Masters of Computer and Information Sciences

Semester 2, 2018

ASSIGNMENT 2
Team Development Practice
Contribution to final marks: 60%

Based on the discussions in class, the following decisions about the assignment are documented

Due dates:
Final product available via the cloud to Client before
Wednesday Week 12, October 17th 11:59pm .
Product Increments available at the Sprint Review on the Friday at the end of every sprint

Final Academic Deliverables (evidence portfolio and Team Reflective Report) available before
Friday Week 12, October 17th 11:59pm .
Evidence log Increments available by Friday EVERY WEEK
Final report increments available at the end of every sprint at the sprint reviews

See separate planning document for more detail on expected process and sprint deadlines

Team Effort
This is a team assignment to be managed and submitted as a Team of 3-4 people.
All Team members will receive the same mark for this assessment unless it is indicated in the Team Participation Form that contribution or participation was not equal.
The lecturer may also negotiate an unequal assignment of marks under certain circumstances.
Please upload to the team Evidence Wiki on Blackboard a completed AUT team participation assignment cover sheet. THIS SHOULD BE SIGNED BY ALL TEAM MEMBERS. 100% is the benchmark. Lower than 100% means this team member did not participate equally in the group. So someone with 50% participation contributed half as much effort as someone with 100%.

Submission Requirements:
The evidence that will be evaluated for your grades in each section below will be based on the evidence from:

1. Your Week by week Evidence Log
2. Your Team Reflective Report and how this evidence is used to explain your team processes and capabilities and leaning.
3. Analyzing your use of GitHub and the code in GitHub
4. Analysing the use of a user story/task board
5. Analysis of Slack conversations
6. Evaluation of other documents/artifacts
7. Your sprint reviews
8. Your product increments
9. Observations of your team work

Areas of evaluation
1. Requirements Quality (10%)
2. Planning and Monitoring including risk (10%)
3. Technical capability with tools Git/GitHub, TravisCI, Heroku (10%)
4. Technical capability with technology stack (front and backend) (10%)
5. Quality assurance and testing process including code quality and design (15%)

6. Use of an Agile way of working (10%)
7. Collaboration and Performance as a team (10%)
8. The product quality (25%)

Electronic Evidence Log (20%).

A team log summarising all team work **and short reflections** on this work **for each week in each sprint**. This should include the following evidence with some annotations (written words!) describing what is going on, any learning and other comments. The material in your team log will be used to write your team reflective report (see next section).

My preference, for ease of access, is that the log is maintained on Blackboard in your team Wiki.
If you are going to use Google Docs or similar please have a different folder for each week. Also please invite jbuchan42@gmail.com so I can view it. MAKE SURE IT IS CLEAR WHAT TEAM NUMBER YOU ARE.

The content of the log should include photos, documents, screenshots and descriptions/reflection such as that below
ORGANISE this INTO A WEEK BY WEEK STRUCTURE (Sprint 1, W1; Sprint1, W2; Sprint 2, W1; etc)

- Photo or screen shot of the story board at least at the start and of each week. Describe any changes and the product backlog order and info on user story cards.
- Photos/documents showing DoD,
- Sprint goals for each sprint (1 sentence - related to the high level feature you will deliver)
- A Team agreement, NO TEAM ROLES.
- Photos of the sprint planning at the start of each sprint
- Photos of the burndown chart at the start and end of each sprint
- Photos of whiteboard drawings or documents showing how the design of the product has changed
- Photos of face-to-face and screen shots of electronic **daily planning** meetings (like stand-up meetings - answering 3 questions)
- Photos of setting up and use of tools - screen shots, configuration files, instructions. This includes IDEs or text editors, Git/GitHub, TravisCI, web frameworks, testing framework, cloud deployment tools, Docker, Slack, electronic story board, bug management software etc....
- Photos or pdf files showing any product mockups and feedback and changes.
- Photos of sprint reviews (one per sprint) and a discussion of learning and changes from a product perspective
- Photos of the results of doing acceptance tests
- Photos of doing retrospectives (one in sprint 1 and one in sprint 2) including a list of NEW things to try in the next sprint and why and how you will know if they are better.
- Photos of any other meetings
- Photos of meetings and emails/Slack correspondence with Patricia the PO.

Your lecturer should also be invited to have access to:
- Your team Slack channel
- Your team electronic story board
- Your team GitHub

Team Assessment Report (wiki style) (50%)

Information in the week by week team log will be used to iteratively create a final reflective team report INCLUDING EXAMPLES and evidence. The report should include a description of what you experimented with and what you learned in the following areas of software development. You should demonstrate improvement and capability with examples and evidence. The content of the report is as follows.

Iteratively Understanding what to build (Requirements Management)

1. Explain how you **User stories** are good quality and were used to capture user requirements - including who it's for and why they want it (the value/purpose to for the user)
2. Evidence shows user stories and how some changed over time and why.
3. Explain how the user stories are testable with examples of Acceptance tests.
4. Discuss how A mock-up is used at least once during product development to verify understanding of the requirements.
5. Short descriptions of the main user types and what characteristics are important and how this might affect your application design or requirements.
6. A list of non-functional (quality) requirements is shown with a description of how they were incorporated into the product and how they were tested.

Forecasting commitment to deliverables and checking on progress (Planning and Monitoring)

1. How did the use of a physical story board compare to the use of an electronic one? Any impact on team development?
2. Evidence of use of techniques to forecast what will be delivered by the end of each sprint. Discuss the pros and cons of each.
3. the meetings? Was it useful to timebox meeting?
4. Evidence it was done at the start of the sprint
5. Description of the process with photos
6. Results of the process (forecasts)
7. The actual work compared with the forecasted. Comments on differences
8. Photos of all meetings and the purpose and outcome of the meetings.
9. How did you keep track of time in
10. Explain how you monitored progress with evidence of monitoring (e.g. photos of story board, burndown charts)
11. Show the **roadmap** at the beginning of the development and explain how it changed during the 3 sprints and why

Thinking about what could go wrong and trying to avoid it happening or limit the impact if it does (Risk management)

1. Describe a list of the Top 3 risks that could jeopardise successful product development (in the eyes of your team and PO) and how you will reduce the chance they could happen and/or the impact they would have on re-work or quality of the product.
2. Discuss how these were updated during development. Show examples.
3. Were any examples of impediments or blockers identified during the "daily" planning "standup meetings"? Give examples and how they were resolved.
4. How was the risk of misunderstanding the PO lessened? Give examples.

Working Collaboratively

1. Explain how useful (or not) the daily communications and planning meeting was to help the team coordinate and collaborate, with evidence.
2. Explain how GitHub was useful for team collaboration? Give examples.
3. Explain how you got participation of all team members at meetings? Give examples.
4. How did it work with no team roles? In what ways were you a self organising team? Give examples.
5. How did you get agreement on decisions? Give examples.
6. How did you handle conflict and disagreements in the team? Give examples.
7. Explain the Team agreement and if it was useful to collaboration and if it changed and why.

Product Architecture

1. Show evidence of separation of data and business rules and user views. (e.g. MVC structure)
2. Explain the product architecture and how the Technology stack used implemented it

Code craft

1. Show examples of code that show **intentionality**
2. Show examples of code that show Small methods
3. Show examples of code that show Small unit tests
4. Explain what code standards were used and show examples of code that show Coding standards

Capability with Tools

Show evidence of the use of all tools used including screenshots and set up instructions and configuration files
Especially

1. Git/GitHub
2. TravisCI
3. Unit testing (e.g. Mocha) and other automated testing
4. Heroku
5. Use of Trello or Asana (or similar) for user stories and acceptance tests

The product quality

1. Describe the use of unit testing and how it was useful. EXTRA MARKS for showing evidence of Test first development
2. Describe your continuous integration process with evidence including a description of setting up TravisCI
3. Describe your review process for code quality - including pull requests in GitHub
4. Describe how you treacked bugs or other issues
5. Describe how you tracked whether tests passed or failed and when they were re-tested

Use APA 6th reference formatting for citing others' published materials. (APA_6_AUT in Endnote)

Include your class (SDM2018) and Team Name in your naming of the User Story Boards and GitHub and other documentation to make it easy to allocate the marks to the right Team!

Purpose of Assignment
This assignment relates directly to the following course learning outcomes:

3. Critically assess, compare and contrast the distinguishing features of a variety of software development approaches and methods
4. Recommend and justify the selection of development approaches, methods and practices across the full range of development activities for different development contexts
5. Use a selection of industry standard models, tools and techniques that support development methods
6. Critique research literature in the area of software development methods
8. Apply some software development methods and critically reflect on the experience.

Aims

The aim of this assignment is to develop a software application as a vehicle to practice and reflect on methods and techniques related to team-based software development across the full development lifecycle. This problem-based approach motivates some of the learning expected in the course as well as applying some learned methods and tools. This includes project management and organisational methods as well as technical methods.

In summary the aims are:

- To gain experience in the full development lifecycle of software development from idea to implementation using Agile practices.
- To gain an overview of the common practices and tools used in commercial Agile software development.
- To understand some of the benefits and challenges of commercial Agile software development
- To contribute to a research project for the Software Engineering Research Lab (SERL) at AUT

Note that you are assessed more on evidence of progress and learning related to the (eventual) well-executed application of methods rather than the final software product.

You should communicate frequently with your lecturer (assessor of the process and evidence) to understand the requirements of the evidence portfolio and process, as well as the Product Owner (assessor of your delivered product) to understand the requirements of the product.

Some constraints on the process and tools are imposed to simplify the experience and assessment complexity. The learning should focus on the principles associated with these methods and tools that are transferrable. For example, incremental, iterative development is well-evidenced good practice and can be implemented in a number of ways. We are using Scrum as a framework to manage this.