Quality Management Plan

Jensenligan

February 2014

1 Todo

Copy paste these to a valid section, rewrite a bit and if not complete add a slash todo
http://latexforhumans.wordpress.com/2009/03/13/todonotes/).
If you dislike the document structure change it, David
Simply think, what tasks can/do we do to directly improve quality of the product. Or, what are the factors that somehow influence the quality of the product. Everything is valid!!

- Add more todos, increase detail of todos
- How using pivotal improves document-ability and traceability etc.
- How using git improves x.
- How using scrum improves y (see scrum pm).
- How using a (frequent) demo based development improves z (right prioritisation of functionality etc) (see scrum pm)
- Unit tests!!!! A lot.

2 Developer interest and Productivity

We see developer interest and productivity as our main factor influencing the quality of the device. The development of the device is done by a small team of 6 members, which means that all members have extremely high impact on the final product. The performance of our team is directly relative to the productivity of each member. It is therefore very important that each member feels connected with the product, its purpose and eager to work with it.

Therefore we internally develop a product (the laser simulated turret) which may not be seen as either serious or interesting to the external viewer. The laser simulated turret, which is nothing more than a toy product, is however a product that all developers feel closely engaged and motivated to work with.

However, the projects external identity (the intimidating security camera) is more formal and has a more direct customer. This allows for all formalities regarding customer discovery and product verification to be executed, all-while keeping the developers interested and entertained at work.

Following the previous statement that developer productivity directly increases software quality, this section discusses tools and practices prioritized to improve on this productivity.

2.1 Development process and tools

Please note that for classical development, more in depth verification, testing and general quality management tasks need to be executed and documented. However, by working in scrum with a focus on agile practises, there is much more pressure on our development process and its tool-set. Following the agile manifest, "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.".

2.1.1 Git and source version control

By using git we reduce needed maintenance of build breakage and conflict merging. It also eases the process of reverting to previous development versions as well as the maintainability of multiple versions.

2.1.2 Coding standard

By establishing a strict coding standard we improve cohesion of code between developers and thereby its understandability. We also reduce confusion and chances of arguments regarding syntax conventions.

2.1.3 Demo driven development

By having tightly repeated demo's executed, the development team is enforced to focus on prioritizing aspects impacting the demo's of the device. This directly assist in reducing scope creep.

2.1.4 Pivotal

Pivotal is a collaborative task management tool. By using a task management tool we improve upon the document-ability and traceability of our work. Together with pivotal we can assure that no one works on the device without quickly stating what they will do first. Thereby all work should at least have some base for documentation as-well as a time and date. This also helps constrain scope.

3 Design

3.1 Modularity and Object oriented design

By focusing on modularity, that is, when logically separable parts of the product is also encapsulated and separated in code, we directly increase the maintainability and flexibility of our code. It gets easier to overview as well as understand and edit. In other languages, encapsulating via classes is the only way to go. However, in python, only separating using multiple files is enough, and this will be our main goal. To try and separate code in as many files possible.

A good object oriented design maximises cohesion and minimizes coupling and dependency. Minimal dependency reduces costs of late bug detection as well as increase extensibility and modifiability. Low coupling also enforces information hiding which improves code readability and understandability.

4 Responsibilities

By working in scrum, all responsibility, including quality assurance's is equally spread over team members. It does however lie in the responsibility of our product owners that the backlog stories they add and prioritize lie in the best interest of the customer.

5 Product Performance Management Plan

Performance is in most cases indeterminstic, hence, no optimisation shall be applied without profiling and testing. All performance optimisation is initially discarded in favor of productivity. This means the development team mainly shall favor python during the complete development time. When and if cases arise that gives performance related issues and if these issues are confirmed as important by the product owner and or customer, only then are the team viable to rewrite specific parts of the product in a lower level language such as C/C++. These rewrites shall only be on parts shown as bottlenecks during profiling.

6 Quality Assurance Requirements

6.1 Safety

The laser used shall be of no class stronger than 2 and shall not be able to cause any eye damage by direct exposure.

6.2 Security

SSH or physical access to the device shall be required to configure the device.

6.3 Usability and ease of use

The user should not be required to have any experience with ssh or Linux to use the final product. For configuration, only editing a config file found on the sd-card is required.

7 Verification management plan

Requirements verification and validation will be done repeatedly every other week with the customer involved. Requirements validation is customer based, meaning that all information used to state validity should come directly from the customer, rather than any written specification.

8 Standard practices

9 Faults management plan

10 Correction costs and early detection

Although a good object oriented design minimizes dependency, all software will contain parts that are transitively dependant on others. This often means that if a bug is created but not discovered, there are risks that continued work of the software becomes dependant on the bug. Hence, the later a bug is discovered, the more dependency and the higher cost for change.

10.1 Testing

To discover bugs as fast as possible, and reduce the chance of bug dependency, unit tests will be used. Unit tests provide automatic verification of functionality as well as announce breakage of dependable code.

10.2 Problem reporting and corrective action

Bugreport/handling

11 Risk management

Adverse event

12