Software Engineering and Project
Group Project Specification

David Milanese
david.milanese@adelaide.edu.au

Amali Weerasinghe
amali.weerasinghe@adelaide.edu.au

July 2017
Version 1.1

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 for this course. In this document you will find the procedures and assessable components.

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 success with LeJOS. Should you wish to use alternatives - do so are your own risk, and make sure you do your research before commiting.

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 release 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 can't 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).

2.1 Progress

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 weekly progress data. This is in-line with common industry practice – managers need to know hours worked on projects.

Again in accordance with industry practice, and following from above, we will expect to see (and be able to assess) documents as used by your group in accounting for time spent. We expect that your group will adopt a practice of using *timesheets*. We expect that all group members will submit timesheets weekly, and that your group will allocate resources for processing these timesheets, and making sure they are recorded in appropriate project documentation. A template timesheet that contains a weekly break down of tasks by the hour will be provided that you may wish to use.

At the end of the course (with the final submission of documents), each group is expected to submit a statement (agreed to and signed by all members) setting out the relative contribution of each group member to the overall effort of the project. Ideally, these will be equal. We may verify contributions by checking timesheets and repository logs.

At the end of the project the group will be required to perform a peer assessment on each of their peers to provide feedback to the lecturers your reflection on your peers contributions to the project. This peer assessment will be factoring into everyone individual grade for the project.

3 The Software Process

From the beginning, your group needs to decide upon what software process will be used. Almost certainly, you *won't* be using any of the suggested processes in the exact form outlined in the text books. Rather, they can be used as a starting point to develop an appropriate process which suits your group and this project. In any case, you will need to understand the process your group is following at any given time, and be able to fully justify your choices. You might change the process as the project progresses; again any changes will need to be fully justified.

This course *requires* the production of documents; Software Requirements Specification (SRS), Software Process Management Plan (SPMP) and Software Design Document (SDD). Your process should support this. The documents are a large component of your groups assessment and should not be skipped.

You will discover that you need to get familiar with the tools quite early on in the process and you will also need to discover the capabilities of the robot in terms of control-

lability, accuracy etc. Be prepared to give a solution or idea based on what your group understands of the robots abilities during requirements elicitation and design. You should not underestimate the difficulty of writing code for the robot. Start early, develop and test in small increments and take full advantage of the source code control system.

4 Meetings

Scheduled weekly meetings starting from week 3 till week 12 will occur on the day of your allocated tutorial. When the groups are formed, your allocated meeting time may not be within your selected tutorial time. If this is a concern, please notify the course coordinator. In the allocated client meeting slot, your group will have short meetings with the course lecturers and tutors. Our role is portray the client. We *may* change our role to managers to help guide you throughout the project when necessary. These meetings are to be *fully* minuted by a member in your group and submitted at the following client meeting.

You are encouraged to ask questions in the meeting as well as seek guidance, — depending on the nature of the meeting. Your interactions, or lack there of are assessable in this course. The minutes will be part of the documentation you are required to produce, and the requirements pertaining to documentation apply to the minutes as well.

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 (so they can check with their IT team) or misleading - causing the client to conform to approving things that are misinterpreted.

If your group has some un-answered question on some technical aspect of the hardware we ask that you kindly post that question on the course forum 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 all the lecturers/tutors and we will do our best to get back to you. If you have finished your meeting with the client, you may wish to discuss non-client material with the tutor. Due to the limited timing, this is rare.

Each of these meetings should have an agenda; the group is responsible for setting and submitting this agenda. For some meetings, there will be agenda items that must be present (for example, prototype demonstration). Those will be revealed throughout the course.

5 Tools

An intrinsic part of this course is learning about, *and effectively using*, new tools. Most of the tools documented here are actually requirements. You cannot choose to use another tool or ignore the required function. 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".

Wherever possible, we have chosen tools that you will find familiar. The following sub-sections define tools that you are required to use.

5.1 Source Code Control Systems

You are required to make use of a *source code control system*. This requirement is non negotiable. Your group will be given a access to a private Git repository via the School of Computer Sciences 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 and Git (here) if you have not already done so. .

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

- Got 15 minutes and want to learn Git?
- Become a git guru
- Glossary - Git Commands
- Glossary - Git Terminology

There are no shortages of videos, tutorials and articles online that you may use.

You are required to have a version of all your documentation (requirements specification, designs, manuals etc), together along side your 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. 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.

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.

5.2 Documentation

This project requires you to produce professional quality documentation. The tool we have chosen for this purpose isLaTeX. All your documentation (including meeting minutes) must be in this format, and must be stored in the source repository. There are a number of tutorials online on LaTeX .

You will discover that LaTeX is a very simple tool to use, and permits you to implement professional documentation rather simply. LaTeX is fairly typical of the kind of documentation tools widely used in the industry. Using LaTeX actually makes your life easier, but requires a very small investment in learning how to use it so, again, start early.

This document is produced using LaTeX and the source is on the web page, so you can see how it looks. You may use various LaTeX clients or online IDE as long as there is a PDF feature.

Do not even *think* about using MS-Word or similar tools for the documentation and then using the "Microsoft Word to LaTeX" converter. LaTeX files are edited using a conventional text editor and you don't need to worry about formatting etc. It is actually *much* easier to generate quality documentation using LaTeX!

You will also need to produce proper documentation for your code and your group should devote a small amount of effort upfront choosing the appropriate documentation tool(s) to use. Java has JavaDoc which provides a satisfactory level of documentation

capabilities however there are other more powerful tools available, such as Doxygen, which you might consider.

You may wish to consider other types of documentation that might be useful to you and your group such as code conventions, environment and setup, etc.

5.3 Configuration Management

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

Your build script should contain a target (or equivalent) for all releases presented to the client. You may be asked to demonstrate your ability to build, so make sure it works and the team knows how to use it.

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

5.4 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, Eclipse, Atom, IntelliJ, etc. Of-course you may use what is most appropriate for you. Using the right tool can greatly reduce your development time and is well worth the investment.

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.

5.5 Integrated Development Environments

There is no requirement for you to use an 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*.

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!

Lastly, a *lightweight* IDE you might consider is something like JDE – which is an extension to the *emacs* text editor. Many students find editors like emacs confusing or difficult to get used to (basically because all features are available from the keyboard, rather than endless levels of menus and mouse-clicks). So this is not necessarily a suggestion for everyone.

In this project, the major benefit using an IDE would be the discipline the IDE enforces on you. The major drawbacks would be the learning curve and the discipline the IDE enforces. Choose wisely and do your research.

6 Milestones

You will be expected to set a series of milestones that must be accomplished by specific dates. The milestones will be delivered during the client meetings on week 8 and 9, respectively. Note: the mid-semester break is between weeks 8 and 9.

We will supply a form that enables you to specify milestones. You can think of these forms as check-lists - if you ensure your code covers all the listed things adequately, you can be assured you will meet the milestones.

Your group will be responsible for providing us with a completed milestone form for your group set milestones at least 1 week in advance of their presentation. We would expect that you would hand them over at the scheduled meeting the week before.

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* and embarrassing for you.

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 programming, 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.

7.1 Testing Tools

You are *required* to use a tool such as 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 COBERATURA, to ensure that your test cases thoroughly exercise all your code.

7.2 Testing and the Repository

We will be asking to see your tests, from time to time. This means you need to be able to get your testing infrastructure from the repository and demonstrate the testing. The testing infrastructure must be in the repository! - Be prepared.

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 us 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 us know. 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!**

9 Assessment

The project is assessed by us as you go along. There are numerous assessment artefacts you will be producing along the way that count towards your grade in the project. Don't forget the client meetings and any presentations you do individually or as a group will factor into your respective marks. Unusually, the "correct" operation of the end product you produce, is only a minor component of the assessment – this is a Software Engineering course and we are interested in the *process* more than the *product*. The actual details of the percentages allocated toward each assessment component are documented in the course profile.

The components that are assessed are given a mark which applies to your group as a whole. You are also given some individually assessed tasks along the way. Furthermore, we moderate the group marks as we go through the process of "distributing" those marks among the group members: this is done according to the contribution each member has made. The nett result of this is that a group that attains a bare pass grade for the project *could* end up with two members getting distinctions and four members failing; if it was the case that only two members contributed to the effort. You have been warned! That being said, if you are struggling with the course, don't be afraid to seek help.