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 lecturers.*]

Developing a Car Park Control System using Agile Methods

Release Date: **Monday, 30th March 2015**
First coursework submission (Requirements): **Thursday, 30th April 2015**
Final coursework submission (Final Report and Software): **Friday, 29th May 2015**

Coursework Submission details: check the information on *QMPlus*.
Marks returned: Approximately 2 weeks after the final coursework submission.

1. Specification of coursework

Queen Mary University plans to build a new car park in the campus. This car park has 30 parking spaces, one entrance, one payment station and one exit. Both the entrance and the exit are barrier controlled. This car park is for University staff use ONLY during normal weekdays. It also opens to the public during weekends, public holidays and college holidays.

You are an Agile software development team for developing the car park control system. The specifications and requirements are described as follows:

Staff use:
University staff can use the car park at any day of the year and they are charged at a flat rate of £1 per day. The car must be registered on the system to use the car park. Once registered, University staff can get in and out of the car park by scanning the campus card at the barriers. University staff should receive a monthly bill from the system and the charge is automatically deducted from their salary.

Public use:
The car park is open to the public at weekends, public holidays and college holidays only. A customer is charged according to the tariff shown in **Table 1**. A customer should be issued a ticket at the entrance barrier. Payment should be completed in advance of reaching the exit, via the automated payment station. The payment station accepts coins of 50p, £1 and £2 ONLY and it CANNOT give change.

Up to 2 hours	50p
2 to 4 hours	£1
4 to 8 hours	£2
8 to 12 hours	£3
12 to 24 hours	£5

Table 1 – Public Tariffs

The car park system should:

- Register staff cars.
- Open the entrance barrier when a registered member of staff scans the campus card or when a ticket is issued to a customer.
- Not open the entrance barrier or issue a ticket when the car park is full.
- Open the exit barrier when a registered member of staff scans the campus card or a customer inserts a paid ticket.
- Process the payment of tickets.
- Generate monthly bills to the registered staff.
- Show the car park status, e.g. is it full? How many spaces are left?
- Be updated with the tariffs and opening times to public.
- Record of maintenance: cash collections, fill up with blank tickets, etc by the operator.
- Be easy to use: i.e. the user should be able to use the system using common sense or with simple instructions.

You are asked to define detailed requirements, develop and test the above described software control system for the car park using Agile methods. Your design must be flexible and extensible, so that it can adapt to continuously changing business requirements and can be used in a general market. For example, possible future changes may include:

- Add more parking spaces.
- Print customer receipts.
- Give change.
- Accept notes and credit/debit cards.
- Automated number plate recognition.
- Generate a car park usage report.
- Store all transaction information.
- More than one payment station.

Your design of the campus cark park system must be capable of adapting to such future changes. That is, when developing a new car park 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.

2. 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.

3. 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.

4. Second submission: Final Report and Software

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

1. Summary of Responsibilities and Achievements (see the template on QMPlus)
2. Project management
    - The project management in your team working. E.g. using project management techniques, planning, estimating, decision making and adapting to changes.
    - A photo shows your team working.
3. 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.
4. 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.

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

    - Discuss the choice of any design patterns you have applied in your design and any design patterns that can be applied in the future to improve your design to adapt to changes.

5. Implementation and Testing
    - 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 control system, the entrance, the exit and the payment station. A possible example of this is illustrated in **Figure 1**:

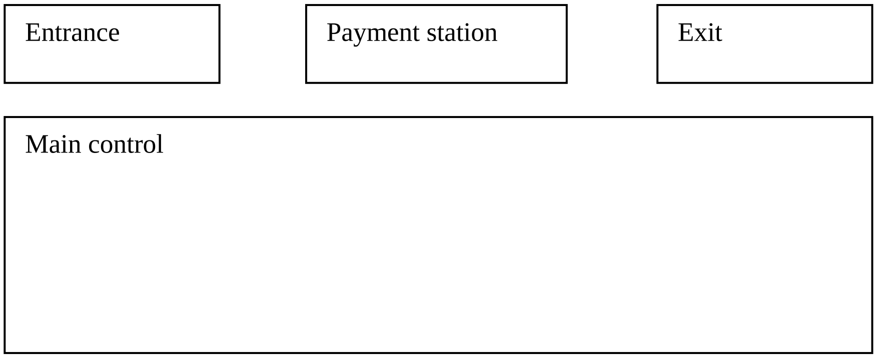

Figure 1

    - You only need to consider the logical information flow at the entrance, the exit and the payment station. For example, you may use a textbox to enter the ID of a campus card or a ticket (or use text files).
    - Discuss the test strategy and test techniques you have used in your testing.
    - Discuss the using of TDD.
    - A set of test programs using Junit.
    - All the code (product programs and test programs) should be pasted into the report's Appendix.

6. All reports should include an introduction, a conclusion and a list of references.
7. Main screenshots of the system should be included (**Note**: Convert them to JPEGs).

5. Submission Details

There are THREE electronic submissions to QMPlus:

- Product backlog in EXCEL format. This must be named **ProductBacklog_groupXX. xlsx**, where **XX** is your group number.
- Final report in PDF format (maximum 25 pages, excluding the Appendix). This must be named **FinalReport_groupXX.pdf**, where **XX** is your group number.
- Software: file stored in ZIP format containing all the .java files, Javadocs and a Readme file of your product programs and test programs. Do not include .class files,

as your programs will be re-compiled. This must be named S**oftware_groupXX.zip**, where **XX** 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.

6. 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 a SIMPLE Java GUI (AWT, Swing). Students should use J2SE 1.5 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. text) 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 course lecturers 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.

7. 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%**) 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.