**Quality Standards:**

***Identify Quality Issues:***

Potential quality issues include:

* Inconsistent or messy code
* Code does not conform to a standard
* No version control workflow / procedure is in place
* The software is error prone
* The software crashes frequently
* “ Coupling and Cohesion “- Roland

***Describe impact where no measures taken:***

If no measures are taken to prevent the about quality issues, the following may occur:

* The product doesn’t meet the requirements of the project.
  + The client is not happy with the product.
  + Internal investigation finds that the product does not do what is required of it by the client.
* Code is hard to read / change / debug.
  + If code does not follow a consistent standard – then it will be very hard to fix problems that occur between code written by different programmers.
  + This could mean more time is spent debugging / fixing code than is allowed / necessary.
  + This also makes code hard for new team members to read and become familiar with as they might expect some standardisation.
* You do not reach a particular deadline.
  + If the code takes longer than expected to write / fix, it is likely the product; or a certain piece of functionality, will not be ready in time for the client.
  + This makes the overall quality of the product worse.
* There is poor, if any, version control and stability.
* Version control software allows teams to manage many different changes on the same project without much, if any, overhead.
* Without proper VCS procedures, it is possible for code to become cumbersome to combine and makes it difficult to separate tasks between many members.
* This slows the development process down and reduces code quality.

***Review quality standards for procedure:***

*Reviewing “GitFlow” workflow from Atlassian (*<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>*)*

The outline of GitFlow is simple:

* There are branches which have certain restrictions on how they are merged
  + A ‘develop’ branch which comes out of master and can only be merged into a “release” branch
  + A ‘release’ branch which comes from the develop branch and can only be merged back into the develop branch or into the “master” branch (which must be titled with a version number)

https://cdn-images-1.medium.com/max/1600/1\*9yJY7fyscWFUVRqnx0BM6A.png

* + “Feature” branches; which are created from the develop branch, which are designed to contain changes that add certain features to the project. These get merged into develop.
  + “Hotfix” branches will be created from the master branch if there is a problem that needs to be fixed. These hotfixes get merged into develop and master when they are completed.

This can be checked by a supervisor or lead developer by making sure no branch is merged outside of its restrictions, and that all features are being added in separate branches. This should be done after every commit is pushed and at the end of the day. If the commits are maintaining the structure of “GitFlow”, the benefits will be retained.

***Review quality standards for code:***

*Reviewing “C# Coding Conventions” by Microsoft*

*(*<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>*)*

The goal of a coding convention is to maintain consistency across all pieces of code, streamlining the processes of reading unfamiliar or old code, writing new code, and debugging erroneous or broken code.

Microsoft’s C# Coding conventions outline:

* Naming conventions
  + Qualifying methods or properties (doesn’t have to be done fully if a higher name space is imported)
* Layout Conventions:
  + Using the pre-set code editor settings (more on MSDN pages)
  + Writing concise, succinct, and clear statements. No more than 1 per line.
  + Indent continuations with a tab space if not automatically done
  + Add at least 1 blank space between properties and methods
  + Use brackets (parentheses) to make clauses more apparent and clear.
* Commenting Conventions:
  + Place comments on separate lines
  + Start with capital letter.
  + End with a period.
  + Insert a single space between the slashes of your comments (shown on MSDN)
  + Do not create blocks of asterisks
* Language Conventions:
  + Implying locally typed variables (using Var instead of the Data type if it is apparent from the right side what data type is being used)
  + Only concatenate short strings, otherwise use StringBuilder.
  + Use concise syntax over a number of language elements such as
    - Objects,
    - Arrays,
    - Delegates,
    - (More shown on MSDN)

To review how your code follows the MSDN C# Coding Convention – create a checklist and check how well your code follows this coding standard. The better you match the standard, the better quality your code is.

***State the benefits of these standards:***

*GitFlow:* GitFlow has major benefits for development that works on a release schedule. It allows minute control over how features are added and how releases are deployed and fixed (if necessary). It creates a neat-and-tidy structure for commits and enhances the development process through convenience and useability.

*C# Coding Conventions:* Microsoft’s C# coding convention outlines the best practices for C# coding while also streamlining the process of reading unfamiliar code, writing new code, editing old code, or fixing broken code. It outlines language, layout, naming and documentation conventions which all improve the consistency of the code.

***Discuss quality issues encountered:***

Code was unreliable:

* The first prototype of the application didn’t accurate encompass the outcome of the project nor did it satisfy the requirements of the client. The code did not function as expected and a lot of code had to be rewritten to remedy this. This could have been prevented if more quality control was taken into consideration earlier into prototyping.

The product wasn’t user-friendly:

* While the application worked perfectly fine for our testing team – the client was unhappy with the layout and asked for a re-design. We talked with the client and rapidly fixed this issue. The client was happy with how we handled the issue. This problem could be prevented if more in-depth discussion about the user interface and experience. This solution may require more resources to be distributed among the relevant teams.

***Determine quality standards for production:***

During development of the product, production must be above a certain level. This level is set by the following outcomes:

* Code is of high quality and is consistent.
  + Code should be written to avoid having to rewrite large percentages of the code.
  + Check code follows convention frequently.
* Version control follows the correct and outlined procedure.
* The product developed is reliable and is frequently checked against the specification.
* Estimations taken must be educated, discussed, and reserved. It is better to overestimate and be pleasantly surprised than to underestimate and be caught off guard.

***Determine quality standards for the final product:***

After development of the product, it should equal or surpass a given standard. This standard is set by the following outcomes:

* The client is happy with it. (Conduct user acceptance testing)
* It fits the client’s needs / the requirements of the project.
* It is a reliable product.
* There is a clear communication channel for future improvements and feedback has been discussed both internally and externally by the development team.