University of Westminster
Department of Computer Science

6COSC001W Enterprise Application Development – CWK 1 (2019/20)

Module leader	A. Basukoski
Unit	Coursework 1 – Analysis design and evaluation.
Weighting:	50%
Qualifying mark	30%
Description	Use of UML for the analysis and design of a domain problem. A report on the design decisions and the suitability of the design for implementation.
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): - LO1 Demonstrate a thorough understanding of the design and implementation of server-side applications, together with a critical - LO4 Assess how software quality issues impact on software design. - LO7 Autonomously manage a small project with respect to time and task management and be able to critically evaluate personal performance.
Handed Out:	13th October 2019
Due Date	11th November 2019. Submission by 1:00pm
Expected deliverables	PDF report uploaded to Turnitin as pdf file.
Method of Submission:	Electronic submission on BB via a provided turnitin link close to the submission time. The file you upload should have the following naming format: <Cw1_StudentNumber_FullName.pdf> E.g. Cw1_w123456_JohnSmith.pdf
Type of Feedback and Due Date:	Feedback and marks 15 working days (3 weeks) after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.

Assessment regulations
Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission
If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Coursework Description

Problem Statement

You are to design a time management and reporting tool for personal use. This tool will enable you to enter events for a given date and time, as well as contact and location details if needed. Events may be one-off or recurring and will be either appointments or tasks. You should also provide at least a weekly view showing the events for a given week. Finally, you should have an option to produce a report that predicts the time usage for the following four weeks. You may choose how you do the prediction; one way is by extrapolating the weekly time usage from previous weeks.

Your task is to develop a design document for this application. You are also to produce a report describing the reasons for your design decisions, and how suitable you deem your design is for implementation. The design document will consist of these sections:

Part A – Requirements

State the software/system requirements for the system using this style:

R.1. The software shall allow the user to insert a new contact.
R1.1 The software shall allow the user to enter name.
R1.2 The software shall optionally allow the user to enter an address.
...
R2. Allow the user to enter an event.
Description: For each we might put a sentence or paragraph further describing the requirement if it is unclear**.**

... **and so forth. See further examples in appendix A.**

Record as many as you can. These are requirements so do not discuss how it shall be done just what needs to be done.

Marks Available 10%.

Part B – Use Case Diagrams

1) Draw one or more use case diagrams for the system.

Note: The use of correct UML is important here.

Marks Available 5%.

2) For each use case – create a use case description for the main flow (see

Appendix B **for the template on which to base these)**

Marks Available 10%.

Part C – Classes

1) In consideration of your use case descriptions, discover as many classes as possible and record these in a CRC (Classes, Responsibilities, and Collaborations) table. (See **Appendix C** for a template).

Marks Available 10%.

2) Draw a domain model for the system (**see lecture notes for an example**). You do not need to put in any attributes or methods but you must label all associations and use correct UML when describing generalisations and aggregations etc.
Marks Available 15%.

Part D – Collaboration

For each Use Case draw an analysis sequence diagram **(see** Appendix D **as an example) using the Model View Controller as a design pattern.**

Marks Available 15%.

Part E – Activity

Draw a *detailed* activity diagram to show the calculation of the prediction algorithm you will use. This should be detailed enough for you to produce a coded algorithm.

Marks Available 15%.

It is important that you be as concise as possible with your design as you will be required to implement and deploy your design for the second coursework.

Part F – Report

1. A 500 to 1000 word report on the reasons for the design decisions you took. 10%

2. A 500 to 1000 word evaluation of the suitability of your design for implementation. Think about how you would react if you were the team leader and someone gave you this design for implementation. Is it detailed enough, clear enough. What are its strengths and weaknesses. 10%

Marks Available 20%.

Coursework Marking scheme

The Coursework will be marked based on the following marking criteria:

Criteria	Mark per component	Mark provided	Comments
PART A Requirements	**10**		
PART B Use cases			
- Diagram	5		
- Descriptions	10		
PART C Classes			
- CRC	10		
- Domain model	15		
PART D Collaboration	15		
PART E Activity	**15**		
PART F Evaluation			
- Design decisions	10		
- Suitability for implementation	10		
Total	**100**		

APPENDICES

Appendix A

This tool will allow an individual to keep a record of their events and also use the tool to predict time usage for following four weeks (month).

User Interface

This will be built as a .NET C# windows desktop application. You are free to design the user interface as you wish but here are some recommendations. The interface **will have a number of views**:

1) A contacts view for entering and updating the details of contacts.
2) A view for entering and updating event and task details.
3) A weekly view of appointments and tasks.
4) A view that enables the user to see the predicted time usage over next four weeks.

It is up to you how you design your forms. We are purposely not giving you a design example to avoid everyone having the same design. You are advised to create mockups and storyboards and modify them iteratively as you develop your design document.

Your design decisions may be included in your report.

Storage of run time data

The data for the events will be created by a view that allows the specification of the events to be entered for a date, and this should be a programmatic dynamic interface. Once the user has finished you need to save the event data as an XML file OR in a database of your choice. When the application is run again (after closing) the system shall use the XML or Database data to populate the data on your interface.

Appendix B

Use Case Scenario Template

Use Case: **<Enter Use Case name here>**
<Enter a short name for the Use Case using an active verb phrase.
e.g. Withdraw Cash, Register Customer, Rent Video, Calculate Sales Tax, etc.>

Id: UC--<Enter value of Id here>
Note: Enter a unique numeric identifier for the Use Case. e.g. UC-001>

Description

<Enter description here>

Note: Briefly describe this use case.
e.g. User wished to either update or enter a new event.

Primary Actor

<List the Primary actor here>

<List the Actor who's goal is being satisfied by this Use Case and has the primary interest in the outcome of this Use Case.
e.g. Store Clerk>

Supporting Actors (if any)

<List supporting actors here>

<List the Actors who have a supporting role in helping the Primary Actor achieve his or her goal.
e.g. Customer, Store Manager>

Stakeholders and Interests (if any)

<List Stakeholders and their interests here>

<List the various entities who may not directly interact with the system but they may have an interest in the outcome of the use case. Identifying stakeholders and interests often helps in discovering hidden requirements that are not readily apparent or mentioned directly by the users during discussions.

Pre--Conditions

<List Pre--Conditions here>

< List the system state/conditions that must be true before this Use Case can be executed.
e.g. Must have created at least one event.>

Post Conditions

Success end condition
<List success end condition here>

Note: Enter the successful end condition of the Use Case where the Primary Actor's goal is satisfied.
e.g. Event date has been updated.

Trigger (where needed for dynamic views)

Note: The event that starts this Use Case>
Example: the user has updated an event date.

<List Use Case trigger here>

Main Success Scenario

Note: Enter the Main flow of events. i.e. The steps narrating/illustrating the interaction between Actors and the System. Describe Actor's actions/stimuli and how the system responds to those stimuli. Describe the 'happy path/day' scenario, meaning the straight and simple path where everything goes 'right' and enables the primary actor to accomplish his or her goal. Main flow/path should always end with a success end condition.

1. User selects add event.
2. User enters date.
3. User enters time.
4. System displays event summary.

Variations

<Enter variations here>

A variation would be if the user enters an invalid value and the software would then warn and not allow entry until the user entered an appropriate value.

Note: For the coursework you don't have to be exhaustive, it will be sufficient if you include major variations to show your understanding.

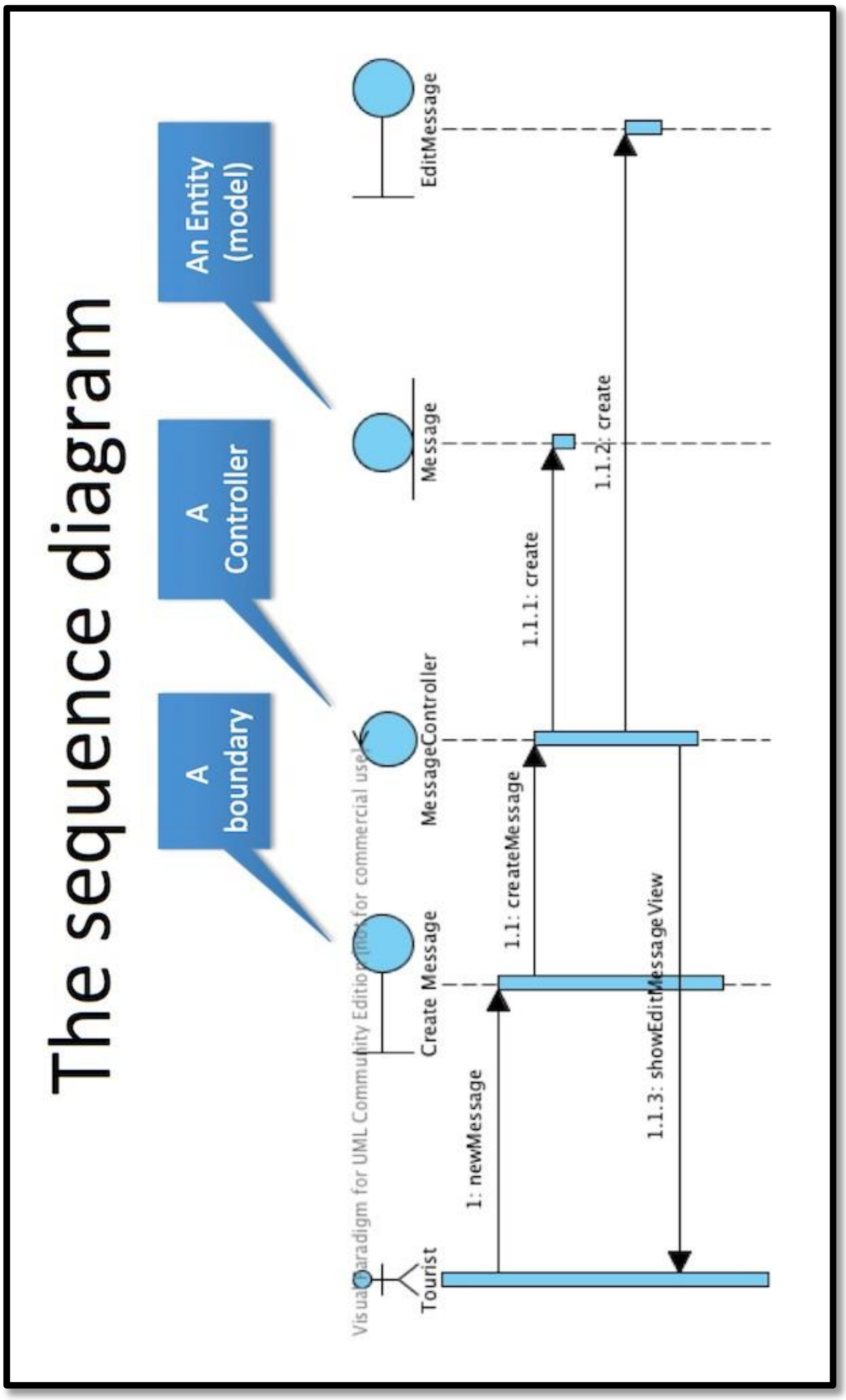
End of template

Appendix C

Class Name	Type	Responsibility	Collaborations
Contact	Model	Holds all Contact details properties such as name, address ...	Event
ContactView	View (Form)	View for entering and updating Contact details	Contact, Event

This an example CRC

Appendix D


The sequence diagram
A boundary
A Controller
An Entity (model)
Visual Paradigm for UML Community Edition [not for commercial use]
Tourist
Create Message
MessageController
Message
EditMessage
1: newMessage
1.1: createMessage
1.1.1: create
1.1.2: create
1.1.3: showEditMessageView