**SEMESTER TWO EXAMINATION – MAY 2021**

**MAIN**

|  |  |
| --- | --- |
| **DEPARTMENT:** | Computing |
| **MODULE TITLE:** | Software Engineering: Concepts and Methods |
| **MODULE LEADER:** | Dr M B Özcan |
| **SUBMISSION DEADLINE:** | 08 May 2021 at 09:30 BST |
| **SUGGESTED DURATION:** | 2 hours and 10 minutes including reading time |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXAMINATION CONDUCT:**

1. The [University Academic Conduct Regulation](https://students.shu.ac.uk/regulations/conduct_discipline/Academic%20Conduct%20Regulations%202018-19.pdf) outlines the behavioural expectations of candidates completing any examination.
2. Students are responsible for ensuring that they know how to submit their exam script, when the deadline is and that they submit the script in enough time before the deadline expires. It is anticipated that Blackboard will be slower around submission times.
3. It is a fundamental principle that students are assessed fairly and equitably. The [University Academic Conduct Regulation](https://students.shu.ac.uk/regulations/conduct_discipline/Academic%20Conduct%20Regulations%202018-19.pdf) defines unfair behaviour relating to an examination to be 'cheating'. The University will investigate and may sanction any acts or behaviours which breach the Code of Academic Conduct.
4. In view of the University Covid-19 response and the exceptional actions necessary to conduct off campus exams, all students are reminded that this is an individual task and that students who contact or collude with other students to complete their exam may be subject to sanction later.

**INSTRUCTIONS TO CANDIDATES:**

1. Whilst the release period for this exam is a 24-hour period, this exam should take no longer than the above suggested duration to complete.  Candidates should be sensible about the amount of time spent completing and submitting work and make allowance for technical issues prior to the deadline.
2. Answer THREE questions out of FIVE on the paper. There are two sections: A and B. **You must answer ONE question from Section A.** You can answer the remaining questions from any of the two sections.
3. If you answer more than three questions, only first three questions will be marked, one of which will be a question from Section A (if attempted) irrespective of its order.
4. Academic support will be available in two 1-hour slots between 9:40 am to 10:40 am and then 4 pm to 5pm on 7th May 2021.
5. It is possible that candidates may encounter technical issues during the exam; if this happens the candidate should consult the below student guidance document on My Hallam which contains useful information on hints and tips, contact numbers and links to support: <https://www.shu.ac.uk/~/media/home/myhallam/Guides/student-exam-guidance.docx>
6. Any changes or clarification to the exam paper will be communicated via the module Blackboard site announcements. It is recommended that students monitor Blackboard announcements prior to submission of their final script but particularly in the first hour after release of the exam paper.

**SECTION A**

**Question 1**

The code in Figure 1.1 is a small part of one possible implementation of the **Meeting Scheduler case study**. It shows the code for a method named “GetDateConflict”, which is a member of a class named “MeetingRequest”.

The purpose of the GetDateConflict method is to determine if the responses of the potential attendees to a meeting request have produced a **Date Conflict**. The definition of a Date Conflict is taken from the original specification document:

“A [date] conflict is strong when no date can be found within the date range and outside all exclusion sets; it is weak when dates can be found within the date range and outside all exclusion sets, but no date can be found at the intersection of all preference sets.”

Consequently, the three possible results of the GetDateConflict method are “None”, “Weak” and “Strong”.

The GetDateConflict method has two arguments:

**attendeeResponses** is a List containing zero or more Response objects.

**possibleDate** in a MeetingDate object that contains the calendar date, start time and finish time of a proposed meeting.

A Response object encapsulates the Exclusion Set and Preference Set provided by each potential attendee. MeetingDate objects are used in the Exclusion Set and Preference Set of a Response object.

A Response object has two methods that are both used in the GetDateConflict method:

**IsExcluded:** has one argument, which is an instance of the MeetingDate class representing a possible meeting date. The return value is true if the possible meeting date is in the Exclusion Set of the potential attendee and false if it is not.

**IsPreferred:** has one argument, which is an instance of the MeetingDate class representing a possible meeting date. The return value is true if the possible meeting date is in the Preference Set of the potential attendee, and false if it is not.

For further information about the operation of the GetDateConflict see the comments in the code in Figure 1.1 where actual lines of code are in **bold** and comments are in *italics*.

(a) Draw a Control Flow Graph for the GetDateConflict method. Label the nodes in the graph using the numbers on the left hand side of Figure 1.1.

**(9 marks)**

(b) Calculate the Cyclomatic Complexity of the GetDateConflict method. Describe the method you used.

**(3 marks)**

(c) List all of the linearly independent paths through the GetDateConflict method. Use the numbered node labels on your Control Flow Graph produced in (a) to produce paths in the format 1-2-3-4-5-6.

**(4 marks)**

(d) For the following proposed meeting date, produce three test cases, that is, one for each of the three possible return values of the GetDateConflict method. For each test case you should specify a date range as a Calendar date, Start time and End time and state whether that date range should be in the Exclusion Set or the Preference Set of the Response object.

Proposed date:

Calendar date: 23 July 2020

Start time: 10:00

Finish time: 12:00

**(8 marks)**

(e) In order to implement automated unit testing using the “stub” type of mock object available in Microsoft Fakes, the Meeting Scheduler program would have to be redesigned. Explain how you would change the way Response class is used. Show how you would modify the one or more lines of code in the GetDateConflict method that would be affected by this redesign.

**(9 marks)**

**public class MeetingRequest**

**{**

...

**public string GetDateConflict(**

**List<Response> attendeeResponses, MeetingDate possibleDate)**

**{**

*// Set default value of date conflict*

[1] **string dateConflict = "None";**

*// Iterate over a responses from potential attendees*

[2] **foreach (Response response in attendeeResponses)**

**{**

*// Is possible date excluded by attendee's response?*

[3] **if (response.IsExcluded(possibleDate))**

**{**

*// Single exclusion means strong date conflict ...*

[4] **dateConflict = "Strong";**

*// ... so no point checking other responses.*

**break;**

**}**

**else** *// Possible date is not excluded.*

**{**

*// Is possible date NOT preferred?*

[5] **if (response.IsPreferred(possibleDate) == false)**

**{**

*// One not preferred means weak date conflict ...*

[6] **dateConflict = "Weak";**

*//... but do not exit loop as later response*

*// could produce a strong data conflict.*

**}**

**}**

**}**

[7] **return dateConflict;**

**}**

...

**}**

**Figure 1.1: Code of GetDateConflict method in MeetingRequest class**

**Question 2**

In the Meeting Scheduler application, potential attendees are asked to provide a response to a proposed meeting date. A meeting date consists of a **Calendar date**, **Start time** and **Finish time**.

A potential attendee’s Response consists of: zero or more dates on which the participant cannot attend known as the **Exclusion Set**; and zero or more dates on which the participant would prefer the meeting to take place known as the **Preference Set**. Items in both the Exclusion Set and the Preference Set also have a Calendar date, Start time and Finish time.

When processing the responses of potential attendees, the Meeting Scheduler must determine if any date in a response’s **Exclusion Set** overlaps with the proposed meeting date. If there is any overlap, then the proposed meeting request would be excluded.

The functional requirements that determine if there is an overlap between a proposed meeting date and a date in the exclusion set are:

* The Calendar date of the proposed meeting is the same as the Calendar date of the item in the response’s Exclusion Set.

AND

* The Start time OR the Finish time of the item in the Exclusion Set lies between the Start time AND Finish time of the proposed meeting date.

Start and Finish times are specified at 15 minute intervals, e.g. 12:**00**, 12:**15**, 12:**30** and 12:**45**. The minimum duration of a proposed meeting and of an item in the Exclusion Set or Preference Set is 1 hour.

(a) Carry out a Boundary Value Analysis to determine if the date of an Exclusion Set item overlaps with a proposed meeting date. This analysis should involve the Start time and Finish time of an Exclusion Set item and the Start time and Finish time of a proposed meeting date. Assume that both have the same Calendar date. Also, assume that the time for each item is valid, that is, the Start time is always at least one hour before the Finish time.

**(9 marks)**

(b) Produce a Test Plan based on the Boundary Values listed in (a). Specify the Start time and Finish time of a proposed meeting. For each test case specify a Start time and a Finish time of an Exclusion Set item. Also, for each test case state the expected result.

**(12 marks)**

(c)

(i) Give an illustration of an aspect of user interface for a meeting scheduler that will suit using a Coded UI testing tool.

**(6 marks)**

(ii) Write a description of tests using a Coded UI Test in Visual Studio. This description should be in the form of a list of the tasks necessary to create the test in Visual Studio, record the test, run the application, for the scenario you illustrated in part (i) above.

**(6 marks)**

**SECTION B**

**Question 3**

1. Consider the Secure Car Park System outlined in the **Appendix**. Assume that it is decided that a rapid prototyping approach is needed to clarify vague aspects of this system before commencing production quality implementation. In this context, you are required to do the following tasks:

(i) Prepare FIVE pertinent questions that can be asked to users during requirements validation about the payments, chip-coins and their use, car security and car park space aspects of the system.

**(5 marks)**

(ii) Prepare TWO user stories, together with their acceptance tests, which will be used as the basis of developing a rapid software prototype.

**(12 marks)**

(iii) Produce a brief design, with the aid of diagrams or sketches, of a low-fidelity rapid software prototype. Your plan should clearly show sample prototype screens.

**(10 marks)**

(iv) Briefly discuss how you would carry out the requirements validation process using the artefacts you produced in (i), (ii) and (iii).

**(6 marks)**

**Question 4**

1. Consider the agile development process model and briefly discuss how it handles a significant change in requirements late in the development of the Secure Car Park System outlined in the **Appendix**.

**(10 marks)**

1. The software industry has been subject to dramatic changes since the 1970s. In his Stevens lecture, Wasserman[[1]](#footnote-1) summarised these changes by identifying several key factors that have altered software engineering practice. One such key factor is *criticality of time-to-market for commercial products*.

Discuss this particular factor and its impact on the overall software quality.

**(10 marks)**

1. Software Engineers sometimes strive to produce "zero defects" programs. In doing this they may not be satisfying the needs of the customer.

In light of your answer in (b), what factors should the software engineer consider when deciding whether to release software containing bugs?

**(13 marks)**

**Question 5**

a) The Common types of architectural views of a software design include:

* Decomposition view
* Dependencies view
* Generalization view
* Execution view
* Implementation view
* Deployment view
* Work-assignment view

Apply any four of the decompositions views to the assignment's case study, **"A Meeting Scheduler System"**, and give a ranking on the importance of these four in the context of software design. **(16 marks)**

b) Explain how you may design and conduct an empirical study to investigate the effectiveness of two alternative designs for a particular aspect of **"A Meeting Scheduler System".** **(17 marks)**

**Appendix**

**THE SECURE CAR PARK SYSTEM: PRELIMINARY DEFINITION**

This preliminary description is deliberately intended to be sketchy. Acquisition, specification and validation processes are needed to complete it and lift any shadowy areas.

This system is expected to simulate chip-coin operated car parks in Sheffield city centre. Each car park will have one entry and several exit points. Entry and exit points are equipped with a barrier and a chip-coin machine. Drivers who wish to use a car park approach its entry point and collect a chip-coin situated next to the entry barrier. Furthermore, drivers are expected to pay fully before they leave a car park using pay machines, which may be situated at various points in the car park. Car parks' charging structure may vary, but will be displayed at entry points, as well as next to pay machines. In addition, some car parks may operate different discount schemes whereby drivers who are employees of specific employers may get discounts negotiated between car park managements and employers.

Capacity and available spaces of each car park will be displayed in big electronic screens throughout Sheffield city centre to enable drivers to make informed parking decisions.

Finally, some car parks can offer "secure parking". There may be different ways of implementing this aspect both in hardware and software terms. Consider one such scheme, which is currently being operated at a car park in Derby, in which each parking bay has a built-in sensor on the floor. Once a driver parks his/her car over a secure bay, s/he has to enter the bay number along with a password on a separate device at exit. Once this is done, his/her car is said to be "secure" in that unless the car's security is released on return to car park by entering the same bay number and password, alarm will sound and exit barriers will be locked if the car is moved out of the bay.

**END**

1. A. I. Wasserman, "Toward a discipline software engineering", IEEE Software, 13(6), 23-31, 1996. [↑](#footnote-ref-1)