CO2401 Software Development (Part 2) 2014/15

Group Assignment Specification v1.0

Issued: January 2015

Overview

The assignment for the second part (Nick's bit) of CO2401 assesses your ability to carry out various software development activities in order to produce an engineered solution to a problem. It involves designing, testing and evaluating a solution. The tasks are split into two stages and some investigation is required, so you should plan the work carefully.

Working successfully as a team is a key requirement for working in industry, and for this assignment will be working in the same small groups* as for your Professional Skills project.

Using the specification and implementation of the **software your team is already developing for Professional Skills*** as your case study, complete the activities described in the next sections to produce the required design and test documentation.

***** If you are **not** working in a team of Software Engineers or Games Developers on a project for Professional Skills, you should refer to the alternative scenario and will work in a different team. Please look on Blackboard for the team lists and separate alternative scenario documentation.

You must develop the solution to the assignment mainly in your own time.

In keeping with the practical industrial scenario, there will be milestones set for deliverable components of the assignment; the difference in this case is that rather than the company taking a substantial financial penalty for a missed milestone, you will be penalised with a percentage reduction of your assignment mark.

The stages (and deadlines) of the assignment are as follows:

1. System Design (Deadline: **18th February 2015**, **5.00pm**)
   (deliverables to include: detailed UML model of your software – see over)

2. Testing Strategy (Deadlines: **18th March and 29th April 2015**, **5.00pm**)
   (deliverables to include: test strategy document and test cases – see over)

At each stage you should collate the various components and upload a **single .pdf** document for the **team** by the deadline. On the Assignment cover sheet, write your **team name/number** and the **names of each member** of the team who has contributed.

After the second stage, you must also **individually** complete an on-line peer-assessment (quantifying the contribution to the project of your team-mates).

Please refer to the separate guidance notes on the submission of assignment work, which are available on blackboard in the computing noticeboard section.

Team Organisation – Important!

This is a relatively lengthy and demanding assignment, where you will need to work well together as a team – frequent and reliable communication is key to achieving this. Each team is responsible for organising itself into an appropriate structure, and I would suggest that you begin by finding out exactly *what* needs to be uploaded *when*, and decide who will be responsible for making sure that happens correctly for both deliverables of the project.

Remember, late or missing deliverables will incur a penalty for the *whole* team…
Other important roles include organising the communication process within the team, co-ordinating the sharing and updating of documents, scheduling, chairing and recording meetings.

As a team, I expect you to behave professionally: **Record minutes of all your meetings** (which should happen at least once a week during term-time), and **keep a record of the work each team member does** each week. These records are not *directly* assessed, but will be used as evidence in determining how well you have worked as a team. They should therefore **be uploaded to Blackboard each week in the form of a single page report**, titled "CO2401 Team <*number*> Report: Week <*number*>". At your first meeting you should agree between yourselves a process for signing off and uploading the weekly report.

A separate document providing advice on the recording of team meetings will be available on blackboard.

A brief peer-assessment of who has contributed what to the assignment must be produced by each team member after the final hand-in.

You should allocate work between the team members fairly, with the expectation that each member of the team will receive the same mark (although I reserve the right to vary individual marks based on the information as recorded above). If one team member is failing to make an appropriate contribution I would expect to see this identified in the weekly reports, and evidence in the minutes that action to re-allocate tasks had been taken so that the team as a whole does not suffer.

Note 1: If there are any particular issues with your *specific* Professional Skills software product that lead your team to believe that it is not possible to meet all the requirements of this assignment, please discuss them with the CO2401 module leader at the earliest opportunity.

Note 2: It is assumed that by the end of the assignment you will have some software on which to run the tests you will design in Part 2*. The software you create is not being assessed (for CO2401, at least). It is not necessary, therefore, for you to have a functioning software application in order to pass this assignment, although the maximum mark your team can achieve will necessarily be restricted if you do not have software to test.

*If you are following the alternative scenario you will **not** be expected to have working software, and are therefore exempt from completing deliverable 2 of Part 2.

Part 1: System Design

Deadline: Wednesday 18th February 2015, 5.00pm

Your Team must produce detailed software design documentation for the system described in the appropriate scenario (see above). The design artefacts outlined below must be uploaded together (as a single pdf document) by the deadline. **Take care over the organisation and presentation of your work: this is a professional report.**

Your Software Design documentation should include:

a) Initial analysis of the requirements at the Domain level, comprising:
   i. A use-case diagram, identifying the actors who interact with the system, and with a brief textual overview of each use-case (1 paragraph per use-case).
   ii. Detailed step-by-step descriptions of **at least three** of the (more complex) use-cases - identifying the actors involved, including pre- and post-conditions, and with alternative and exceptional flows where appropriate. A list of candidate classes resulting from performing noun-phrase analysis on the requirements definition.
   iii. A list of domain classes derived from your list of candidate classes, including a (one-paragraph max.) description of (or justification for choosing) each class.
   iv. A *domain* class diagram showing the relationships (with multiplicities) between the classes identified above.

b) A fully detailed *design* class diagram featuring your original conceptual classes plus any others necessary for implementing the system. You should include a CRC card for each class which appears on the class diagram.

c) A sequence diagram for scenarios based on *each* of the 3 use cases described for part a, and consistent with the class diagram in part b.

   Hint: *Make sure that the messages (with parameters and return types) on the sequence diagrams match the methods on the design class diagram, and that the direction of calls from one class to another on the sequence diagrams are consistent with the navigability shown on the class diagram.*

   * *Where appropriate, alternate and exceptional flows may be shown either on the main diagram, or on separate sequence diagrams.*
   ** *If you have updated the detailed Use Case description since Phase 1, please attach the most recent version as an appendix to your submission.*

d) A state transition diagram (STD) for **two** of the more complex classes in your system. (You should choose classes which nave at least four states if possible.)
These STDs should be consistent with both the sequence diagrams from part b), and the class diagram from part a).

   Hint: *Make sure that events on the STD match the operations defined for the class on the class diagram, and that they happen in the order allowed by the sequence diagrams*

e) A detailed configuration management plan for the entire project.

Part 2a: Testing Strategy

Deadline: Wednesday 18th March 2015, 5.00pm

1. Your team must produce a test plan and suite of test cases for the system described in the appropriate scenario **as designed in your team's previous submission**. Your team should hand in a single test document which contains the following parts:

    a) An overall test plan for the system (*Guide: 1 - 2 pages*) outlining your company's strategy for Verification and Validation testing at each phase of the development of the system. *(Hint: You should include a strategy for Unit Testing, Integration Testing and System/Acceptance Testing.)*

    b) A comprehensive, numbered, set of appropriate Unit tests for **one** module, chosen from those identified in the test plan. For each test, you should give a single sentence justification for the test, and an indication of the expected result. The tests should make sense in the context of your detailed design submitted at Phase 3 of the assignment.

    c) A set of Integration tests, designed to verify the inter-working of the module chosen for part b) with a second module, according to the test plan. (You may assume that this second module has been Unit tested separately.)

    d) Brief details of any Stubs or Test Harnesses which will need to be developed in order to facilitate the Unit Tests and Integration Tests you have designed.

    e) A set of **three** interactive test scripts, based on the use-cases described in detail in Part 1 of the assignment, which would be suitable for customer acceptance testing. (You are not expected to cover the functionality of the complete system with these tests.)

Part 2b: Test Results and Review

Deadline: Wednesday 29th April 2015, 5.00pm

2. Formal test results from running all the tests designed in parts 1 b) c) and e) above, on your completed application. **Note:** *Your tests do not all have to pass!*

3. Your team should submit a separate report which reviews the overall development process your project has gone through. Comment on the appropriateness of the strategies your team undertook during each Phase, in the context of the life-cycle your team has followed. Briefly discuss what you would do differently in the light of your experiences, were you to work on a similar project in the future. (*1000 words max*)

4. Each individual member of your team must complete the pro-forma (electronic, via Blackboard) outlining their contribution to the project across both parts. This can be completed any time between 18th March and 29th April.

Learning Outcomes and Indicative marking criteria

The following learning outcomes are assessed:

L.O.3: Design a software solution from a requirements specification

L.O.4: Produce a robust test strategy using appropriate software testing methods

Part 1

3rd: A simple UML design class diagram showing at least 5 classes.
Simple sequence diagrams for the specified use cases, involving some appropriate objects.
Simple state transition diagrams for the appropriate classes, one of which is showing at least 3 states.

2nd: A complete and structurally correct UML class diagram showing properties and methods at the appropriate level of detail.
Complete and structurally correct sequence diagrams for every use case in the system.
A structurally correct state transition diagram for a complex class, showing a plausible life cycle.

1st: A complete, detailed and structurally correct UML model in which all the diagrams are consistent with both each other and the requirements appropriate to the given scenario.

Part 2

3rd: *Test Plan:* A basic testing strategy which tests key components and illustrates an understanding of testing. Outline of appropriate test harnesses / function stubs etc., to support the strategy. Identification of some suitable tests for each level of testing.
Evaluation: A reflection of your team experience identifying strengths and weakness of each component and the assignment as a whole. Identify problems that arise from the lack of an appropriate specification or user considerations.

2nd: *Test Plan:* A test strategy which is justified according to the requirements, which details of test harnesses and/or other methods of testing which illustrate test repeatability. Reasonably complete set of unit and integration tests for the modules identified.
Evaluation: Alternative improved approaches to each of the Phases discussed if appropriate.

1st: *Test Plan:* Comprehensive test strategy, including consideration of test tools which automate the testing process to minimise interaction with the tester. Full requirements traceability from the specification to all levels of testing. Tests carried out and results recorded appropriately.
Evaluation: All methods discussed underpinned by published work

Note: to get a 1st, it implies that the 3rd and 2nd criteria have been satisfied.