Software Engineering and Project
Group Project Specification

David Milanese
david.milanese@adelaide.edu.au

August 2018
Version 2.0.1

1 Introduction

This document is prepared as a precursor to the client specification to outline the requirements of the group project you and your group will be participating in or this course. In this document you will find the procedures and assailable components for the group project.

2 Software

For the group project your group is *required* to produce the code for the EV3 robot. We strongly recommend using LeJOS - a Java programming environment for the Lego Mindstorms EV3 robots. This course has had prior success with LeJOS. Should you wish to use alternatives - do so at your own risk, and make sure you do your research before committing to it.

You will be required to provide documentary evidence that you have fully complied with any licensing requirements associated with any code that has been re-used. Copyright re- lease documentation is required from the copyright holder of any re-used code, unless that code is specifically designated as free for use. However, the reused code (excluding code from the platform/framework) cannot be more than 10% of your total project. **It is considered an academic offence if you exceed this 10% limit.**

One of the requirements for the system you are producing is that it has no encumbrance to use - which means all the licensing details *must* be sorted out. If you wish to use some code and cant find the required licensing information, you *must* assume that the code cannot be used (the default copyright protection is that code is owned by the author).

3 The Software Process

Your group will be practicing SCRUM in this group project. You will learn more about SCRUM in the first week of lectures. SCRUM is widely accepted in software and other industries. The SCRUM process will operate in 2 week intervals called *sprints*. Each sprint will finish on the day of your tutorial session, *please refer to the lectures, MyUni or Access Adelaide for your tutorial session.*

4 Assessments

This course has a number of *group* and individual assessment components. *Please see the course outline for the course weightings.* Some of the assessment will occur during the client meetings/tutorial session *(see section Meetings)*. This is intended to give you and your group feedback as quick as possible, so be prepared for it. Other assessment and feedback will be made available via *MyUni*.

4.1 Milestones

During your group project your group will have 2 milestone deliverables due. Prior to each milestone, you will be required to negotiate with the client what your group aims to deliver. The first milestone negotiation will occur on *week 6*, then delivered the following week. The second milestone negation will occur on *week 8*, and also delivered the following week. The negotiations and deliverables will be performed during their respective client meetings. To summarise:

1. Milestone 1 negotiation on **week 6** and delivery on **week 8**
2. Milestone 2 negotiation on **week 8** and deliver on **week 9**

4.2 Progress Reports

You will be responsible for ensuring your group delivers what is required, on time and on budget (in this course, time is budget). To this end, we *require* that you generate and document regular progress data. This is in-line with common industry practice – managers need to know how many hours you've worked on projects.

You and your group are *required* to submit a **2** small reports at the end of the sprint via *MyUni*:

1. An individual report detailing your progress, successes, concerns and problems.
2. A group report listing contributions made by each member.

These reports will be used by the teaching staff to monitor and assist you in your groups progress. The reports should take no longer than *15 minutes* to complete and count towards your course grade (*see the course outline for the weighting*).

Please note that when you submit the second report you are acknowledging that the entire group agrees on the individual contributions. If a group member has an issue with the report, it is to be resolved before submission. We will not accept group members disputing their contributions at the end of the course. If you do not agree with the contributions, raise them early and they can be resolved before the marks are finalised.

4.3 Documentation

In this course your group will produce Software Project Management Plan (SPMP), Software Requirements Specification (SRS), and Software Design Document (SDD). Each of these documents will have a draft due date and a final submission due date. The draft due dates are listed as follows:

- SPMP draft due **week 4**
- SRS draft due **week 5**
- SDD draft due **week 6**

Please see MyUni for precise due dates and times.

The draft submission should show your group has thought about the sections in each document, although they may not be structured or complete. **The final submission of each document will be at week 12.** Each of these documents are intended to be *evolving*. It is your responsibility to make sure they are maintained. *Please be aware that leaving the final changes to the documentation till week 12 will result in a large overhead* - keeping in mind you will be studying for your other courses.

4.4 Presentation and Demonstration

At the end of the course your group is *required* to present your group project to the teaching staff on your **process** and demonstration of the **product**. *Please see the course outline for the weightings.* Each group will be given 30 minutes to present their project. We recommend 15 minutes for the presentation, 5 minutes for Q&A, and setup and 10 minutes for the demonstration. This is your time, so structure and plan it accordingly. It is not required that each team member speaks in both parts of the presentation, however, each team member *must* be present and speak at some point during the presentation.

The presentation on the group process should include (not limited to) use of Git, Trello, Quality Assurance and a reflection. The aim is not to sell your self as the best group, but

to show how you did the project, and what you learned from it.

5 Meetings

Weekly meetings with the client will start from **week 3** till **week 11** occurring during your allotted tutorial time. In the tutorial time slot, your group will have a 25 minute meeting with the course lecturers and tutors. The lectures/tutors primary role is portray the client. Clients are not usually tech-savay, so it is inappropriate to talk about technical details (e.g. classes and interfaces). The lectures/tutors may also act as supervisors to help guide you throughout the project when necessary.

While acting as clients the lecture/tutors are not able to answer highly technical questions. In practice the response you receive from the client can be nothing, delayed responses so they can check with their IT team or misleading — causing the client to conform to approving things that are misinterpreted.

Each client meeting is to have an **agenda** and **minutes**. This means you will need to plan ahead of time what your group wishes to cover and allocate a team member to take minutes. Each minutes must be presented with the following weeks agenda. *Tip:* Incorporate the previous minutes into the start of each client meeting.

If your group has some unanswered question on some technical aspect of the hardware we ask that you kindly post that question on *MyUni* or lookup the answer yourself on the internet. If the question is technical in nature but not generic (i.e. it relates to some design decisions you have made) please send an email to the lecturers and we will do our best to get back to you.

6 Tools

An intrinsic part of this course is learning about, and *effectively using*, new tools. Software Engineering workplaces often have their own selected tool chains they use and new engineers are required to use them and adopt the "house style". For the purposes of this course, we have chosen suitable tools that you may find familiar. The following sub-sections define tools that you will use in the group project.

6.1 Project Management

A free online software tool called Trello will be used to manage your SCRUM project. Trello is an online collaborative pin-board that will be used for tracking and allocating tasks in your SCRUM project. It is your responsibility to create and setup your Trello board. **You must invite the teaching staff to your Trello board.** *Please see MyUni for your teaching staffs email addresses.*

6.2 Source Code Control Systems

You are *required* to make use of a *source code control system*. Your group will be given access to a private Git repository via the School of Computer Science Enterprise GitHub portal. The repositories will be available when the groups are formed. In order for you to be added to your groups repository you **must** login to the School of Computer Science Enterprise GitHub portal with your student credentials. You should familiarise yourself with GitHub Enterprise and Git if you have not already done so. Please note, GitHub Enterprise is slightly different than GitHub.

For those who have not used Git before, Fir is a decentralised version control system where the act of checking out a repository creates a local copy on on your machine. Commits to the repository are performed on a local of the repository. The local changes are made available to the team via a "push". Here are few links that will be helpful to getting to know Git and GitHub:

- Git Essentials - https://code.tutsplus.com/courses/git-essentials
- Resources to learn Git - https://try.github.io/
- Become a Git Guru - https://www.atlassian.com/git/tutorials
- Glossary - Git Commands - https://www.atlassian.com/git/glossary
- Glossary - Git Terminology - https://www.atlassian.com/git/glossary/terminology

Using source control in a group may be new for some of you. Make sure to become familiar with fundamental concepts such as branching, merging, tagging, and resolving conflicts as well as design of your repository structure. You are required to be able to demonstrate your skills of the version control software via a terminal. However, you may wish to use alternate options during development.

You are required to have a version of all your documentation (requirements specification, designs, etc), together with all source code, for each working version of your system. You are also required to be responsible for the maintenance of this source code control system. **Accidental deletion of your files and other disasters is *your* problem** – consider the necessary precautions to prevent these cases from occurring.

We will, from time to time, ask you to produce and demonstrate past versions of your system. You should be able to check out an old working version out of the repository and build it on demand. Make absolutely certain that you can. *Each member of your group should expect to be asked to demonstrate this skill during the course of the semester in one of the meetings.* We will also be monitoring commits to your repository. Failure to use the repository properly will drastically affect your grade. Forking your groups git repository is strictly prohibited. You must only use the repository for managing the source code — with the exception of Subversion for submitting to web submission. Learning how source code control systems are used in the software engineering process is a learning outcome for this course.

Further to this, we will be checking that each team member makes use of the repository. For instance, if all your team members are making 10 commits to the repository per week, and we see that you have made none, we will deem you to have not made a contribution – which will be (unpleasantly) reflected in your marks.

6.3 Documentation

You may use any word processor for your group documentation. We highly recommend online collaborative tools such as Overleaf or Google Doc. Whatever you choose, be consistent, and use them with intent and structure. *This should be covered in your SPMP.*

You should also consider documentation generation tools for your source code. If you are using Java, Doxygen is a popular choice.

6.4 Configuration Management

The Source Code Control System should also include files that can build the software. A popular tool is the standard Unix "make". *You must provide a build script for each version of the system.* Remember, actually using the tools is one the of the goals of the this course.

Another typical build system for Java projects is "Ant" which you should familiarise yourself with during the course.

6.5 Editors

There should be no need to mention editors! You *need* to use a proper industrial strength text editor for this project. It is not a *requirement* in the sense that we will be asking you

to demonstrate your mastery of the tool, but it is really important.

If you have got this far by using "notepad" or "gedit" or any of the other primitive tools - stop now! Take a small amount of time to learn a more advanced tool. There are many choices such as, Sublime Text, Atom, Visual Studio Code, etc. Of course you may use what is most appropriate for you. We have found groups who use the same environment have less issues during development. Find a good editor and use it effectively! If no–one in the group has much experience with using normal text editors then as a group you should evaluate some.

6.6 Integrated Development Environments

There is *no* requirement for you to use an Integrated Development Environments (IDE). Your group might elect to use one, but you need to be aware that the effective use of an IDE requires a significant time investment. Using an IDE is like using an advanced editor but with much more language support. The IDE provided in the Computer Science labs is *Eclipse*. Another alternative is IntelliJ for Java projects. Again, make sure you do your research before committing to anything.

Each group should at least examine Eclipse and evaluate it as a productivity tool. Remember that the purpose of the project is to get you exposed to some tools and proficient in others. If you go to a job interview and are asked about IDEs – it might be embarrassing if you know nothing about them!

7 Testing

Testing is a critically important aspect to any engineering process. We *refuse* to be part of your testing process - *you* are expected to have adequately tested anything you present to us beforehand. Coming to a demonstration with code that doesn't compile, behaves strangely, or exhibits serious problems, is *unacceptable*. You will learn industry practices for ensuring your do not ship/deliver code that dosn't work.

Testing also applies to other non-code related parts of the process. If you are asked to give a five minute presentation and you take ten minutes, we will ask the obvious question: *did you test this presentation beforehand?* Whenever we ask you to present something, you must assume we are operating as *clients* and not your lectures/tutors! Getting a mark of zero for a presentation will cause you considerable damage with respect to your grade. Our expectation is that whatever is presented *has been tested/rehearsed already.*

As a group you should also remember that we have considerable experience with pro-

gramming, software engineering and this project in general. It is very disappointing when we are demonstrated a system and can see a way to break it and the group hasn't anticipated the defect. If you find a defect and can't fix it before the demonstration – make sure you have a convincing explanation, at the very least make it known to the audience and have a strategy in place to fix it.

Tip: Consider filming the demonstration before the presentation if you believe their will be a problem with the live demonstration.

You are *required* to use **JUnit** (or equivalent) to test your code. Your group might have a designated testing manager, but *each group member* must contribute at least one test case.

You are also required to make use of a code coverage tool such as (not limited to) **EMMA** or **COBERTURA**, to ensure that your test cases thoroughly exercise all your code. Consider using Mocks in your testing to improve how much of your code you can test. A good software design is one that is testable. Mocks may be one approach to make your code more testable.

8 Group Issues

Each year, at least one group runs into trouble with one or more members: not doing what is asked, "disappearing"', etc. We ask that if you have problems within your group you first try to resolve the problems internally. Interviewers for software engineering positions often ask questions relating to group work, difficulties in your group and how you over came them. If an internal intervention does not work, you need to contact the course lectures/coordinator as soon as possible. This is a fast paced course, so we need to take corrective action quickly.

If you decide to withdraw from this course *let your group members know.* You should also let the course coordinator know too. This is common courtesy - your decision will affect your fellow students and save everyone headache.

We would like the class to be stable after the first week - so **if you do intend to withdraw, please do so early!**