Train Trax: Train Monitor for Positive Train Control Test Beds

Software Development Process and Configuration Management Plan Document

Version 1.5

12/01/2015

Corey Sanders

Rashad Madyun

Stephen Jalbert

# 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 |
| 1.2 | 11/6/2015 | Updating to give further clarification about the workflow for configuration management, and improve the organization of the document. | Rashad Madyun  Corey Sanders |
| 1.3 | 11/16/2015 | Renamed the “Team Operation Document” to the “Software Development Process and Configuration Management Plan Document”.  Refactored the labelling used for the document.  Added labels to all tables and figures.  Fixed where the Process Chart in the Team Process section hid text from the section. | Corey Sanders |
| 1.4 | 11/24/2015 | Reorganized document so that the division of the software development process and configuration management is clear.  Also, a distinction was added for software and hardware configuration management.  Also, added some details on what is considered a feature and how all of the modules of Train Trax get incorporated into baselines/releases.  Lastly, added some details about how meeting minutes for team meetings are archived. | Rashad Madyun  Corey Sanders |
| 1.5 | 12/1/2015 | Added reviewer checklist for software and documents, added content to identify sections for both hardware and software.  Added page numbers and table of contents for figures.  Added section to explain the stages of the team process, moved team process to earlier in document. | Rashad Madyun  Corey Sanders |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 2](#_Toc436769845)

[1 Purpose 7](#_Toc436769846)

[2 Software Development Process 7](#_Toc436769847)

[2.1 Team Process 7](#_Toc436769848)

[2.1.1 Process Description (Overview) 8](#_Toc436769849)

[2.1.2 Planning 9](#_Toc436769850)

[2.1.3 Requirements 9](#_Toc436769851)

[2.1.4 Design 9](#_Toc436769852)

[2.1.5 Implementation 9](#_Toc436769853)

[2.1.6 Test 9](#_Toc436769854)

[2.1.7 Maintenance 9](#_Toc436769855)

[2.1.8 Delivery 9](#_Toc436769856)

[2.2 Tools 10](#_Toc436769857)

[2.3 Reusable Software Products 11](#_Toc436769858)

[2.4 Team Communication Strategy 11](#_Toc436769859)

[2.5 Team Quality Assurance Strategy 11](#_Toc436769860)

[2.5.1 Quality Assurance Audit Criteria 11](#_Toc436769861)

[2.6 Risk Management Strategy 12](#_Toc436769862)

[2.6.1 Risk Evaluation 12](#_Toc436769863)

[2.6.2 Addressing Risks 13](#_Toc436769864)

[2.6.3 Identification 13](#_Toc436769865)

[2.6.4 Monitoring 14](#_Toc436769866)

[2.6.5 Control 14](#_Toc436769867)

[3 Configuration Management Plan 14](#_Toc436769868)

[3.1 Configuration Management Roles (Hardware & Software) 14](#_Toc436769869)

[3.2 Software Configuration Management 15](#_Toc436769870)

[3.2.1 Version Controlled Software Documents 15](#_Toc436769871)

[3.2.2 Change Control 15](#_Toc436769872)

[3.3 Hardware Configuration Management 26](#_Toc436769873)

[3.3.1 Hardware Version Controlled Documents 26](#_Toc436769874)

[3.3.2 Change Control 26](#_Toc436769875)

[3.4 Maintaining Traceability Requirements (Hardware & Software) 29](#_Toc436769876)

[3.5 Releases (Hardware & Software) 29](#_Toc436769877)

[3.6 Delivery (Hardware & Software) 30](#_Toc436769878)

[4 References 30](#_Toc436769879)

**Table of Figures**

[Figure 1 Process Work Flow 7](#_Toc436757540)

[Figure 2 Creating a New Document Work Flow 16](#_Toc436757541)

[Figure 3 Updating an Existing Document Work Flow 17](#_Toc436757542)

[Figure 4 Implementing a new software Feature Work Flow 20](#_Toc436757543)

[Figure 5 Software Defect Correction Work Flow 22](#_Toc436757544)

[Figure 6 Creating a New Document Work Flow 27](#_Toc436757545)

[Figure 7 Updating an Existing Document Work Flow 28](#_Toc436757546)

# Purpose

The purpose of this document is to provide information about the work being done to manage the Train Trax project. It is intended to describe the process by which our team will operate for the development of the Trax system.

# Software Development Process

This section describes the process in which our team will operate during the development of the TRAX system.

## Team Process



Figure 1 Process Workflow

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
* Create Project Description
* Create Project Management Workbook
* Development a Rough Magnitude of Estimate
* Create Schedule
* 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

### Process Description (Overview)

The first semester of work developed by the software development team will include the phases of: elicitation, development of a requirements specification, development of a test plan, development of a design methodology, and development of a prototype. During the elicitation stage, the process in which the software development team will have a discussion with the customer to determine what is desired for the project will occur. During this stage a detailed description of the project will be developed, a project management workbook and overall rough magnitude of estimate that includes a project schedule will be created to determine the amount of work required to complete the project. Once the elicitation stage is complete, the requirements specification document will be created from the feedback and interaction with the customer of what is desired for the project. After the requirements phase, a test plan will be developed to ensure that all requirements have been implemented and function correctly once the implementation is completed. The next phase after the development of the test plan is the Development of the Software Design methodology. The design methodology includes a detailed description of the system architecture, detailed description of components and any other design decisions or tradeoffs that need to be documented. Developed concurrently with the Software design will be a prototype of the Train Trax system. The prototype will be developed to show all of the user interface screens associated with the system, as well as an example of the data they will be stored and retrieved from the database by the Train Monitor Terminal, and a demo of the IMU sensor functionality.

The second semester of work completed by the Software development team will include the phases of: implementation of the Software Design, execution of the test plan, correction of any software defects, delivery of the software, and maintenance. The implementation of Software Design stage will begin during the 2nd semester of the two-part class and will continue until all requirements have been implemented. The Execution of Test Plan that was developed during the 1st semester will commence immediately after the implementation stage has been completed. Once all implementation and tests have been completed, the final Delivery of our product will be made to the customer at the end of the 2nd semester of the two-part class. The delivery will consist of the full product release delivered via Canvas as a zipped fill if possible and a physical medium if there are size limitations that prevent it from being sent out in Canvas. All hardware and software associated artifacts will be included in the delivery, including physical devices that will be stored in the Train Lab. Once delivery of the product has been made, if there are any corrections or adjustments necessary, they will be performed by the software development team before the end of the semester. Maintenance beyond the scope of the two semester class will be performed by another software development team.

### Planning

A project description is made to explain the problem that needs to be solved, identify the scope of work for the project, and to identify stakeholders for the project. Next, team members are chosen and a development process is decided and recorded in this document (SDP). Next, an estimate is created for the project based on function points and a schedule is made and archived. The team scheduling tool output is what is archived, including a graphical representation of the schedule.

### Requirements

Multiple meetings with the customer are held to gain insight on exactly what the customer's needs and expectations are for the project. All of the relevant information for the project is captured, such as the design of the existing system, an improved description of the scope of work to be performed by the project, and any constraints stakeholders have for the project, in the Software Requirements Specification (SRS). Formal requirements are captured and early model for the proposed system created from the analysis of formal requirements are also recorded in the SRS. The team should also archive customer approval of requirements which should be expressed over email.

### Design

Analysis models from the SRS are used by the team to design the architecture of the project.  The design chosen for the project is recorded in the software design document (SDD).  It includes models that describe the structure, behavior, responsibilities, and interactions of parts of the system.  The SDD should be sufficient for a separate development team to implement the solution.

### Implementation

Source code and any other artifacts are implement features according to the schedule and are tracked with change requests. As portions of the project are completed they should be integrated together and saved in a version controlled repository.  The exact process of how features should be implemented is described in the Configuration Management Plan (CMP).

### Test

Testing is conducted routinely as each feature for the project is implemented. Passing all required tests are mandatory for considering a feature as completely implemented.  Details for the type of testing that is conducted and how it links to requirements is recorded in the system test plan (STP) . Step-by-step instructions for how to conduct testing for the entire system is detailed in the Test Description Document (TDD).

### Maintenance

Corrections that are needed to be made after a delivery, while the second semester is still in progress, are also tracked as change requests. Approved change requests are implemented, tested, and baselined and all affected work products are updated and reviewed.  The CMP provides exact details on how this is done.

### Delivery

Completed features of the project at a given milestone may be assembled together and baselined into a package for delivery. The package is the loaded on a medium of the customer's preference (likely a zipped file on Canvas and a CD-ROM). When delivered, electronic confirmation of delivery is required. The CMP records exact details for the delivery process.

## 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
* Pencil
  + UI Views
* FluidUI <https://www.fluidui.com/>
  + UI Prototyping

Documentation

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

Standards

* <https://google.github.io/styleguide/javaguide.html>

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

## Reusable Software Products

There are not any existing software products created by the development team or customer that are planned to be reused as part of this project.

## 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. Meeting minutes are recorded for the team meetings in the Version Control System used by the team in addition to the version controlled items discussed in the configuration management plan section.

## 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.

### 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 un-measureability. The following questions were used as a basis to review the requirements for defects of un-measureability:

* 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 seen to occur, but would not be surprising if it did not happen. | Very likely to occur. |
| 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 |

Table 1 RISK Characterisitics Table

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** |

Table 2 RIsk Severity Mapping to Other Risk Characteristics

### 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.

# Configuration Management Plan

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 for hardware and software artifacts, and retrieve the latest version of the work product.

## Configuration Management Roles (Hardware & Software)

This section describes the roles held by participants in changing all software and hardware work products.

* *Project Manager:* Responsible for overall planning, coordination, and managing of the project tasking, including change requests. The Project Manager ensures that the product is delivered on time and on budget.
* *Reviewer:* The Reviewer finds defects with the product(s) involved with a change request in the Formal Review Process. While reviewing, the reviewer will take note of any defect found or question that needs to be answered.
* *Scribe:* There must be one scribe per formal review. During the review, 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 Moderator oversees formal reviews and drives 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.

|  |  |
| --- | --- |
| **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 |

Table 3 Team Roles

## Software Configuration Management

This section describes the process in which configuration management will performed for all of the software components for the TRAX system.

### Version Controlled Software Documents

* Project Description
* Rough Order of Magnitude (ROM) Estimate
* Software Development Process and Configuration Management Plan Document
* Requirements Specification Document
* Software Design Document
* Test Plan Document
* Test Description Document
* User Manual
* All Software for Project

### Change Control

This section describes the process in which changes are made and incorporated into the work products of this project. The primary vehicle for incorporating changes is through a change request. Change requests are made from YouTrack, and a formal review must be made for changes made according to a given change request prior being incorporated into a baseline of a product. Changes added to a baseline are tracked with version information.

Versioning for any controlled work product becomes formal 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/or is released to the customer.  
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. Source code included with a release will include a Release Notes Document that describes the changes and features included in the version as well as revision history.

#### Document Changes

When making changes to a document, 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.

##### Adding New Document

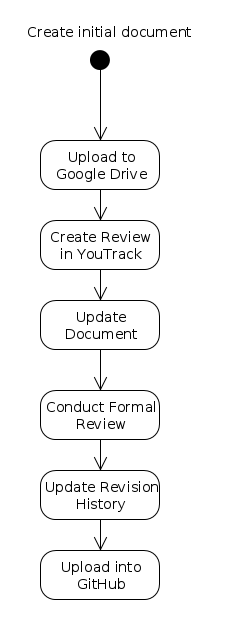


Figure 2 Creating a New Document WorkFlow

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 an Existing Document

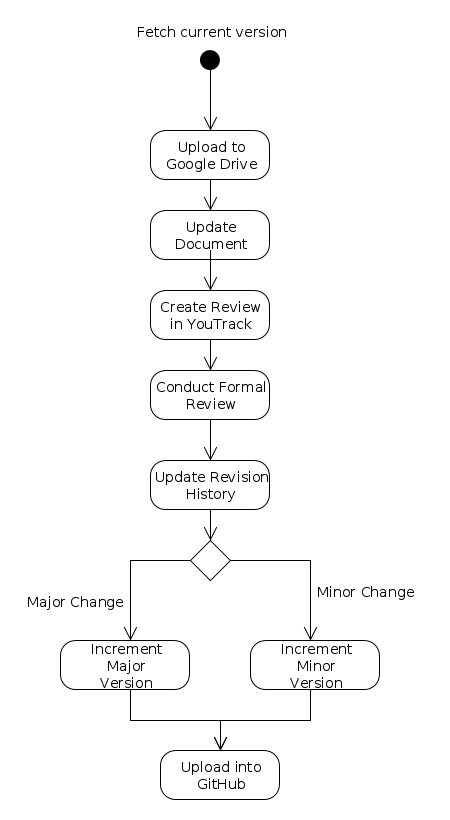


Figure 3 Updating an Existing Document Work Flow

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.

##### Formal Review Process for Changing Controlled Documents

* 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.

###### Formal Review / Reviewer Checklist for Documents

This checklist contains things to look for while reviewing a work product.

* Is the portion of the document under review?
* Document 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 system specification (i.e. Traceability matrices have been updated for documents.)
* Complete and precise
* Exhibit characteristics of high quality design (reusability, reliability, low coupling, high cohesion etc.)

#### Source Code 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. A feature may belong to any of the components of the system since all components belong to each branch. For example, a capability that is added to firmware for a hardware module is considered a feature. Another example of a feature is adding a new function to the GUI. 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 (i.e. software maintenance), 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 Creating a New Software Feature

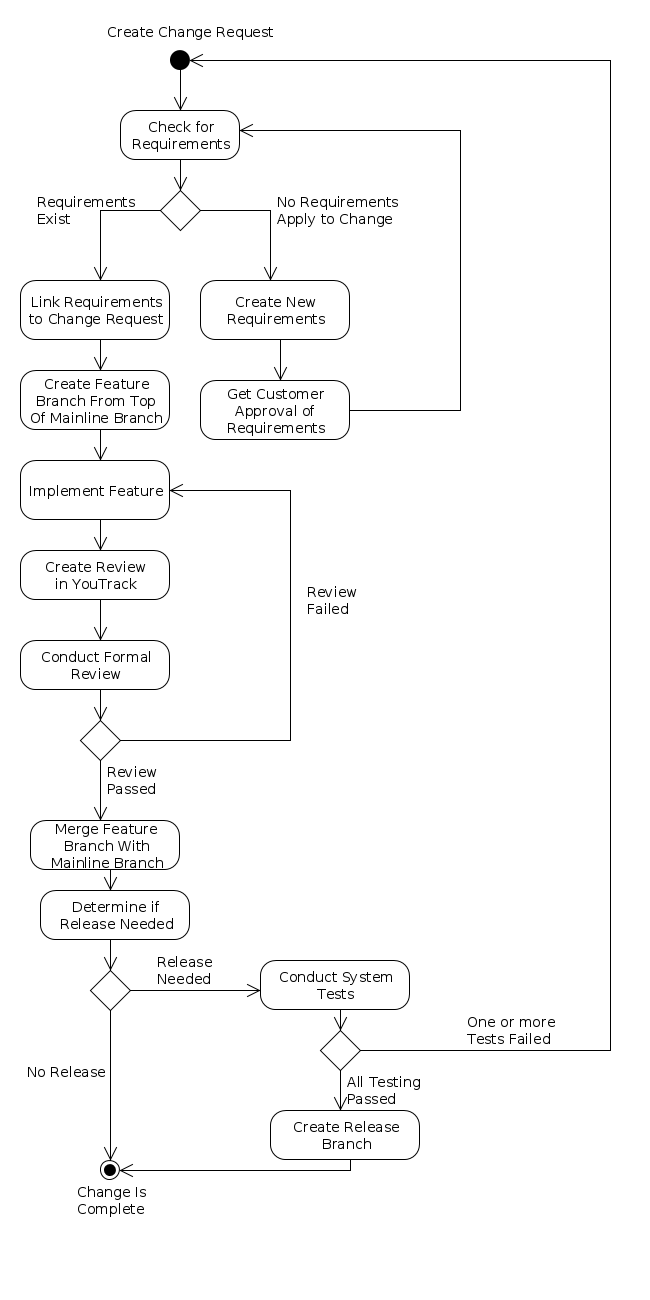


Figure 4 Implementing a new software Feature Workflow

1. A Change Request is issued on YouTrack as a proposed change.
2. The team then links the change to an existing requirement that has been approved by the customer if applicable. If not, requirements must be added for the feature and approved by the customer before the Change Request can be updated to Approved and allowed to be worked.
3. Pull the latest changes from the mainline branch to the target feature branch.
4. Implement Feature
5. Create an Issue for review.
6. Link changes in Feature Branch to Issue
7. Follow Steps in Formal Review Process.
8. Push changes from the feature branch to the mainline branch.
9. If Release is required, then perform tests according to test plan.
10. Go to step 1 if needed for any corrections due to outcome of tests.
11. Create Release Branch.

##### Steps for Software Changes to an Existing Baseline

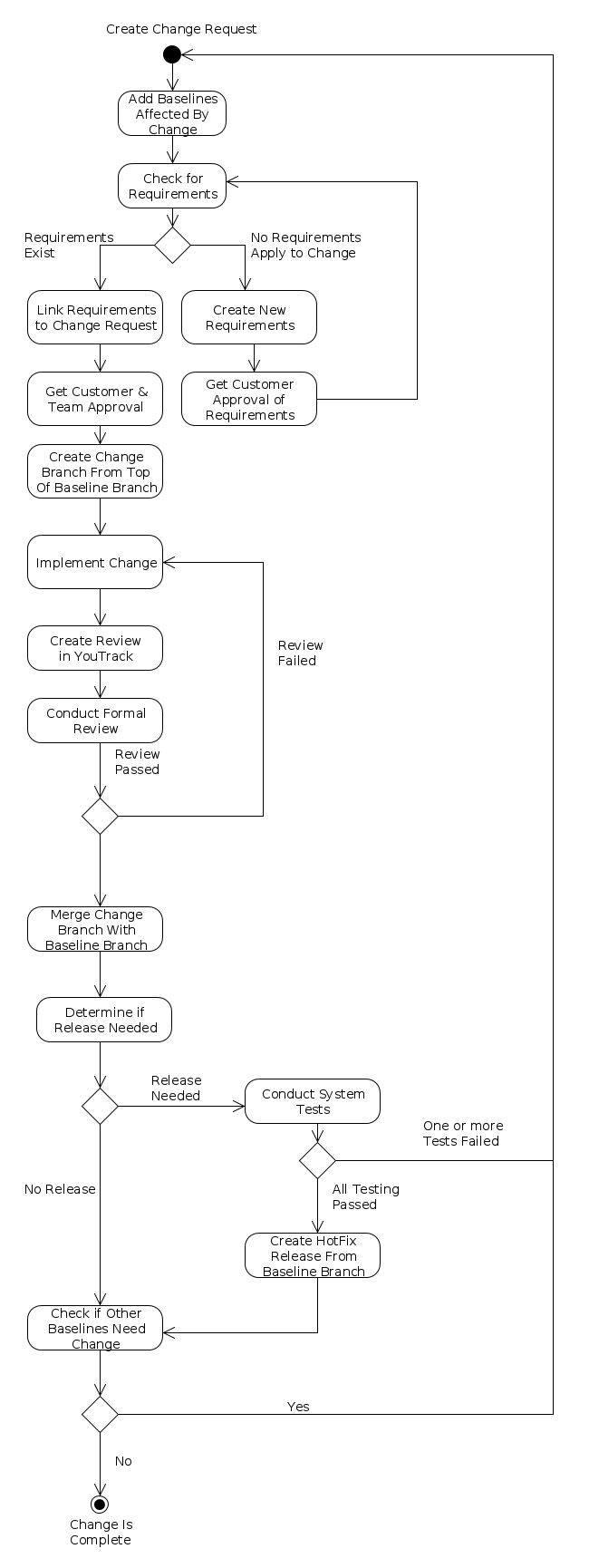


Figure 5 SOFTWARE DEFECT Correction Work flow

1. A Change Request is created on YouTrack for a proposed change.
2. Update the request to include all of the baselines that need to be changed.
3. Link all requirements that are impacted by the change. If no requirements apply, new requirements must be created and approved by the customer prior to working.
4. Once the team and customer approve the change, the Change Request is updated as Approved and details of the approval are recorded.
5. When the change has been is worked, Pull target baseline release branch to the targeted Change/Bug branch.
6. Correct Issue/Implement Change
7. Create an Issue for review.
8. Link changes in Change/Bug Branch to Issue
9. Follow Steps in Formal Review Process.
10. Push changes from the Change/Bug Branch to the target baseline branch.
11. If Release is required, then perform tests according to test plan.
12. Go to step 1 if needed for any corrections due to outcome of tests.
13. Go to step 1 and repeat for any other baselines that need to be changed according to the change request.

##### Formal Review Process for Changing Source Code

* Create the source file in development environment.
* Create a You Track Issue to Review the source files for each predefined component area
* Select task leader for each component area
* Add source file to you/track issue for predefined component / design area.
* Add reviewers as Watchers to the issue.
* Set review period (every week or every so many lines of code added?)
* The exported copy of the code is attached to the issue.
* Comments for the review are attached to the issue
* 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 testing required per the Test Plan.

###### Developer Checklist For Source Code Changes

* Link change request to requirements
* Create change request for new requirements.
* Update requirement traceability matrices
* Update unit tests / integration tests
* Implement change request
* Update system tests
* Update any design changes
* If needed, provide estimate effort required to complete change.
* Update affected work products and baselines

###### Formal Review / Reviewer Checklist for source code Changes

This checklist contains things to look for while reviewing a work product.

* Is the portion of the software under review?
* Code consistent with the rest of the product?
* Not missing any project requirements given by the Instructor?
* Well commented?
* Free of unnecessary jargon/complexity?
* Traceable back to the system specification (Traceability to Change Requests for Code.)
* Exhibit characteristics of high quality design (reusability, reliability, low coupling, high cohesion etc.)

###### Moderator Checklist for source code changes

* Check that change request links to requirements
* Traceability matrix up to date
* Appropriate unit tests and integration tests to test change. Both working and fail
* Verifies that change functions as expected
* Verifies that defects found during review are corrected
* Checks that requirements and design are updated to reflect a change
* Checks that affected work products are documented
* Verifies that changes have been incorporated into baselines correctly

###### Test Manager Checklist for source code changes

* Verifies that testing of the product meets required Code Coverage defined in the System Test Plan.
* New system tests added to test change
* New system tests trace to requirements
* Verifies that system tests, including new ones pass.

###### Configuration Manager Checklist for source code changes

* Make sure that revision history and release notes have been updated correctly
* Makes sure that change requests are linked to affected baselines
* Incorporates changes into baselines

## Hardware Configuration Management

This section describes the process in which configuration management will performed for any work products that describe hardware components for the TRAX system.

### Hardware Version Controlled Documents

* Project Description
* Rough Order of Magnitude (ROM) Estimate
* Hardware Selection Document
* Software Development Process and Configuration Management Plan Document
* Requirements Specification Document
* Test Plan Document
* Test Description Document
* User Manual

### Change Control

This section describes the process in which changes are made and incorporated into the work products of this project. The primary vehicle for incorporating changes is through a change request. Change requests are made from YouTrack, and a formal review must be made for changes made according to a given change request prior being incorporated into a baseline of a product. Changes added to a baseline are tracked with version information.

Versioning for any controlled work product becomes formal 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/or is released to the customer.  
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. Source code included with a release will include a Release Notes Document that describes the changes and features included in the version as well as revision history.

#### Document Changes

When making changes to a document, 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.

##### Adding New Document

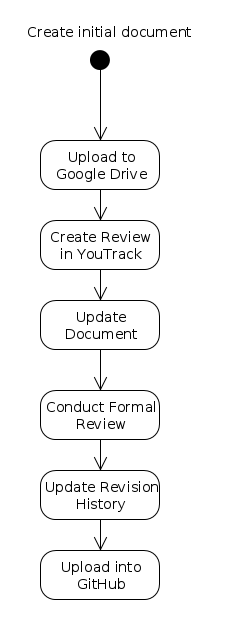


Figure 6 Creating a New Document WorkFlow

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 an Existing Document

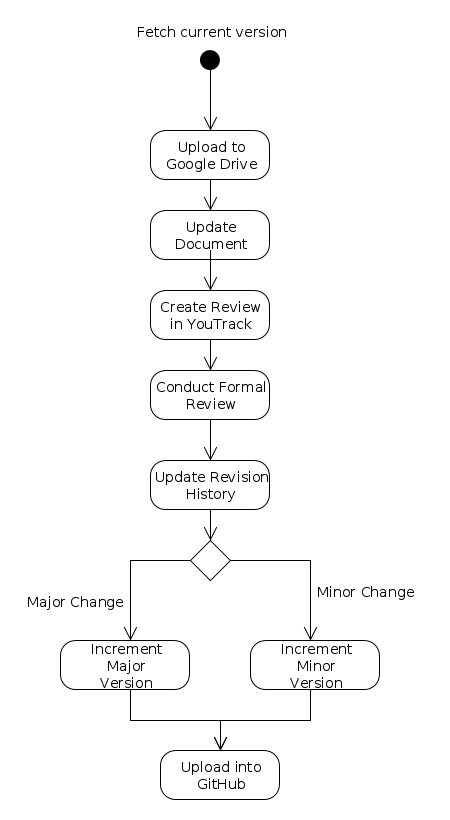


Figure 7 Updating an Existing Document Work Flow

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.

##### Formal Review Process for Changing Controlled Documents

* 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.

#### Firmware Changes

Any source code changes for hardware will follow the same process used for other software in the project described in Section 3.2.2.2

## Maintaining Traceability Requirements (Hardware & Software)

A traceability matrix will be maintained for each controlled document including that has a dependency on the software requirements defined in the Software Requirements Specification (SRS). This presently should be the System Test Plan (STP) and the System Design Document (SDD). It will contain a record of how each work product traces back to an original requirement. Requirements associated with hardware will also be maintained via traceability matrix. It is responsibility of the author to update appropriate traceability matrices when working a change request. Reviewers of the change request should check that the matrices have been updated appropriately. It is the responsibility of the Moderator to make sure that the traceability matrices are up-to-date before updating a change request as complete.

## Releases (Hardware & Software)

A full product release is a group of all of the work products that are included as part of the system expected for the customer to use including all software and hardware components for the system. This includes the documents listed in the Version Controlled Documents section as well.

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.

All hardware and software artifacts are included in the same baseline for a release. There is one mainline branch that all hardware and software changes are incorporated into. Documentation for hardware and software are combined where possible. In particular, the development process, configuration management, design, requirement, and test documentation contain details for both. Firmware developed to run on the hardware uses the same process and practices documented for all other software products. Any third-party firmware or software components used by the project are included in the baseline as part a Release as well. For example, a given baseline will include all of the software developed to date to run on the hardware, any special firmware needed by the hardware, and completed features for the software products, as well as all controlled documentation for the hardware and software.

## Delivery (Hardware & Software)

Deliveries are planned to be made to the customer weekly. A delivery consists of a full product release. Releases will be delivered via Canvas as a zipped file including all items that have been reviewed and approved by the group up to the delivery date. A full product release will also be released via Canvas and possibly a physical medium if there are size limitations that prevent it from being sent out in Canvas. All hardware and software associated artifacts will be included in the delivery.

# References

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