Planning Phase

Software Requirements Plan

Each requirement will be explicit and mutually exclusive. Requirements will be classified as functional, non-functional or safety requirements. Requirements will be accompanied by rationale and assumptions if applicable. Software functionality will be divided into distinct subsystems or features which provide modularity and easy integration. Requirements will be assigned priorities based on the severity classifications of the associated hazards and overall importance to the system, and development of relevant code will be scheduled according to these priorities. Traceability between requirements, code and tests will be documented in traceability matrices and updated as new requirements are discovered. Requirements will be implementation-free and "negative requirements" will be avoided, as they are hard to verify. Requirements will be quantitatively verifiable, except certain non-functional requirements (e.g. usability, security), which will be verified by code-inspection or other forms of qualitative testing. All requirements will trace to higher-level requirements, except for derived requirements.

Two requirements documents will be produced: one for high-level requirements and one for low-level requirements. Do research and ask questions from Stuart to make sure we produce good requirements. Link low-level requirements to high-level requirements.

Do we have to write Severity Classifications? For each risk, you would put a severity classification.

High-Level Requirements:

These are requirements which define broad attributes of the program such as reliability

Low-Level Requirements:

The low-level requirements provide more specific details on the implementation of the program.

Design level classes, more broader, could be traced to 2-3 classes in

Use JavaFX and specific interface features. This is a derived requirement instead of low-level requirement. This could be a design constraint.

Note:

All low-level requirements must be traced back to high-level requirements. Not all derived requirements must be traced back to high-level requirements?

Software Development Plan

All software will be developed in Java and will conform to the adapted “Power of ten” rules for Java, created by David Pierce. All development will be undertaken using the latest version of Java (Java 22.0). All code will be statically analyzed using SpotBugs, and all warnings will be effectively treated as errors. The IDE used will vary between group members, however at least one group member will use Eclipse, as it offers a rich plethora of static analysis tools.

Software Verification Plan

Each module will be tested independently under both expected and anomalous conditions using unit testing. Additionally, the system as a whole will be tested using integration testing, and physical inputs and metrics will be tweaked to simulate unusual conditions (e.g. stormy weather, failed sensors etc). Throughout the development phase, code quality will be ensured using various static analysis tools, and usability of the system will be assessed manually. The adapted “power of ten” rules for Java will be followed stringently to ensure good-quality code. Eclipse will be used for coding as it supports a large range of static analysis tools, including definite-assignment analysis and pattern-based tools, such as SpotBugs. Regression tests will be rerun after every major change to the codebase, as this will ensure that new bugs are not introduced. Fault injection will be used to test the module’s robustness under unusual conditions, and to estimate the quality of our integration testing. Assertions will be used to check preconditions, postconditions and invariants, and any violation will trigger the return of an explicit error to the caller. Return values will be checked and errors caught and handled appropriately.

No proofs required.

Configuration Management Plan

Use version control for the softwasensre. GitLab will be used for version control of documents necessary for certification, including plans, requirements, traceability matrices, code and test cases. After deployment of the system, potential hazards and risks will be identified by logging, and documented. Any updates to the system after release will follow a similar process of certification. Features will be developed independently in

separate branches and will later be merged into the final system. This development process ensures changes can be rolled back independently of other features/subsystems of the system. Issues will be reported using the GitLab issue tracking system, and each issue will result in a new branch. The baseline will not be changed without consultation among the team. Traceability documentation will be updated as changes occur, to ensure that if a requirement is changed, the development team knows which modules must be changed/removed. As required by DO-178C, less critical data items, such as verification results and software configuration management records, will be less stringently managed, however they still require retrieval and traceability to source. For the purposes of this project, we are ignoring most requirements of CC1 (control category 1) and CC2, except for retrieval, traceability, baselines and problem reporting.

How you are controlling the infrastucture: tools, version control. If code is changed, testing suite has to be re-run.

Quality Assurance Plan

Coding standards will be verified by visual inspection or static analysis tools. Traces between code and requirements will be verified by reading and verifying function comments, which will follow the Javadoc commenting style. Static analysis tools will be used to verify that all warnings have been dealt with. Usability of the interface will be verified by giving documentation of user testing procedures to certification authorities.

Need someone else to review our processes (such as tutors or the lecturer).

Could also be coding standards such as how do we check off each other's code. If one person commits, at least one other person should review the code.

How do we do the weekly meetings? Quality of tool and processes.

Glossary

Baseline - A stable and protected snapshot of software at a given point in time