of Tickets to track and manage progress	2
Evidence of regular commits to version control and useful comments	2
Architectural component diagram	3
Appropriate component implementation in OSGi	2
Implementation of user stories	3
Appropriate exposure and documentation of component interfaces	3
Ant build script and dependencies managed using ivy	2
Working Jenkins job	1

Sprint 2 Acceptance Test Harness

The goal of the second sprint will be to develop an acceptance test suite for the system. Each team will also be required to specify scenarios which describe the *conditions of acceptance* for each user story implemented. These scenarios must be tested using an automated test harness that can be executed in the Jenkins continuous integration environment provided.

Specific tasks for the sprint are as follows:

1. Addition of new tasks to ticket management system.
2. Implementation of a test target in the project ant script and an alteration to the team's Jenkins build job to invoke the target.
3. Development of a collection of scenarios describing the conditions of acceptance for each user story implemented. These should be written in a story file and stored in the project source code in the appropriate test package.
4. Implementation of a test harness that exercises the documented scenarios, using either JBehave or manually implemented tests in JUnit.

Criteria	**Marks available**
Evidence of use of Tickets to track and manage progress.	5
Evidence of regular commits to version control and useful comments.	5
Test target in ant script and re-configured Jenkins build job.	2
Conditions of acceptance scenarios file(s) for user stories and associated test harness.	14
Conditions of acceptance scenarios for non-functional requirements and associated test harness.	14

Sprint 3 Modification and Refactoring

During the final sprint you will be asked to make a modification to the proposed system due to changing customer requirements. The required change will be announced on Sunday 2nd Mar.

You may plan that the effort required for this task is equivalent to the implementation of a single user story. The overall implementation will also be re-assessed following the previous four weeks development.

Specific tasks for the sprint are as follows:

1. Additional conditions of acceptance scenarios to accommodate the revised requirements. If necessary, the project test harness should also be altered to implement the scenarios.

2. Revised system component diagram to reflect any changes needed to accommodate the changed requirements. This should be documented on the team project wiki on a page called *RevisedComponentDiagram*.

3. Implement the revised requirements in the system.

4. Refactor the system design and implementation as appropriate to improve quality.

Note that the final task will require a variable amount of effort, depending on the completeness of your submission for Sprint 1.

Criteria	**Marks available**
Evidence of use of Tickets to track and manage progress.	5
Evidence of regular commits to version control and useful log messages.	5
Use of ant, ivy and Jenkins to manage the build and testing of the system.	4
Conditions of acceptance scenario file(s) for the revised customer requirements and associated changes to the test harness (if required).	4
Alteration of system component diagram to accommodate the revised customer requirements.	4
Implementation of features to realise the system modification	4.
Appropriate use of component interfaces to expose component behaviour for use and testing.	4
Overall system implementation quality.	10