Queen Mary
University of London

School of Electronic Engineering and Computer Science

EBU5304 – Software Engineering Group Coursework

30% coursework. [*Student groups are allocated by the module organiser*.]

Developing a Tourist Train Journey Management System using Agile Methods

1. General information

In the next few weeks, your team will be required to develop a tourist train journey management system using Agile methods. Iterations should be planned and Agile methods should be used in all activities, from requirements, through to analysis/design, implementation and testing.

It should be noted that determining the requirements of a system is one of the most important and complex phases in any development project. The given specification contains a lot of noises—requirements are described in an abstract and ambiguous way. You should apply requirement finding techniques and Agile methods to extract the relevant information at appropriate level. And most importantly, you need to set a priority for the features that are implemented in accordance with both ease of implementation and meeting customer requirements. As Agile is designed to adapt to change, do expect that new requirements or change of requirements will be announced at any time during the development stage. As in real systems though, there may be more details you want to know that are missing from the given specification. You can make your own assumptions but do NOT over design. Keep your design SIMPLE. Bear in mind that there is no absolute right answer – your solution may be perfectly appropriate.

Handout release date: **Monday, 14th March 2016**
First coursework submission (Requirements): **Monday, 25th April 2016**
Final coursework submission (Final Report and Software): **Monday, 23rd May 2016**
Group demonstration:
Tuesday 31st May 18:30-20:30, groups 1-40
Thursday 2nd June 18:30-20:30, groups 41-82
Friday 3rd June 13:30-16:00, groups 83-101

Marks returned: Approximately 2 weeks after the final coursework submission.

2. Specification of coursework

Tourist train is a unique way of enjoying sightseeing in a large area. Tourists can hop on and off at any of stops to commence the tour or spend some time at their favourite places. A tourist train network is going to be built in a newly developed tourist area. Your company won the bid for developing the software system of the train network and you are the Agile software development team that is responsible for developing the train journey management system.

The purpose of the required journey management system is to manage tourist trains that are serving a number of routes on the new area. The system manages the allocation of trains to journeys and tracks their progress along the routes. Each route is served by a timetable, which states the times at which a train will depart from designated train stops along the route. A single outward and return trip on the timetable is known as a journey. Each route may be served by a number of trains, but only one train is assigned to a particular journey at a time. Driver information should also be managed. Drivers are assigned to journeys when they are available.

An example route timetable, which has three journeys, is given below:

Route 1 Outward			
Central Station	9:00	10:00	11:00
Stop A	9:13	10:13	11:13
Stop B	9:25	10:25	11:25
Stop C	9:38	10:38	11:38
Stop D	9:55	10:55	11:55

Route 1 Return			
Stop D	10:00	11:00	12:00
Stop C	10:17	11:17	12:17
Stop B	10:30	11:30	12:30
Stop A	10:42	11:42	12:42
Central Station	10:55	11:55	12:55

All trains are based at the central station and must return to the central station at the end of the day. Each train has an on-board screen to display the next stops and times. Each train stop has a screen to display the upcoming train information and train timetable. The system keeps track of the current movement of the trains throughout their journeys. The operation manager can manage routes, assign train/driver to journey, schedule the train timetable, track the train location and remotely control trains to start/stop. The driver can start/stop the train manually on board.

The main areas of functionality of the journey management system are:

- Maintain train and driver information.
- Manage routes and journeys.
- Assign train to journey: A train can only be assigned if it is available (i.e. it is not assigned to another journey).
- Assign driver to train: A driver can only be assigned if he/she is available.

- Check train live status:
    - Update location: this occurs when new information is received about the new location of a train. Information regarding a train location can come from a variety of sources, including the tracker on the trains, the sensor at the train stops and messages from driver. This information will consist of the details of the train and the stop it has just arrived at. This information is recorded by the system and is used to update the current known location of the train. (Hint: 1. think! Is this function part of the management system? 2. Program to interface – define the interface but leave the implementation details to the client code. 3. Think about software simulation)
    - Get current location: this provides information about the last known location of a train on a particular journey. If a train is not currently on a journey no information will be available.
- Manage timetable.
- Start/stop train remotely.
- Synchronise data with on-board train information display and train stops information display.

Your tasks are to define detailed requirements, develop and test the above described software system for the train network using Agile methods. Your design must be flexible and extensible, so that it can adapt to continuously changing requirements and can be used in a general market.

Your design of the software system must be capable of adapting to such future changes. That is, when developing a new system, you should be able to reuse the existing components. When adding new features to the existing system, you should make the least impact on the existing code.

For any details of the system or operation which is not clearly stated, ***you may make your own assumptions***. In your report, make it clear where you have made assumptions. Feel free to design the system as long as it satisfies the basic requirements, but do NOT over design it.

3. Agile project management

Each coursework group has 6 students[1]. You are the Agile team working together to complete the coursework. All students in a group must work on all aspects of the project, in order to obtain full software engineering skills.

You should use the techniques you have learnt in the lectures to manage the project, e.g. Scrum, daily stand up meetings, working around a table, scrum master and one hand decision making etc.

4. First submission: Requirements

The **first coursework submission** is the **Product Backlog**.

- Use the template provided on QMPlus, feel free to modify it to meet your needs.
- The submission must be an EXCEL file.

5. Second submission: Final Report and Software

The **final coursework report** should contain the following parts:

i. Summary of Responsibilities and Achievements (see the template on QMPlus)

ii. Project management

- The project management in your team working. E.g. using project management techniques, planning, estimating, decision making and adapting to changes.

iii. Requirements

- Apply the requirements finding techniques.
- Software prototypes (these can be hand-drawn).
- Describe any changes of the product backlog since the first submission.
- Iteration and estimation of the stories.

iv. Analysis and Design

- A design class diagram describing the design of the software classes in the system, show the class relationships. Note that your design should *address the issue of re-usability of system components*. You should provide clear justification for your proposed approach and show that your design is adaptable to change where necessary.
- Discuss the design of the software.
- Discuss the extent to which your design and the code that implements it meets the main design principles of programming.

[1] Due to the size of the cohort, some groups may occasionally have 7 students instead.

v. Implementation and Testing

- Discuss the implication strategy and iteration/built plan. Discuss the experience of pair programming.
- Discuss the test strategy and test techniques you have used in your testing.
- Discuss the using of TDD. Note: TDD is not required for developing the whole system, however, you should try to use TDD to develop a few programs.

vi. All reports should include an introduction, a conclusion and a list of references.

vii. Main screenshots of the system should be included in the appendix (**Note**: Convert them to JPEGs).

The **software** should contain the following parts:

i. A working system written in Java. All main functionality should be implemented. Code should be well documented. You should create a few simple GUIs to represent the main system.)

ii. A set of test programs using Junit.

iii. JavaDocs.

iv. User manual.

v. Note: You only need to consider the logical information flow for controlling the train remotely or sending information to other systems. For example, you may use a button to start/stop trains. You may display a message show “information sent”.

6. Demonstration

ALL group members MUST attend the demonstration. The demonstration will last 20 minutes. It is OK if only some features are implemented, your code is incomplete or has bugs – just show your latest built of the working system, you still can receive full marks of the demonstration. Remember we look at setting a priority for the features that are implemented in accordance with both ease of implementation and meeting customer requirements.

You will be asked to demonstrate the core functions: manage route, journey, train and driver information. Assign trains to journeys and assign drivers to trains. Track train live location (this will be done through software simulation). Detailed instructions of the demonstration will be sent out in due course.

7. Submission Details

There are THREE electronic submissions to QMPlus:

- Product backlog in EXCEL format. This must be named **ProductBacklog_groupXXX. xlsx**, where **XXX** is your group number.
- Final report in PDF format (maximum 25 pages, excluding the Appendix). This must be named **FinalReport_groupXXX.pdf**, where **XXX** is your group number.
- Software: file stored in ZIP format containing all the .java files of product programs and test programs, Javadocs, user manual and a Readme file to instruct how to set up/configure and run your software. Do not include .class files, as your programs will be re-compiled. This must be named S**oftware_groupXXX.zip**, where **XXX** is your group number.

Only one EXCEL file, one PDF report and one ZIP file should be uploaded for each group – this is the responsibility of the **group leader** (or one of the members in case the group leader is ill/absent). The group leader MUST check that they have completed the full submission.

It is the responsibility of each group to check that their report has been correctly converted into PDF format **before** it is electronically submitted. Do NOT email reports (or other parts of the coursework) to lecturers – only coursework materials submitted via QMPlus will be accepted.

DO NOT LEAVE YOUR COURSEWORK SUBMISSIONS TILL THE LAST MINUTE – NEVER TRY TO CONVERT AND SUBMIT YOUR COURSEWORK E.G. 2 MINUTES BEFORE THE DEADLINE! Marks will be lost for late submission of any section of the coursework.

Note that all code **MUST run from the command line** with no requirements to install extra software (e.g. database, IDE, …). Any code that cannot run from the command line will be heavily penalised.

8. Important notes

Although a real system would require more advanced features (such as consideration of security issues, database access, etc) this would distract from the core software engineering skills that must be developed on this course. The following guidelines MUST be followed; otherwise groups will lose many, if not all, coursework marks.

- **Standalone system.** The system must be developed as a standalone Java application with SIMPLE Java GUIs (AWT, Swing). Students should use Java SE 7 or above.
- **NO network programming.** Students must NOT write HTML, JavaScript, servlets, JSPs, sockets, ASPs, PHPs, etc … This is NOT a network programming course. You must develop a stand-alone application and ignore any networking concerns.

- **NO database implementation**. Students should develop this application without using a database, i.e. do no not use JDBC, Access/Postgres/MySQL database etc. Students should concentrate on their software engineering skills without being concerned by more precise deployment issues. Instead of a database, you should use a proxy e.g. flat (i.e. plain text file, CSV) file(s). **Hint**: Students should think about the use of Java interfaces for the database (or proxy database) access.

- **Code development.** Code should be written for maintainability and extensibility – it should both contain Javadocs and be clearly commented. Responsibilities should be separated – one class should be responsible for one thing only.

- **Code delivery.** A Readme file should explain clearly how to install, compile (i.e. what to type at the command line) and run the code. All code should run from the command line and MUST NOT require users to install any extra software (e.g. database, Eclipse or any other IDE) or extra Java libraries.

- **Key Points of report.** Markers will be impressed by quality, not quantity – a huge report is not an indication that the software engineering is good. Markers will be impressed by groups who can criticise their solution and indicate how this can be improved in future iterations. Students should take care over the presentation of the report (and check for spelling and grammar mistakes) – they should imagine that this report will be presented to the client. Students should not spend hours and hours concentrating on making the most beautiful GUI! For example, no marks will be gained for designing a lovely logo. The focus of this work is software engineering – correct functionality and elegance of code (classes that do only one thing, methods that do only one thing, code that is not duplicated, delegation, i.e. following the principles outlined in the course) are much more important.

- **Key points of Participation and Achievement.** If students are not turning up to meetings or doing any work, then the module organiser need to be informed **immediately**. The coursework is marked out of 100 – 90% are group marks and 10% are given for individual participation/achievement as presented to the markers. If a student has not participated at all in the group they will get NO individual marks and NO group marks.

9. Marks breakdown (approximate)

Maximum mark: 100.

Group mark (approximately 90 marks)

- Ability to extract and define the system requirements using Agile techniques **(30%)**
- Ability to refine the requirements through analysis and Ability to design high quality software **(30%)**
- Correctness of Java code (**20%**) – the code must match the design. *If the code does not match the design* (or if there is no correspondence between the design/code), then ALL these marks will be lost.
- Testing (**10%**) - appropriate test strategy and correctness of using Junit to test.
- Project Management **(5%)**
- Quality of report **(5%)**

Individual mark (approximately 10 marks)

- Analysis of the Summary of Responsibilities and Achievements
  - Accurate description of achievement.
  - Participation in the group: Quality and accuracy of work performed under individual group members' responsibility; Evidence of the performed work; Understanding of the performed work.

You, AS A GROUP, are responsible for managing any issues and for completing all of the tasks.

Please use the messageboard on QMPlus for enquires and discussions; do not email the lecturers unless it is a personal related issue.