Quality Assurance Manual

MEng Year 3
Department Of Electronic Engineering
University of York

Software Engineering Group Project

WeTech

Document Control

Version	Modified by	Date	Section modified	Remarks
1.0	HO	07/11/2018	All	Adapted from “Sample Quality Manual”
1.1	HO	14/11/2018	2.3	Completed role description draft.
1.2	HO	29/12/2018	2.2, 2.4, 2.5, 2.6	Imported role drafts
1.3	HO	12/01/2019	4.1, 4.2, 4.3, 4.4	Added project management draft notes.
1.4	IW	14/01/2019	Appendix A and header.	Added testing report template. Edited header to correct company name.
1.5	HO	18/01/2019	1, 2.1, 4	Formatted/Developed into paragraphs
1.6	HO	20/01/2019	All	Formatted into paragraphs, added to roles
1.7	HO	21/01/2019	4.5, Appendix A	Added QA review description and report template
1.8	IW	21/01/2019	3	Added table with deliverables.
1.9	IW	27/01/2019	4.3	Added info and diagram about TDD and added reference section.
2.0	IW	28/01/2019	3	Added another test reports to deliverables table.

Contents

1. INTRODUCTION

1.1 Company Profile

Based in York, we are an agile, inclusive and dynamic company working towards the creation of accessible and applicable experiences for our customers. Our main focus is on the design and development of innovative software solutions. We strive to create unique, personable applications tailored towards the user, that tend to an available market and benefit currently unresolved problems. We ensure all of our outputs are of high quality and compliant with current industry standards. Our products aim to make the lives of our customers easier, whether that be through wide-use applications or boutique solutions to smaller, more specific, cases. We design our applications with the potential for expansion and the possibility of user support via future updates. In each of our projects we put an emphasis on user efficiency and user satisfaction.

1.2 Company Vision

- To integrate our products into the forefront of their field.
- To provide a satisfying, uniform experience with opportunities for further development based on customer feedback.
- To ensure our products are of high quality, easy to use and affordable.
- To contribute effective solutions to current technology and user issues.

2. ROLES AND RESPONSIBILITIES

2.1 Organizational Structure

In order for a team of people to work together efficiently a well defined structure must be realised. This means individual roles should be devised and distributed, early in the process, based on preference, skills and experience to ensure that all members have a clear sense of their responsibilities and personal objectives. The Project Manager holds the task of upholding cooperation between team members as well as ensuring each member is adhering to their corresponding responsibilities. Communication is vital throughout the project in order to plan, design and implement effectively. This must be conducted regularly through weekly team meetings, lead by the Project Manager, in which all members are required to be present. An online company group chat should be created to ensure that contact between members is available “out of hours”. Sub-meetings, between select members of the team, should also take place when necessary, to discuss areas of the project in greater detail, the results of which are to be shared with the rest of the team. For tracking progress, as well as financial budgeting, a time sheet will be implemented to monitor each members working hours.

Roles required to be distributed between team members include:

- Project Manager (& Deputy Project Manager)
- QA & Documentation Manager
- Software Development Manager
- Creative Manager
- Testing & Integration Manager
- Finance Manager
- Risk Manager

All roles hold a specific set of objectives and responsibilities corresponding to the areas of the project in which they are active. Although each these roles will be more involved in some aspects of the project more than others, the functions of each role should be considered throughout all stages of the process.

2.2. Project Manager

2.2.1. Role Description

The Project Manager's primary responsibility is to ensure that the project is run smoothly and that sprint and overall goals are achieved within the agreed time. The Project Manager will serve as a link between the company and the customer and make sure to maintain a good relationship and communication. They will also lead weekly group meetings to discuss current project progress and required actions.

Tasks for the Project Manager include but are not limited to:

- Track progress against tasks.
- Ensure that team members complete their tasks before deadlines.
- Ensure that appropriate resources are available when and where required.
- Balance workload based on team member resource availability.
- Manage project timeline and maintain relevant documentation.
- Keeping client aware of the status of the project.
- Manage client expectations.
- Maintain an issue log and be responsible for resolving and managing issues.
- Communicate issues to client where applicable.

2.2.2. Risk Management

Conflict between group members - Understand the positions of each member involved and reach a working agreement.
Mis-communication with client - Communicate frequently to ensure that the product aligns with the clients vision and notify of any changes made.
Objectives not completed by deadline - Monitor the progress of the project and increase resources when and where necessary.

2.2.3. QA Metrics

Project Manager QA will be measured by team and customer feedback requested by means of a survey. The survey will be sent regularly to all stakeholders.

2.2.4. Deputy Project Manager

The Deputy Project Manager will take on all the responsibilities of the Project Manager in case of their absence. Due its nature, the deputy leader role can be undertaken as a side-role by another member.

2.3. QA & Documentation Manager

2.3.1. Role Description

Quality assurance ensures that products produced by the company, and the documentation that accompanies, is all of the same high standard and meets customer requirements. This role is responsible for creating and carrying out frequent processes/checks to maintain company standards and quality. This includes consulting with other members to decide the company's desired coding and formatting preferences, regularly checking up on updated work and ensuring the standards are met, managing and planning the creation of all required document deliverables.

Tasks for the QA & Documentation manager include but are not limited to:

- Ensure that the product meets the customer requirements/specifications at each development stage of the project.
- Responsible for setting standards in coding, formatting and layout that can be reproduced across all work throughout the project.
- Setup and monitor a change management system to ensure that the most up-to-date documents are always used when new changes are made.
- In conjunction with the test & integration manager, run review processes to ensure quality standards at the end of each development stage.
- Keep track of document/report deliverables and devise a plan of completion for each.
- Ensure that all documents are formatted as to the standards agreed by the company.
- Ensure records are taken of each meeting (e.g. meeting minutes).

2.3.2. Risk Management

Missing Documents - Ensure backups of newest document versions are always made.
Mixed Document Versions - Ensure newest versions of documents are always used when making changes and document change log is updated after new each new addition.

2.3.3. QA Metrics

Document Formatting - How many documents have been successfully formatted to defined standards in a given time frame versus how many were expected to be completed.
Document Deadlines Met - Submission of document deliverables vs timetabled deadline.
Product QA - Number of features that meet the customers requirements/specifications vs expected.
QA Metrics - How many QA metrics have been completed vs expected.

2.4. Software Development Manager

2.4.1. Role Description

The Software Development Manager is responsible for planning, organising and overseeing all aspects of the software development on any given project. They should have a clear understanding of the requirements and specification of the project to make effective, high-level design choices and assign relevant sprint tasks to each team member. They must regularly check software module submissions to ensure that the code meets the company-wide coding standards and is compatible with all other software modules. This means working closely with the Integration and Testing Manager and QA & Documentation Manager to develop or update the company-wide coding standards. They are also responsible for ensuring that all software modules have changelogs and that these changelogs are kept up to date.

Tasks for the Software Development Manager include but are not limited to:

- Developing an effective, modular, high-level software design based on the specifications and customer requirement.
- Determining a set of company-wide coding standards in collaboration with the QA & Documentation Manager and the Integration & Testing Manager.
- Determining project-specific standards such as the programming language(s), interfaces and software architecture.
- Ensuring regular software-development sprint tasks are successfully completed by each member of the team.
- Checking submitted software modules to guarantee both adherence to company-wide coding standards and cross-module compatibility.
- Ensuring all modules' changelog documents are up kept up to date and maintaining a master changelog document.
- Organising regular peer code reviews with current team members.

2.4.2. Risk Management

Failure of code - Work closely with Testing & Integration Manager to determine a full set of test cases and ensure modules work in all circumstances, as well as to confirm cross module compatibility. Once in the implementation phase, organise bi-weekly code reviews to help continuously improve each team member's programming skills.
Code inconsistencies - Determine a set of clear, company-wide coding standards at the start of a project for everyone to follow. Check each software module in a given project when it is submitted to ensure it adheres to these standards.
Module changelogs not updated - Use Github's change history to fill in these gaps afterwards. Speak with team members who have not been updating the changelog to ensure that they do in future.
Loss of project files - Once in the implementation phase, take weekly backups of any project files from the given projects' Github repository.

2.4.3. QA Metrics

Development progression - The number of modules successfully completed in a given time frame versus how many were expected to be completed.
Total time spent coding - The amount of time each team member spent coding over a given period versus how much time they were expected to spend.

MTTR (Mean Time To Repair) - The mean of the time taken from an error being found in a module to this error being repaired.

2.5. Creative Manager

2.5.1. Role Description

To oversee the creative design process of the final product. Utility and functionality are often overlooked if the product does not function or appeal to the customer in a way which makes it intuitive and usable. Prioritising values such as convenience and ease of use in the final design is one of the responsibilities of this role which leads into the main responsibility of designing and finalising the GUI. Other responsibilities involve working with the marketing and finance managers in order to create a brand and marketing plan to best optimise the marketability of the final product.

Tasks for the Creative Manager include but are not limited to:

- Design and oversee the GUI of the final product.
- Engage discussion with the team and the client regarding the usability of the design of the product.
- Work with the finance manager in order to create a viable marketing plan for the final product.
- Oversee the design of the final presentation.
- Ensuring that everyone on the team believes that the design choices made accurately reflect the values of the company and represent the idea being made in the best possible way.

2.5.2. Risk Management

GUI detracts from core functionality - Work closely with both the team and client to ensure that the final GUI does not inhibit or hinder core features of the product with a poorly designed and unintuitive GUI.
Marketing is unrepresentative of the final product - Work closely with both the team and the client in order to ensure that the way in which the product is marketed is appropriate and representative of the team and client's vision for the product
Delay in design completion - Report directly to the team on the status of the GUI and compare it to the progress of the backend programming in order to review why the delay has occurred and to see if parts of the design flow needs to be re-prioritised.

2.5.3. QA Metrics

Design Appropriateness - Correct and intuitive translation of specification and features.
Module/Component Interoperability - Estimate the cohesion among different components of the design.
Defects - Number of defects identified and how many parts of the front-end design does not match up with the back-end.
Deliverables Submitted/Reviewed - Number and percentage of design elements that are released compared to estimate of how much would be designed for the release.
User/Customer satisfaction - User/Customer feedback.

2.6. Testing & Integration Manager

2.6.1. Role Description

The Testing and Integration manager is responsible for managing the procedures and documentation involved with the testing of the product and managing the integration of modules both internally and externally developed into the product. The primary role for the testing and integration manager is to develop a plan for the overall testing an integration of the product to ensure that the product meets the high quality that customers deserve.

The Testing and Integration manager in conjunction with the Software manager and Project Manager is also responsible for assigning tasks related to testing to members of the group, these tasks will be documented using the standard test report shown in Appendix A. The Testing and Integration Manager is also responsible for collecting and analysing the completed test report and discussing these reports with other group members.

Procurement of externally developed modules is also partially handled by this role along with the software manager, finance manager and Project Manager. The test and integration manager is responsible for reviewing and researching external software modules and API's to ensure that the company procures the optimum software for use in the product.

Tasks for the Testing & Integration Manager include but are not limited to:

- Manage the overall testing of the product.
- Manage the integration of different modules used in the product.
- Assign tests to to team members accounting for their current workload.
- Research APIs and external modules to determine if they are suitable for use with the software being developed.
- Collect and discuss test reports with members of the team.
- Produce an overall test and integration plan for the product.
- Assist with the purchasing of external modules with the software manager and Project Manager.

2.6.2. Risk Management

Group member does not complete test report sheet by deadline - Discuss the reason for overrun with group member and attempt to find resolution.
Test report sheet lost - Keep backups of test report sheets and inform documentation manager of loss.
Implementation phase of project overdue - Inform software manager. Reschedule integration and testing plan. Find reason why this phase is overdue.
Testing reveals error which is hard to resolve - Inform software manager of issue and attempt to find resolution to the issue.
Developed user story is incompatible with rest of product - Inform software manager of issue and attempt to find reason why user story is incompatible.

2.6.3. QA Metrics

Integration and test plan followed - Monitor testing and integration process. throughout the project and compare against final test and integration plan.
Number of user stories tested - Collected from test reports.
Number of fixed errors and time to fix them - Collected from test reports.
Number of unresolved errors - Collected from test reports.

Number of successfully integrated user stories - Collected from test reports.

2.7. Finance Manager

2.7.1. Role Description

The Finance Manager is responsible for managing the documentation, and upkeep of the projects finances. The primary role of the finance manager is to develop a financial plan for the project in order to secure funding and to write up financial reports to keep track of budgets and to maintain a healthy financial situation for the group. The Finance Manager in conjunction with the Project Manager, will collect and approve of the timesheets and allocation of pay.

Tasks for the Finance Manager include but are not limited to:

- Create a financial plan document to ensure the project's budget is feasible.
- Monitor the projects budget through financial reports.
- Monitor members working hours through timesheets.
- Advise Project Manager on improving financial efficiency.
- Report financial status to the rest of the group.

2.7.2. Risk Management

Group member does not hand in timesheets by deadline - Discuss with the member about the reason for and solutions to the overrun, the finance manager should remind everyone of the importance of the timesheets to try to avoid this situation.
Documentation Lost - Keep backups of all documentation regarding finances, using not only computer hard-drives but also using cloud technology in order to reduce the risk of losing important documentation as much as possible.
Overrunning budget - Discuss with Project Manager and the risk manager about the overrunning budget, re-evaluate budget and work to secure more funding if necessary.

2.7.3. QA Metrics

Financial plan followed - Monitor finances throughout the project and compare against financial plan.
Funding secured - Determine, via the implementation of a financial plan, the required funding for the project and endeavor to secure that funding.
Budget Maintained - Every week check to make sure that all the financials are within expected parameters.

2.8. Risk Manager

2.8.1. Role Description

Analyse each stage across all areas of the project to provide foreseeable and reasonable types of risk that are relatable to each task in question, that could delay or stop the project going to market. Draw from risks outlined in each task to provide a general level of risk and how much of a delay the task could take up, if one of the risks outlined occurs. Additionally, view and predict the consequences from the tasks in question not being completed or delayed. Also, to adjust the level risk on tasks based on progress levels through the week.

Tasks for the Risk Manager include but are not limited to:

- Updating team leader on current predictions of level of risk in current tasks being undertaken, to help advise where time is needing to be spent.
- Produce a document about risk predictions on product based upon market, development time and skills needed.
- Produce and update a risk register on task risk level.
- Produce documents on all tasks and their corresponding risks and consequences of missing deadlines or in-completion of task.

2.8.2. Risk Management

Missing risk assessment of product – spend reasonable time researching product and call emergency meeting to speak to team members about their contributions to the product, to get an understand of where risk will arise in the product development in order to give a good final product risk assessment.
Un-updated risk register – attend all meetings and talk to team members regularly about current task being undertaken and cross reference them against deadline and general task risk in order to update risk register accordingly as quick as possible.
Missing task's risk assessment – understand the task that is be undertaken and give a quick brief about risks on the task. Highlight relevant risks to team member that the task is given to.

2.8.3. QA Metrics

Risk assessment of product – accurate predictions on time needed to spent in the selected area of the project based upon highlighted risks that could occur during product development. Also the effectiveness of outlined steps to be taken if one or more highlighted risks arise.
Risk register – regular up to date record on the current risk levels of ongoing tasks.
Task's risk reports – accurate predictions on time needed to spent on the given tasks and the given deadline for the task based upon highlighted risks, assigned team member(s) skills and assigned team member(s) other deadlines.

3. DELIVERABLES

Table of deliverables required throughout the project

Deliverable	Producer	Recipient	Due
QA Manual	QA and Documentation Manager + Managers	All company employees	On company establishment
Functional Specification	QA and Documentation Manager + Managers	Client	End of Specification Phase
Tender Presentation	Project Manager + Mangers	Client	Beginning of project
Financial Business Plan	Finance Manager	Financial backer	Beginning of project
Financial Reports	Finance Manager	Project Manager + Financial backer	Throughout project
Financial Summary Report	Finance Manager	Project Manager + Financial backer	Conclusion of Project
Testing & Integration Plan	Testing and Integration manager	Testing team	Before Integration of modules
Meeting Minutes	Minute taker	Project Manager	After meeting
Test Reports	Programming Team	Testing and Integration Manager	Throughout project
Risk Reports	Risk Manager	Project Manager + Software Manager	Throughout project

4. PROJECT MANAGEMENT METHODOLOGY

In order to be successful in a project, a clear process must be defined, in which the path to meet the objectives is split into achievable procedures. Our methodology defines how the company approaches new projects, the exercises which are performed and the practices that are followed throughout each stage of the process.

General Methodology:

- Client defines required software product/solution
- Requirements and design of product decided
- Product implemented into executable code
- Product tested and reviewed.

This is divided into five main phases, described below.

4.1. Requirements and Specifications Phase

We begin our projects on the basis of a software product/solution specified to us by the client/customer. From this, the product requirements such as the function, application and features of the product can be discussed and understood. The requirements should also be checked and confirmed by the customer to ensure the product is accurate to the customers vision or fits the specified solution. A desired time span for the completion of the product is also deliberated and accepted. Throughout the project the Project Manager will communicate any changes to the product requirements to the customer.

4.2. Design Phase

During this phase members plan how the requirements can be implemented to produce the product. To assist with this a functional specification document is produced. The functional specification should be referred to throughout the implementation stage and acts as an outline functional plan for the product including definite features, resource requirements and the expected completion time. The specification should be edited, further into the development process, if a feature requires change or is no longer necessary. Components of the system and the interfaces between them are laid out without a detailed discussion on the internal implementation. The document should contain only features that aim to meet the requirements. Any unnecessary/inconsistent features should be deliberated with the customer and removed to ensure efficiency in the implementation stage. Features should ideally be ranked in order of priority to easily track which features have been completed and should be approached next. At this stage a financial plan/budget document is also created to ensure project costs are feasible.

4.3. Implementation Phase

At this stage the software team works on transforming the desired features from the functional specification into executable code. Software managers will be responsible for monitoring progress and ensuring written code is accurate to the specification during the implementation process. The software managers will also devise appropriate time scheduling for the completion of each feature, as well as allocating the required resources. Required programming tasks will be distributed between all team members. Paired programming sessions are adopted for completing coding tasks to increase efficiency and reduce the

amount of bugs/errors. When components are finished they are passed on to the testing & integration manager for the next phase. During this process the finance manager will also monitor the budget and individual working costs. Members will track their working hours on individual timesheets which can be viewed by the Project Manager and finance manager. A test driven development (TDD) approach will be used as it allows for rapid delivery of useful software and can adapt to changing customer requirements. A diagram showing the TDD approach is shown in Figure 1.

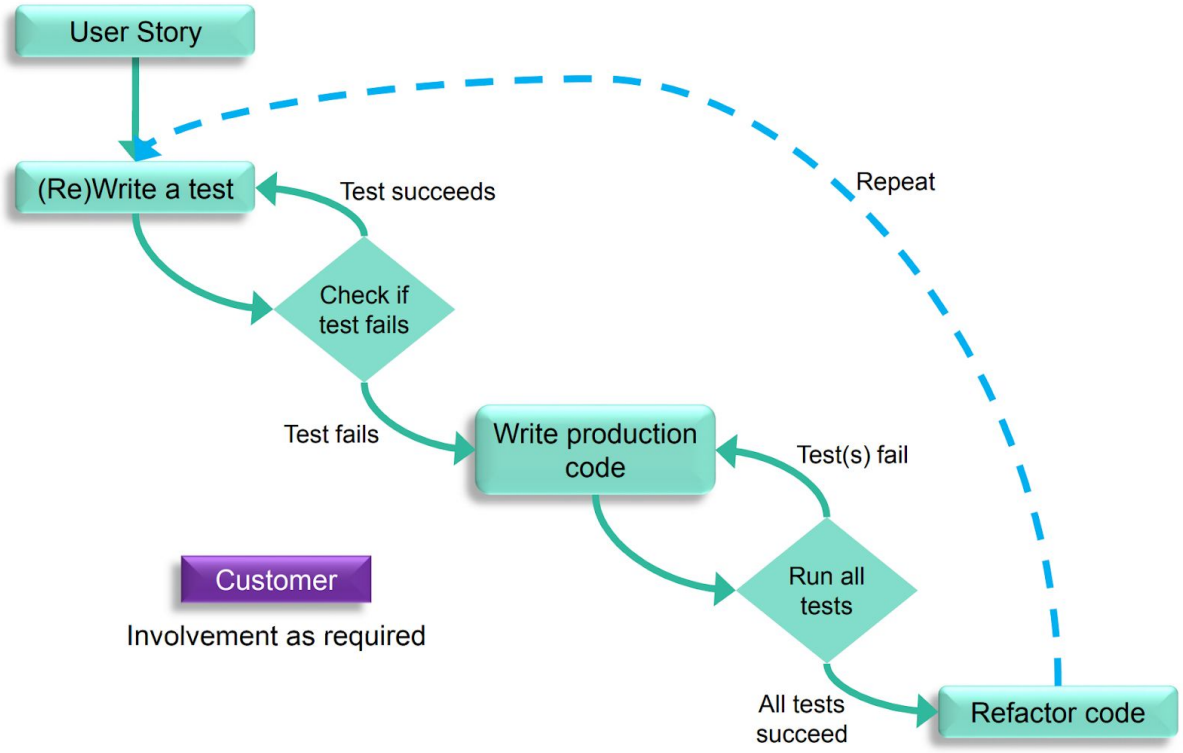

Figure 1: Diagram showing test driven development approach to software engineering. Diagram taken from [1].

4.4. Testing and Integration Phase

When an individual features (or product version) coding has been finalised, the testing and integration phase can begin. During this stage the functionality of features are tested before being confirmed as up to the specification requirements. The testing and integration manager is responsible for designing suitable procedures for testing each feature during this stage. The procedures should be conducted to ensure basic functionality. Software features must be tested against themselves and also in correspondence to other, currently completed, modules to ensure successful integration. Testing of a module/component should be carried out by team member other than its respective implementor. After testing a feature, whether successful or otherwise, a test report must be completed to note the test results, errors corrected and errors remaining. Test reports should be managed by the testing and integration manager. Multiple errors found during the testing phase may result in regression to an earlier stage of the tested feature.

4.5. Quality Auditing Reviews

In order to ensure the final product is of high quality, reviews must be conducted across each stage of the development process. These reviews aim to check that the current product version meets the defined standards of quality, along with reporting the collection of QA

metrics from each member. During these reviews a number of resources must be analysed to comprehensively assess the state of the product before reaching a conclusion, including testing and integration reports, QA metrics and risk reports. All members must be present during the QA review including, on occasion, a customer representative if the deliverable regards their requirements or is related to the final product version. A QA report must be produced to summarise the review and document the results. In order to progress to the next development stage, approval must be given by the QA & Documentation manager and the Project Manager. If the current product version does not meet the required standards the process will regress to the previous stage in order to correct any errors/problems.

APPENDIX A: DOCUMENT TEMPLATES:

General Formatting:

Document formatting must be standardized to give the same sense of quality across the company. Document formatting should be monitored by the QA & Documentation Manager throughout the project.

Defined company standards include:

- All pages should include the title of the document in the top right, in full or abbreviated form, as well as the current version of the document (e.g. QAM/1.0).
- All pages should include the company title in the top left.
- All pages should contain the current page number in the lower right corner.
- All text should be written using Arial font (unless in specific cases).
- Main document titles should be of bold size 14.
- Document section titles should be of bold size 12 and subsection titles of bold size 11.
- General text should be of size 10.

Configuration Change Management:

To ensure that the most up-to-date version of documents are always used when making updates a change management system is employed by the company. This allows a record of the changes made to each document to be kept and prevents any confusion between document versions.

Change Management System:

- All document files should be named with the full title of the document, followed by the current version (e.g. QA_Manual_1.0).
- All documents must include a documentation control table displaying the version of the document, a history of changes made to the document and the modifying members initials.
- When a document is updated it must be ensured both the document title and corresponding change-log is updated also.

Coding Standards:

Standardised coding style will help ensure code is kept readable and maintainable for anyone within the company, even if it is their first time looking at a pre-existing project. The style should also help minimise the number of comments required due to the self-documenting properties of clearly written class, variable and method names.

Programming Style:

- Variable names should be nouns, with single letter variables used only for loop counters.
- Method names should be verbs, or phrases indicating an action being carried out.
- Multiword variables should be in snake_case.
- Multiword methods should be in lowerCamelCase.
- Multiword classes should be in UpperCamelCase.
- Tabs should be used to indent code (not spaces).
- There should be spaces between binary operators.
- Curly braces should be on the same line as methods, classes, conditional logic and loops.
- Test classes should have the same name as their related class with “Test” added to the end.

Commenting & Documentation:

- There should be only one class per file (excluding nested classes).
- Each class must have a block comment at the top, documenting:
    - The class’s content and purpose
    - Date of creation
    - Original author
    - Date of last change
    - Author of last change
- Each method should have a block comment containing:
    - Description of what the method does
    - List of method parameters with descriptions
    - List of returns
- Complex or confusing sections of code should be commented on the line above to add clarity.
- Each file should have a changelog.
- Any external APIs or libraries used **must** be referenced in the code.

Keeping effective comments of classes and methods will help give an easy to understand overview of the class or method for anyone reviewing the file, as well as adding clarity to confusing sections.

Example Java File:

```
package main;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

/**
 * ExampleMainClass is the main class in the coding standards example project. This handles all
 * front-end user interaction with the application.
 *
 * Date created: 03/12/2018
 * Date last edited: 03/12/2018
 * Last edited by: Dan Jackson
 *
 * @author Dan Jackson
 */
public class ExampleMainClass {

	//GUI components
	private JFrame main_window;
	private JPanel button_panel;
	private JPanel label_panel;
	private JButton increase_button;
	private JButton decrease_button;
	private JLabel number_label;

	private int value;
	private int upper_bound;
	private int lower_bound;

	/**
	 * Constructor functions for the main example class. Initialises values, creates the main
	 * window and instantiates all the button listeners.
	 */
	public ExampleMainClass() {

		value = 0;
		upper_bound = 10;
		lower_bound = -10;

		createMainWindow(300, 100, "Example Project Window");
		instantiateButtonListeners();
	}

	/**
	 * Method to create the main window frame and add all relevant GUI components to this frame,
	 * as well as initialising the states of each component and the frame.
	 *
	 * @param with_length - length of the window in pixels
	 * @param with_height - height of the window in pixels
	 * @param with_title - the title of the window
	 */
	private void createMainWindow(int with_length, int with_height, String with_title) {

		//sets up the window
		main_window = new JFrame();
		main_window.setSize(with_length, with_height);
		main_window.setTitle(with_title);
		main_window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		//sets up panels contained in the window
		button_panel = new JPanel();
		button_panel.setLayout(new FlowLayout());
		label_panel = new JPanel();
		label_panel.setLayout(new FlowLayout());

		//sets up the buttons
		increase_button = new JButton();
```

```
		increase_button.setText("Increment");
		decrease_button = new JButton();
		decrease_button.setText("Decrement");
		//sets up the label
		number_label = new JLabel();
		number_label.setText(String.format("%d", value));

		//adds components to the panels
		button_panel.add(decrease_button);
		button_panel.add(increase_button);
		label_panel.add(number_label);

		//adds the panels to the window
		main_window.add(label_panel, BorderLayout.NORTH);
		main_window.add(button_panel, BorderLayout.SOUTH);

		main_window.setVisible(true);

	}


	/**
	 * Method to instantiate button listeners for each button on the GUI.
	 */
	private void instantiateButtonListeners() {

		increase_button.addActionListener(new ActionListener() {
			@Override
			/**
			 * When increase button is pressed, increment value by 1 if it is below
			 * the upper bound.
			 */
			public void actionPerformed(ActionEvent e) {

				if(value < upper_bound) {
					value++;
				}

				number_label.setText(String.format("%d", value));
			}
		});

		decrease_button.addActionListener(new ActionListener() {
			@Override
			/**
			 * When decrease button is pressed, decrement value by 1 if it is above
			 * the lower bound.
			 */
			public void actionPerformed(ActionEvent e) {

				if(value > lower_bound) {
					value--;
				}

				number_label.setText(String.format("%d", value));
			}
		});

	}

	/**
	 * Main program function.
	 * @param args
	 */
	public static void main(String[] args) {
		new ExampleMainClass();
	}

}
```

RACI Matrix Template (with Examples):

R = Responsible A = Accountable C = Consult I = Inform			(If we use a project management tool)	Group Leader	Deputy Leader	Issue And Risk Manager	Software Development Manager	Finance And Marketing Manager	QA And Documentation Manager	Integration And Testing Manager	Creative Director and Marketing Assistant
SMART Activity	**Frequency**	**Deadline**	**Activity ID**	**Aeri Olsson**	**Jack Small**	**Jack Small**	**Daniel Jackson**	**Luke Fisher**	**Harry Ogden**	**Isaac Watson**	**Ho Ting Lok**
Arrange team meetings	Weekly	Before Sunday 8pm		RA	R		I	I	I	I	I
Arrange peer code reviews	Fortnightly	At team meetings		I	I	I	RA	I	I	I	I

Master & Individual Timesheet Template (with Examples):

Master Time Sheet

Week	Name	Date	Tasks Completed	Time Started	Time Finished	No. of Hours
1	Aeri Olsson	17/01/2019	Meeting	13:00	14:00	01:00
2	Harry Ogden	20/01/2019	Developed Functional Specification	15:00	17:00	02:00

Harry Ogden - QA Manual & Documentation Manager

Week	Name	Date	Tasks Completed	Time Started	Time Finished	No. of Hours
2	Harry Ogden	20/01/2019	Developed Functional Specification	15:00	17:00	02:00

Total Hours
02:00

Master & Individual Team QA Metrics Template (with Examples):

Master QA Metrics

Week	Name	Date	QA Metric	How Measured	Results vs Expected	Remarks/Comments
Example	Isaac Watson	23/01/2019	Tested Features	Testing & Integration Reports	2 Features tested successfully, 1 error	
Example	Aeri Olsson	23/01/2019	Team Satisfaction	Google Form	Postive feedback	
Example	Harry Ogden	23/01/2019	Documents Formatted	Analysis of company documents	7 out of 9	Format remaining documents

Harry Ogden - QA Metrics

Week	Date	QA Metric	How Measured	Results vs Expected	Remarks/Comments
Example	23/01/2019	Documents Formatted	Analysis of company documents	7 out of 9	Format remaining documents

Meeting Minutes

Meeting Number:
Date:
Time:
Length:

Minutes taken by:
Group members absent:

Agenda:

1. Actions from previous meeting:
    -
2. Items of discussion:
    -
3. Actions to be completed before the next meeting:
    -
4. Next meeting date and time:
    -

Test Report

Author(s):

Date:

Project:

Report Title:

Circulation List:

Module Under Test:

-

Tests Performed:

-

Test Results:

-

Errors Corrected:

-

Errors Remaining:

-

Comments:

-

Test Programme:

Task Risk Assessment Report:

Task name : ……

Team member(s) name : …… **start date :** …….

Task description

……
……
……
……

Highlighted risks:

- .
- .
- .
- .

Estimations:

Minimum Time spent on task : …… hours **End date :** ……

Reasoning of estimations :

……
……
……
……

Updates:

Date	Notes	Changes to estimations	
		Hours	End date

Task report:

Task duration:

Time spent on task : …… hours **End date :** …….

Quality Auditing Report

Author(s):

Date:

Project:

Report Number:

Report Title:

Circulation List:

Description of deliverable for review

Analysis of deliverable state vs specification

QA Decision:

Evaluation of review conclusion

Comments/discussion points/recommendations

References

[1] S.Porter "*Software Engineering Lecture 2"* [Online] Available: https://www.elec.york.ac.uk/internal_web/meng/yr3/modules/SWEng_Project/Lecture2.pdf