Train Trax: Train Monitor for Positive Train Control Test Beds

Team Operation Document

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author |
| 1.0 | 10/12/2015 | Initial Version. | Stephen Jalbert  Rashad Madyun  Corey Sanders |
| 1.1 | 10/20/2015 | Updating document to describe Team’s Risk Management Strategy and The Test Manager Role | Corey Sanders |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 2](#_Toc432397512)

[1 Tools 4](#_Toc432397513)

[2 Team Process 5](#_Toc432397514)

[3 Configuration Management 6](#_Toc432397515)

[3.1 Version Controlled Documents 6](#_Toc432397516)

[3.2 Change Control 6](#_Toc432397517)

[3.2.1 Adding New Document 7](#_Toc432397518)

[3.2.2 Updating Document 7](#_Toc432397519)

[3.2.3 Software Changes 7](#_Toc432397520)

[4 Team Communication Strategy 8](#_Toc432397521)

[5 Team Quality Assurance Strategy 8](#_Toc432397522)

[5.1 Formal Review Process 9](#_Toc432397523)

[5.1.1 Formal Review Roles 9](#_Toc432397524)

[5.1.2 Formal Review Checklist 10](#_Toc432397525)

[5.2 Quality Assurance Audit Criteria 10](#_Toc432397526)

Figure 1. Team Roles

|  |  |
| --- | --- |
| Role | Team Member |
| Project Manager | Corey Sanders |
| Developer | All |
| Reviewer | All |
| Moderator | Corey Sanders |
| Configuration Manager | Rashad Madyun |
| Backup Configuration Manager | Stephen Jalbert |
| Scribe | All |
| Test Manager / Quality Assurance | Stephen Jalbert |

# Tools

Configuration Management

* GitHub ([www.github.com](http://www.github.com))
  + Git Repository Hosting
* JetBrain’s YouTrack website (<https://www.jetbrains.com/youtrack/>)
  + Track and manage development issues and to store work products.
* Google Drive (drive.google.com)
  + Version Distribution
  + Release
* Canvas (<http://www.canvaslms.com/>)
  + Product Delivery
* Git
  + Documentation version control

Documentation

* Microsoft Office 2013 Suite including, Word, Excel and Powerpoint
  + Documentation modification and review comments.
* Google Drive (drive.google.com)
  + Documentation change collaboration

Requirements Analysis and Design

* Umlet 13.3
  + Model Creation
* ArgoUML 0.34
  + Model Creation

Implementation

* Eclipse 4.5.1
  + UI and Train Navigation Library Development
* Arduino SDK 1.6.5
  + Motion Detection Unit Development
* SQLite 3.8.11.1
  + Train Database Development

Testing

* Junit
  + UI and Train Navigation Library Testing
* Arduino Unit (<https://github.com/mmurdoch/arduinounit>)
  + Motion Detection Testing
* C Unit (<http://sourceforge.net/projects/cunit/>)
  + Motion Detection Testing

# Team Process

The process used by our team will be the traditional waterfall process model. This process model is the sequential software development process in which progress will flow steadily downward like a waterfall through the phases of conception, initiation, analysis, design, construction, testing, production implementation and maintenance. The following gives shows a step by step transition of how our team development cycle will be implemented

* Determine Scope of work
* Development a Rough Magnitude of Estimate
* Development Requirements Specification
* Development of Test Plan
* Development of Design methodology
* Development of Prototype
* Implementation of Software Design
* Execution of Test Plan
* Correction of any Software Defects
* Delivery of Software
* Maintenance

# Configuration Management

The team plans to utilize the services of GitHub for hosting the change tracking and version control repository. This will allow us to store each version of the work products, and keep track of all changes made, and retrieve the latest version of the work product.

## Version Controlled Documents

* Project Description
* Rough Order of Magnitude (ROM) Estimate
* Hardware Selection Document
* Operation Process Document
* Requirements Specification Document
* Software Design Document
* Test Plan Document
* Test Description Document
* User Manual
* All Software for Project

## Change Control

When making changes to a product, the revision history will be kept track of within a table in the document for all changes. Version numbering will start with 1.0 and increment a whole number for every major change and in decimal increments for minor changes. The first version of a product is one that has gone through a peer review from all group members and has been accepted. Google drive will also be utilized by the group to edit and modify work products that are currently under development. Once a work product has been approved and accepted this will be documented within YouTrack and the document will then be moved to the GitHub repository and versioned.

Before each product release, the customer will be emailed to confirm that the product meets all specifications. The customer’s approval will then be documented and stored in the configuration management system. A version will become a formal version after the first release to the customer. From there on, changes must be formal and be toward a new version. The version is finalized after it follows the configuration management process and is approved by all of the team and is released to the customer.  
A full product release will be a group of all of the work products that are included as part of the system expected for the customer to use. This includes the documents listed in the Version Controlled Documents section.

The tags in Git will be used to mark releases and versions. For a product release, new branch will be created inside GitHub to act as the baseline. Each version controlled document will also include a Revision History page describing the changes made since the original document that lead to the creation of a given version.

### Adding New Document

1. Create initial document.
2. Upload document to Google Drive.
3. When draft is complete setup review in YouTrack.
4. Follow Steps in Formal Review Process.
5. Add Version history header and document as 1.0.
6. Upload to GitHub Repository.

### Updating Document

1. Download current version of document from GitHub repository.
2. Save a working version of the document in Google Drive.
3. When draft of update is complete setup review in YouTrack.
4. Follow Steps in Formal Review Process.
5. Update version history header for minor updates increment by .1 for major updates increment by 1.0.
6. Upload updated document to GitHub repository.

### Software Changes

Changes to software are organized along branches of software development in GitHub. The branches are as follows: the mainline branch, release branches, and feature development branches. The mainline branch stores all of the latest reviewed changes to the software product. Release branches are forks of the mainline branch created when a version of the product is released or delivered to the customer. Feature development branches are branches for each developer to implement the current feature that they are assigned to work on. All development branches pull from the mainline branch to ensure that they have the latest changes prior to working on a new feature. When the feature has completed being implemented. The changes in the feature branch are submitted for review according to the Review Process. When the changes have been reviewed, corrected, and approved, the changes are pushed into the mainline branch, and changes from the mainline branch are pulled into the feature development branches. If changes need to be made to a released version of software, then a new feature branch is created for the change, and the process continues just like a change to the mainline branch. Once the change is pushed to the release branch, the change must at least be implemented to the mainline branch like any other feature change and undergo the same change process again. If the change is applied to any other versions, there must be approval from both the team and the customer.

Steps for Changing Software

1. Pull changes from the mainline branch to the target feature branch.
2. Implement Feature
3. Create an Issue for review.
4. Link changes in Feature Branch to Issue
5. Follow Steps in Formal Review Process.
6. Push changes from the feature branch to the mainline branch.
7. If Release is required, then perform tests according to test plan.
8. Go to step 1 if needed for any corrections due to outcome of tests.
9. Create Release Branch.

# Team Communication Strategy

The team has decided to meet weekly via skype on Mondays and Thursdays at 7:00 pm. In person meetings and additional meetings will be scheduled as necessary. The team will also use Canvas and UAH email for general communication during the semester. The team also has scheduled a weekly meeting with the customer on Mondays to discuss progress on the project, and exchange information.

# Team Quality Assurance Strategy

The Quality Assurance (QA) plan describes how the team will implement a procedure to ensure that all products are delivered with the highest quality possible. Each team member will review all work products before acceptance and delivery to the customer. Informal peer reviews of all work products will be conducted multiple times a week. An informal peer review is considered any review where feedback is provided to the author, but a record is not keep about the details of the feedback. Formal peer reviews are reviews in which the members formally discuss issues and document any issues related to the work product. The formal review type will be round robin with comments and suggestions from each team member provided. During the review, all work products will be reviewed for overall quality and correctness. After the review is performed, the Configuration Manager is responsible for verifying that all of the accepted comments and input from each team member has been addressed, and that the overall product has all of the required components. These issues will be documented via YouTrack, and once addressed the issue will be marked as fixed by the configuration manager. Before final acceptance of each work product a Formal Peer Review must be performed. The roles of the formal review are described below in section 1.4.2.

## Formal Review Process

* Create the initial copy of document on Google Drive for comment.
* Once the draft is complete enough to turn in, Google Drive will stop being used as the primary storage for the document, and a copy of the document will be exported.
* Create a You Track Issue to Review the Item.
* Assign the Issue to the Author.
* Add reviewers as Watchers to the issue.
* The exported copy of the document is attached to the issue.
* Comments for the review are attached to the issue either as a copy of the document with review features enabled or as a direct comment.
* A Header is added to the YouTrack issue to describe people involved in the Review, and who the moderator is.
* Once all of the changes have been completed and reviewed by the moderator, the moderator updates the issue with the final outcome and sends the configuration manager the final copy of the document to version as well as an exported PDF that captures all of the review material for the review issue, including proof that all reviewers have no further feedback on the document. This can be implicit as long as it is recorded that the moderator indicates that the issue is  
  completed.
* A change cannot be considered completed until it has successfully performed the test required per the Test Plan.

### Formal Review Roles

This section describes the roles held by participants in a formal review.

* *Project Manager:* Responsible for overall planning, coordination, and managing of the project tasking. The Project Manager ensures that the product is delivered on time and on budget.
* *Reviewer:* All members are reviewers regardless of what other roles they have, examining the work product for defects, using the review checklist as a guideline. While reviewing, the reviewer will take note of any defect found or question that needs to be answered. The Reviewer finds defects with the product.
* *Scribe:* There must be one scribe per formal review. During the meeting, as the moderator is walking through the work item and reviewers are discussing the defects they discovered, the scribe will record these defects in the section of this document that corresponds to the work item being discussed. The scribe ensures that defects found with the product are recorded.
* *Moderator:* The role of the moderator is to oversee the review and drive the discussion. This person will step through the work item, asking if reviewers have any defects that need to be discussed for that increment. The Project Manager will normally fulfill the role of the moderator, unless specified otherwise. Responsible for verifying that all of the comments and input from each team member has been addressed, and that the overall product has all of the required components included before submission. The Moderator ensures that the team is building the product correctly.
* *Configuration Manager:* Responsible for organizing changes made artifacts. The configuration manager is responsible for baselining the system, tracking versioning information, and preparing releases. The Configuration Manager ensures that team can track to all changes made to the product.
* *Quality Assurance / Test Manager:* Responsible for conducting system-level testing of the system. The test manager is responsible for ensuring that the system as whole meets requirements and interoperates correctly with customer environments. The Test Manager ensures that the team is building the correct product.

### Formal Review Checklist

* This checklist contains things to look for while reviewing a work product.
* Is the portion of the product under review?
* Consistent with the rest of the product?
* Not missing any project requirements given by the Instructor?
* Free of any ambiguity?
* Relevant throughout?
* Free of grammatical errors?
* Free of unnecessary jargon/complexity?
* Traceable back to the systems specification
* Complete and precise
* Exhibit characteristics of high quality design (reusability, reliability, low coupling, high cohesion etc.)

## Quality Assurance Audit Criteria

QA Audits will be performed using a defect-based checklist focusing in three specific areas. The Configuration Manager is responsible for making sure audits are performed and the feedback is received from each group member. The first area of focus is the defect of omission. The following questions will be used as a basis to review the requirements for defects of omission:

* Is this concept precisely defined somewhere?
* Is this acronym defined?
* Are these definitions summarized in the glossary of terms?
* Is this objective operationalized through specific requirements?
* Are those requirements sufficient to ensure this objective?
* Is the rationale for this requirement (or assumption) made explicit somewhere?
* If this requirement or assumption relates to another, is the latter specified somewhere?

The second area of focus is the defect of unmeasureability. The following questions were used as a basis to review the requirements for defects of unmeasureability:

* Is there a fit criterion associated with this quality requirement?
* Is this fit criterion stated in terms of measureable quantities and measurement protocol?
* Can test data be derived from this statement to test that the implementation meets it?
* Is this statement stated in a way that discriminates it from alternative options?

The third area of focus is the defect of ambiguity. The following questions were used as a basis to review the requirements for defects of ambiguity:

* Can this statement be interpreted differently in different relevant contexts or by readers from different background?
* What are the possible interpretations?
* Are the other statements using this term with a different meaning?

# Risk Management Strategy

The team is using a proactive risk management strategy where risks are identified as early as possible, then monitor, and controlled where possible.

## Risk Evaluation

Each risk that is identified for the project are evaluated to determine the likelihood of the risk and the impact of the risk. The likelihood of the risk is the probability that the event described by the risk will occur. The impact is an evaluation of how much the event will affect the project if it occurs. Examples of things that impact a project include changes to the amount of work that is needed to complete the project, the availability of team members to work on the project, and the changes to the timeline or schedule for the project. The likelihood of a risk combined with its impact produce the severity of the risk. A qualitative approach has been chosen by the team to describe the likelihood, the impact, and the severity of a risk. There are three tiers for each: low, medium, high. Each tier is described below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Low | Medium | High |
| Likelihood | Not likely to occur. | Realistically can be forseen to occur, but would not be surprising if it did not happen. | Very likely to ocurr. |
| Impact | Little to no change to the project cost or schedule. | A change to the project cost or schedule that would hinder the project but not prevent it from being accomplished | A major change to the project cost or schedule. |
| Severity | Plan Action should be taken if possible. | Planning for risk should be performed. | Action must be taken as soon as possible |

Below describes how Risk Severity Relates to Likelihood and Impact:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Severity Matrix | | Impact | | |
| Low | Medium | High |
| Likelihood | High | **Medium** | **High** | **High** |
| Medium | **Low** | **Medium** | **High** |
| Low | **Low** | **Low** | **Medium** |

## Addressing Risks

* Do Nothing
  + Team agrees to take no action to address the risk.
* Mitigation
  + Team agrees to take an action to either reduce the impact or likelihood of the risk.
* Circumvention
  + Team agrees on a plan to avoid the risk completely.
* Resolution
  + Team agrees on an action or series of actions that will fix the problem that the risk poses.

## Identification

The weekly meetings used to coordinate with the team and the customer are the primary way that risks are identified. Weekly meetings involve both planning and reporting of obstacles or changes.

As part of the weekly meetings among the team, each team member is to discuss any obstacles they are experiencing with the tasks that they are currently working on. If any of these obstacles continue to exist, after the meeting. It is the responsibility of the project manager to record the obstacle as a project risk.

As part of the weekly meeting among the team, each team member is to discuss the overall plan for the next week’s development and the current plan for upcoming targets on the schedule. Any issues that arise out of the discussion involving either execution of the plan or feasibility of accomplishing the targets on the schedule should be recorded by the project manager as a project risk.

As part of the weekly meetings with the customer, the team is to ask the customer about any changes in their expectations for the product and record feedback and concerns about the projects progress. If there are any outstanding project changes or customer concerns that could not be addressed or planned to be addressed during the meeting. The project manager is record them as issues for the team to work, and record any risks involved with addressing those issues.

## Monitoring

Each week it is the responsibility of the project manager to review the status of any outstanding risks for the project and update any changes to the risks as necessary. Risks are tracked as part of the Project Management Document.

## Control

As part of the weekly meetings among the team, it is the responsibility of the project manager to discuss with the team any outstanding risks that he/she believes that action is possible to address or is severe enough that additional planning is necessary for addressing. Any decisions made by the team to address the risk must be recorded as part of the risk tracking in the Project Management Document.

# References

1. https://en.wikipedia.org/wiki/Risk\_management#Identification