Quality Plan

Quality Plan for development project of ObsconZ

Stephen Jackson, Dawid Janelli, Will Hodgson and Theo Ferdinand

Contents Page

Introduction:

This development project is to provide a solution to the ObsconZ company, or more specifically, the client, Miles Roman. This quality plan and the sections that have been written are to ensure that the development process is of high quality so that the client is happy with the final product. The quality plan is also used to ensure that any unforeseen circumstances do not hinder the development cycle too much and that we can produce a solution which is useable by the client in the timescale given to us. The quality plan should be followed closely.

How, when and by whom reviews are to be carried out:

The reviews are an important part of testing not only for the solution/final product, but the development process are of high quality. Due to this, it is important to specify how, when and by whom the reviews will be carried out. The reviews will be carried out by every member of the team, to ensure that nothing is missed when it comes to the high quality of the work that needs to be produced for the client.

For the first phase of the development process, the analysis and requirements, the team will work together for the review, producing high quality UML diagrams and plans to ensure the rest of the project goes ahead with little deviation in the time frame. This is to allow extraneous variables(such as change in requirements) into the process without too much problem to the development. This will be carried out during the entirety of the phase during each iteration of the project. This will allow us to check the quality continuously while creating the analysis and requirements, allowing us to change it as we go along. We will review this phase by each reading through the requirements set out by the client and matching them with our own ideas of the system, seeing how many match and which ones we should change to ensure the client is happy with our analysis and our proposed idea for the system.

When reviewing the design, we will have the whole group coming up with the design together, using paper based versions to come up with the best design possible. Then, when checking the quality, we will have two people go test out the design on a few people, to get a general feeling over the design and the other two people will go and test the design with the client, to ensure he agrees with our quality of work, ensuring our quality is of the highest it can be. This will be carried out in the middle of the design phase, to allow us to get any changes from the client and be able to modify the design in time for the next phase, ensuring the next phase does not need to worry about the quality of the design and to ensure the client is overall happy with the design.

The coding phase review will be carried out during the coding, to ensure proper documentation and coding style and will also be carried out after to test for bugs and test the code/system is of high quality and matches with the clients requirements. This will be carried out by everyone in the team at certain stages. First the team member will test their own code, to look for any obvious bugs and problems. Then code will be swapped so the other team members can test the code, which can look for harder bugs. Once these small bugs have been fixed, the system will be tested as a whole with all of the team, to ensure each component works together and performs how it should.

Dealing With Problems

Dealing with problems is a very important part of a task, both in a group or individually. If a problem is not dealt with correctly or a problem is not planned for in advance, this could have a severe impact on the quality of the final outcome. Our group will deal with all problems together and with help of a risk list so we know how to overcome any problem that may occur.

Throughout this project there are many problems that can occur, from software malfunctions to group fallouts. No matter what the problem is or what scale it is, it is important to be prepared and know what actions must be taken to overcome this problem with minimal impact on the group. We decided we would use 5 simple and basic steps that we would follow to overcome any problem that may rise:

1)What is the problem?

During this step, we will ask ourselves just what the problem is. It is always important to pin point the problem and agree as a team just what that problem is so no-one is confused and we can all start working on it with no doubt.

2) What caused the problem?

This is the step where we've all agreed what the problem is and now we can start to get to work on fixing it. Before we can though we need to work out what caused it. This is where we all start to use testing techniques both individually and as a team so we have all basis covered.

3) What effect has this problem had?

This is possibly the most important step. We need to identify what impact the problem has had to our project. Whether it has affected our time constraints and how badly. If it is a software problem, is it fixable? If it is a personal problem, how badly as if affected the group? This is where we can use the risk list to identify just how badly it will affect our project so we can plan ahead.

4) Plan of action?

After we've discovered the problem and we know what has caused it and we know how badly it has affected our group, we know need to go about fixing that problem. This is where we all pull together as a group and delegate problems to different members to deal with the problem the most efficiently and in a way we are all happy with.

5) Results?

Finally after the problem is fixed and we are all happy we've been through all of the steps and used the risk list to our advantage, we need to make sure this problem doesn't happen again. If it does we know how to deal with it by using past experience of the problem or use different, logical techniques to overcome the problem quicker.

Using these 5 steps, along with the risk list, this will help us to overcome a whole range of problems quickly and efficiently whilst having a minimal impact on the teams attitude and the project as a whole insuring the best possibly quality of our final project and a much happier team.

Sign off process - Process and Documentation:

The sign off process is a crucial part of the project and will be carried out in almost every phase of the project in small steps. This process has to be finished with high quality so that client will be able to understand and use the product with all its features.

To make sign off process easier and successful we have to be:

- **Systematic**: each member of the project will have to meet their deadlines and when needed, work together to reach the target.
- **Consistent**: in every milestone/phase of the project we will have to check if the documentation is complete.

To make sure that the project is understandable each member of the project will have to create documentation of their work and it should be finished after every milestone/phase of the project. It is very important to make sure that documentation is easy to read and that is why we will use clear and understandable sentences, that the client with no IT knowledge will be able to understand.

The documentation will be included in every phase of the project e.g.:

- **Code** – Code needs to be clear and easy to read. Everything should be explained and documented to make it more useable for others.
- **Diagrams** – UML use case diagrams are useless without accurate documentation. All diagrams should include description.

Documentation has a massive impact on creating user guide, which is very important and it must be done before final closure of our project. Documentation and user guide are essential especially when some members of the team could be swapped.

Fit for purpose:

To make sure that our project is fit for purpose we will have to go through several tests and check our work in every possible way. It can save time and reduce our faults to a minimum.

Internal testing is one of the software quality assurance processes. The test will verify and validate the software. We will use this to check if our software is consistent with our specification and/or documentation; and it will provide information about software stability.

There are many ways of testing:

- **Static testing**: is good for very first testing because we do not actually run the software. The test consists of automatic (e.g. compiling) and manual checks of code to find errors. This method checks the correctness of the code and allows you to decide if the software is ready for more detailed testing. In static testing we can distinguish basic code analysis and accurate search for common mistakes.
- **Dynamic testing**: it can be used after successful static testing. This one is testing the process of whole or part of the software while it is running and comparing the output with the expectations.
- **White-box testing**: This relies on testing software through giving on the input such data to the software that it can go through all implemented paths.

We will make sure that our software is fit to purpose through testing it by someone who is unpredictable, and it will be one of our user. Such test is called **user acceptance testing** and main purpose of this test is to allow some of the users to run and try to use our software. This test can be used only when our software is nearly finished so it must be provided with basic functionality. We will be able to check if our software is coded correctly and how it interacts with actual user.

Adapting to requests/change control

Developing a project while under the constant threat of an uncontrollable change is going to be one of our biggest problems. A request to make a change to the project can come at any time so we will always need to be ready for when that time comes. Being caught off guard with an unexpected request for making a change can cause us to panic, waste time and lose progression on the project's development. We thought it be best to develop a solid change management process to prepare ourselves for any changes that can be unexpectedly requested. The process we came up with consists with 5 simple steps to follow to ensure that the best outcomes are made:

1) A request to make a change to the project is received

When a request to make changes is being received we must ask the client to be as specific as possible so we may analyse the request and be able to fully understand what is being asked of us. We must also ask if their requested change requires any needs that may be new to us, such as using new software/technology.

2) Create/Update project change log

The project change log will contain all information regarding any changes that are requested and how we deal with them in the next steps. This log is very useful as we can possibly use information from previous changes to help us through new changes.

3) Analysis the change

As a team we must meet and analysis the requested change, discussing and giving thoughts on how this is going to change the project's current development, any potential impacts it could have and how we are going to be able to implement this.

We have to know what this change will do to our current project development plans such as needing to change dates and timing of working on sections of the project or if significant changes to parts of the project will also need to be changed.

We must discuss potential impacts that making changes can cause to the project. With this we will make changes to your risk list to include all information about the impacts caused by the change. If an impact seems too risky then we will have to get in contact with the client and report it.

Implementing the change into the project can be the easiest but also the hardest step as some changes can be minor or simple while other could have complex solutions and take up a lot of time. The discussion of this step needs to be well thought about to minimize complexion, risk and most importantly waste of time.

4) Make plans for the change

When we have completed our full analysis and discussions about any requested changes, we will need to make changes to our current project plans so that the changes can be added and the development of the project will be back on a straight path to success. All potential impacts will be added to the risk table and from the details of our discussion in change analysis; we will plan out the implementation for the changes we will be making, who is doing what and the timing of work on it.

5) Take actions for the change

Once all previous steps are completed the final step kicks into motion which is to use all plans to implement all changes to the project. Having experience using this change management process will make things easier once/if new changes occur again.

If impacts of the change seems to be too risky or the solution too complex then our leader will have to contact the client, inform them of the problems and make suggestions of slight changes that can be made to their request so that all problems can be effectively solved.

Configuration Management

Since this project will rely heavily on changes, from the team during development and also from the client, one of the processes highly regarded is the software configuration management(SCM). Due to this, we will be using it heavily in the project development life cycle.

We will be using this by identifying certain attributes of the software throughout various points in the development stage and will also perform control of changes. This will help us ensure software integrity and essentially, the quality of the software and overall project.

The SCM generally has four procedures we will follow to ensure a high quality SCM process is implemented.

1) Configuration identification.

This is the process of identifying the attributes of every configuration item of the product(in this case, configuration item has a end-user purpose). These are recorded in documentation and base lined.

2) Configuration control.

This is a process where there have to be approval stages required to change a configuration item and it's attributes(this is used when something in the software or development needs to be changed). By going through approval stages and looking back at the identification, we can ensure the changes are necessary in the system compared with the added time and cost.

3) Configuration status accounting

This is the ability to record and report on baselines associated with each configuration item at any moment in time. This is used so we always are aware of outstanding change requests and that no time or effort implementing the wrong system. This problem will be avoided by knowing at any moment the configuration status of the design description.

4) Configuration audits

This process is broken down into two audits, function and physical. This will occur at delivery or at the moment of effecting a change. The objective of the functional audit is to provide independent evaluation of a software product, verifying the actual functionality and performance is consistent with the requirements of the client. This audit is held prior to delivery. The objective of the physical audit is to provide an evaluation of a software product's configuration items. This is to ensure that all components map to their specification. The use of auditing the configuration is we don't miss features and the system remains maintainable.

Using these steps, we will be able to make changes during development and be confident the changes will not create consequences for the project. Using these steps, we will also be able to ensure software integrity and traceability throughout the software development cycle.