**Felucca**

**MITS CMU**

# Quality, Risk and Configuration

## Project Felucca

## Team BugHunter

# Authors

Di Mu (dimu@andrew.cmu.edu)
Sudi Lyu (sudil@andrew.cmu.edu)
Zihao Zhou (zihaozho@andrew.cmu.edu)
Guancheng Li (guanchel@andrew.cmu.edu)

# Revision History

| Version | Date | Change | Updated by | Reviewed by |
|---|---|---|---|---|
| 1.0 | Apr 29th, 2020 | Add Quality, Risk, Configuration | Zihao Zhou | |
| 1.1 | May 1st, 2020 | Adjust format | Zihao Zhou, Di Mu, Sudi Lyu, Guancheng Li | |

# Contents

# 1 Quality Management

## 1.1 Quality Attributes

### 1.1.1 Isolation

Pharos tools might be used to analyze malicious code, any damage caused by execution should be contained.
- **Source**: Any users
- **Stimulus**: Malicious binary updated by user
- **Artifact**: Job Execution Layer
- **Environment**: Normal job execution conditions
- **Response**: Tasks are executed in an isolated environment.
- **Response Measure**: Damage inside the execution environment is contained

### 1.1.2 Maintenance

Pharos tool set is a tool set that is evolving, Felucca should support future versions of Pharos and be easy to maintain
- **Source**: Developers
- **Stimulus**: Developer update Pharos tool set
- **Artifact**: Tools Manager
- **Environment**: Developer role condition
- **Response**: Felucca and Pharos decouple and version check
- **Response Measure**: Felucca could support developers' update on Pharos

### 1.1.3 Availiablity

Different roles require different connectivity environments, so Felucca should be able to provide full functionality in each environment.
- **Source**: Developers
- **Stimulus**: Developer update Pharos tool set
- **Artifact**: Tools Manager
- **Environment**: Developer role condition
- **Response**: Felucca and Pharos decouple and version check
- **Response Measure**: Felucca could support developers' update on Pharos

## 1.2 Quality Goals

The goal of quality management is to :
- Meet all  'must have' user cases, most of (at least 3 out of 4) 'nice to have' user cases and half 'optional' user cases
- Meet all quality attribute requirements
- Code conforms to our style guideline
- Provide user-friendly UI at front-end to ease the pain of complex arguments in Pharos tools.
- No active defect

## 1.3 Tests

The following is the detail of our group how to perform a quality test:

| Testing Strategy | Technique | Tool | When to perform |
|---|---|---|---|
| Unit Test | Intra-module testing | Unittest, Pytest | Any code update |
| Integration Test | Boundary testing, combinatorial testing, mock module | Python stubs | Merge with development branchs |
| System Test | Requirement based testing | Manual | Module updating, Merge with master branch |
| Usability Test | Peer review | Manual | Front-end delivery |
| Acceptance Test | Client review | Manual | After delivery |

Table 1.1 Detail of quality test

## 1.4 Workflow

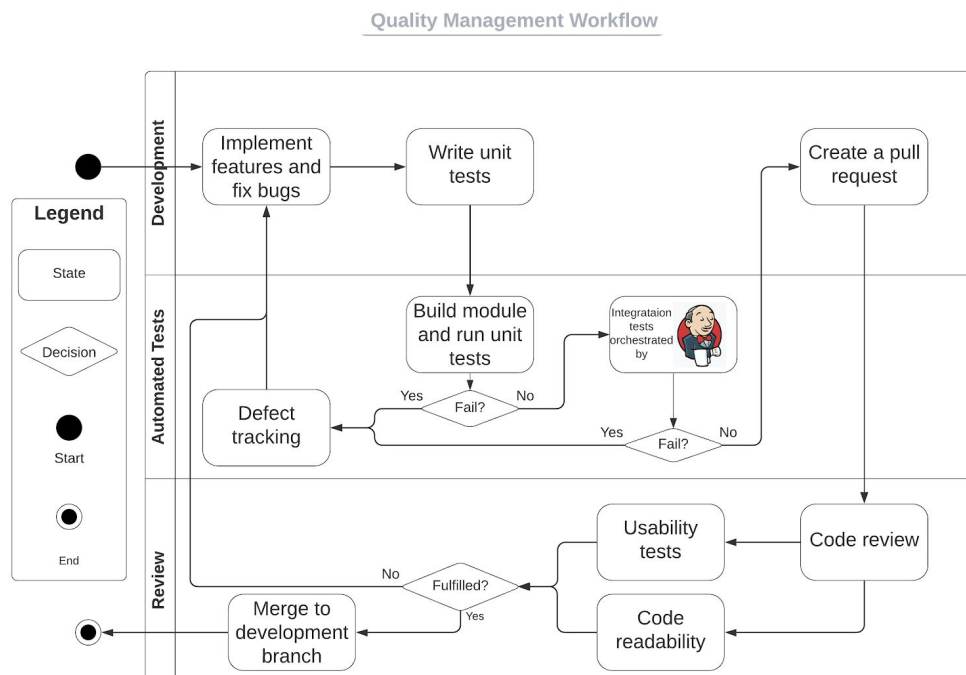The workflow diagram is shown below.



Figure 1.1 Workflow of Quality Management

## 1.5 Criteria

We summed up five criteria as our quality metric.
- **Unit Tests**
  - Code coverage beyond 80%
  - Pass all new-written and old tests
- **Integration Tests**
  - API-based, Written by other developers
- **System Tests**
  - Requirement-based, cover 100% non-functional requirements
- **Usablity Tests**
  - Group Review,  Peer Review, Client verification
- **Code Review**
  - Code Readability
  - Group Review deadline

## 1.6 Defect Tracking

Defects are discovered in unit or Integration Tests, which are caused by insufficient API design or architecture inconsistency.
- **Defect Reviewing**
  - All discovering defects are reviewed in the next sprint meeting to figure out if it is solvable , if it is worth solving, how many hours we need to fix it and the priority of this defect.
  - All solvable and worth solving defects are marked as "active" and issues are created.
  - Important defects will be reported to client
- **Defect Fixing**
  - All "active" defects would be fixed based on its priority

# 2 Risk Management

## 2.1 Risk Management List

| Risk | Part | Impact | Mitigation | Status |
|------|------|--------|-----------|--------|
| No allocator to allocate job to execute | Basic Architecture | High | Add a job manger in our basic architecture | Closed |
| Feluca may be inconsistent with Pharos tool | Code | High | Dig into API and the logistic of Pharos | Ongoing |
| Lack GUI design experience | Background | Low | Communicate with client frequently | Ongoing |
| Sequence Diagram lacks some essential functions | Sequence Diagrams | Medium | Split original Sequence Diagram to several Sequence Diagrams in detail | Closed |
| All team members lack MangoDB development experience | Background | Low | Learning MangoDB while developing | Ongoing |
| If executable binary code is malicious, it is not safe to store in disk after execution | Safety Issue | High | We are considering not to store the binary code after Pharos execution | Ongoing |
| Lack an efficient way to communicate with the client with a strict schedule | Teamwork | Medium | Refine our major problem to several concrete and detailed questions, and make sure the answers of these questions can solve our current problem | Ongoing |

Table 2.1 Risk Management List

## 2.2 Risk Management Process

**Identify risk:** Risk identification is a process for identifying and recording potential project risks that can affect the project directly. This step is crucial for efficient risk management throughout the Felucca project.

**Report to all members:** Risk can occur in many parts of our project, such as Basic Architecture, code, Safety Issue and so on. If one of our team members discovers a potential

risk in our project, he will report to others and schedule a discussion in the next meeting.

**Discuss in meeting:** In the meeting. All team members will discuss this potential risk and decide if it belongs to a real risk. If it is a real risk in Felucca currently, we will put it in our Risk Management List and discuss how to mitigate this risk and come up with a mitigation plan.

**Execute mitigation plan:** Once we have a mitigation plan for this risk, we will execute it as soon as possible. For example, we lack an efficient way to communicate with the client with a strict schedule, and our  mitigation is refining our major problem to several concrete and detailed questions. Next time we will change our way to communicate with our client and make sure the answers of these questions can solve our current problem.

**Monitor risk:** After executing a mitigation plan, we will keep monitoring this risk and make sure whether everything goes well.

# 3 Configuration Management

## 3.1 Code Artifacts: Managed by GitHub



- Each function developed on dedicated function branches
- Merge to master development branch after done
- Open a maintenance branch
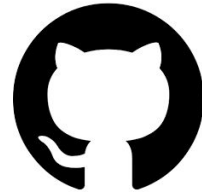
## 3.2 Document Artifacts

- Keep a log for each document artifact
- Log is updated with version number, date of change, document type, description of change and editor each time a modification is made to a document log

**Requirement Document**

| Version | Date | Change | Updated by | Reviewed by |
|---|---|---|---|---|
| 0.1 | Mar 2nd, 2020 | Add user behaviors & template of use case | Di Mu<br>Sudi Lyu | |
| 0.2 | Mar 9th, 2020 | Add the initial version of use cases | Di Mu<br>Sudi Lyu<br>Guancheng Li<br>Zihao Zhou | Hasan Yasar |
| 0.3 | Mar 13th, 2020 | Add use case diagram | Di Mu<br>Sudi Lyu | Jeffrey Gennari |
| 0.4 | Mar 27th, 2020 | Add glossary | Sudi Lyu | |
| 0.5 | Apr 2nd, 2020 | Update use cases<br>Fix some contents | Di Mu<br>Sudi Lyu<br>Guancheng Li<br>Zihao Zhou | Hasan Yasar |
| 0.6 | Apr 6th, 2020 | Add deployment mode | Sudi Lyu<br>Di Mu | Jeffrey Gennari |
| 1.0 | May 1st, 2020 | Re-organize contents | Guancheng Li<br>Di Mu | |

Table 3.1 Requirement Document Log

**Architecture Document**

| Version | Date | Change | Updated by | Reviewed by |
|---|---|---|---|---|
| 0.1 | Mar 21st, 2020 | Construct overall architecture | Sudi Lyu | |
| 0.2 | Mar 27th, 2020 | Reconstruct overall architecture based on pharos toolset requirement | Sudi Lyu | |
| 0.3 | Mar 29th, 2020 | Add glossary | Sudi Lyu | |
| 0.4 | Apr 12nd, 2020 | Add time sequence diagram | Guancheng Li | Jeffrey Gennari |
| 0.5 | Apr 24th, 2020 | Update detailed flow diagram and time sequence diagram to unify the glossary | Sudi Lyu, Guancheng Li | Hasan Yasar |
| 0.6 | Apr 30th, 2020 | Redraw all flow diagram using Lucidchart | Sudi Lyu | |
| 1.0 | May 1st, 2020 | Reformat and add more description | Sudi Lyu | |

Table 3.2 Architecture Document Log

**Statement of Work**

| Version | Date | Change | Updated by | Reviewed by |
|---|---|---|---|---|
| 0.1 | Feb 7th, 2020 | Structure of the document | Sudi Lyu | |
| 0.2 | Feb 10th, 2020 | Scope of work | Guancheng Li | |
| 0.3 | Feb 11th, 2020 | Overview and goals | Di Mu | |
| 0.4 | Feb 12th, 2020 | Add module diagram and time sequence diagram | Sudi Lyu | |
| 0.5 | Feb 17th, 2020 | Add requirements | Sudi Lyu Guancheng Li | |
| 1.0 | Feb 23rd, 2020 | Update requirements Add plan and schedule | Sudi Lyu Di Mu Zihao Zhou Guancheng Li | Hasan Yasar |
| 1.1 | Apr 3rd, 2020 | Add Pharos Knowledge | Sudi Lyu | |
| 1.2 | May 1st, 2020 | Re-organize the contents | Guancheng Li Di Mu | |

Table 3.3 Statement of Work Log