Defect Tracking Policy & Workflow

for

North Shore Extension

**Version 2.0 approved**

**Prepared by Garrett Grube, Jeff Deely, Ritesh Misra, Spencer Worms, Xavier Torgerson**

**Blue Team**

**12/15/2016**

**Table of Contents**

**Table of Contents**

**Revision History**

**1.** **Introduction**

1.1 Purpose

**2.** **Policies**

2.1 Numbering Policy

2.2 Tracking Policy

2.3 Resolution Policy

2.4 Tool Used

**3.** **Bugs**

**4.** **Bugs per Week**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Garrett Grube | 12/7/16 | Document Created | 1.0 |
| Spencer Worms | 12/14/16 | Updated for Work Package 3 | 2.0 |
| Xavier Torgerson | 12/14/16 | Replaced Screenshots with full detail | 2.1 |

# Introduction

## Purpose

The purpose of this document is to state the defect tracking policy used by our software development team. This document will explicitly state and explain the numbering policy, tracking policy, resolution policy, and tools used.

# Policies

## Numbering Policy

We will use GitHub’s numbering policy. GitHub’s policy assigns a unique integer to each defect, starting at zero and counting up in the positive direction by increments of 1.

## Tracking Policy

When the programmer finds a defect, they will add it to the GitHub issue page. The title will have the name of the defect similar to a manner below:

(Train Model) User Interface – about the interface defect

They will then write a more detailed explanation and how to reproduce the defect in the comment section. They will also add a defect label, add which milestone that it will be going to, and who should be working on the defect. The programmers will make comments as they progress on fixing the defect. When they finish the defect and finish testing, they will close the defect. If anyone finds that the fix is not to their satisfaction, they will reopen the issue and notify the person who closed the defect.

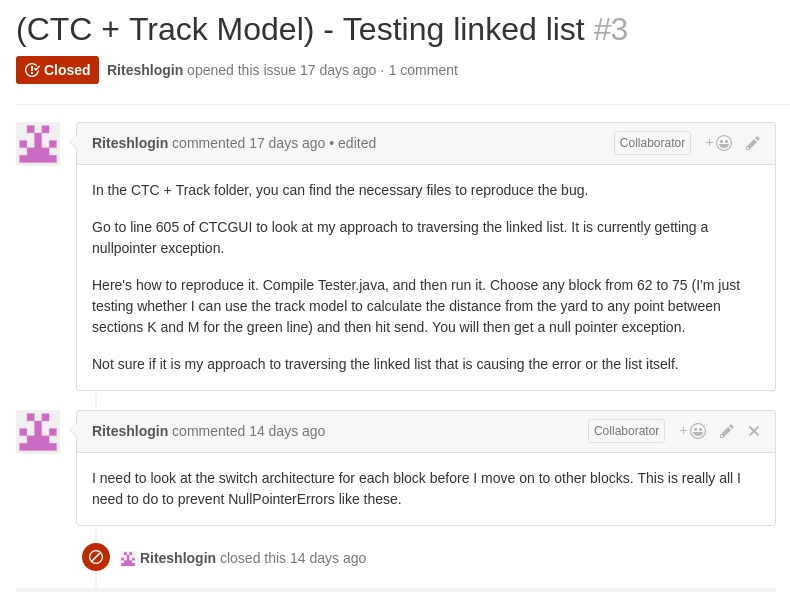
## Resolution Policy

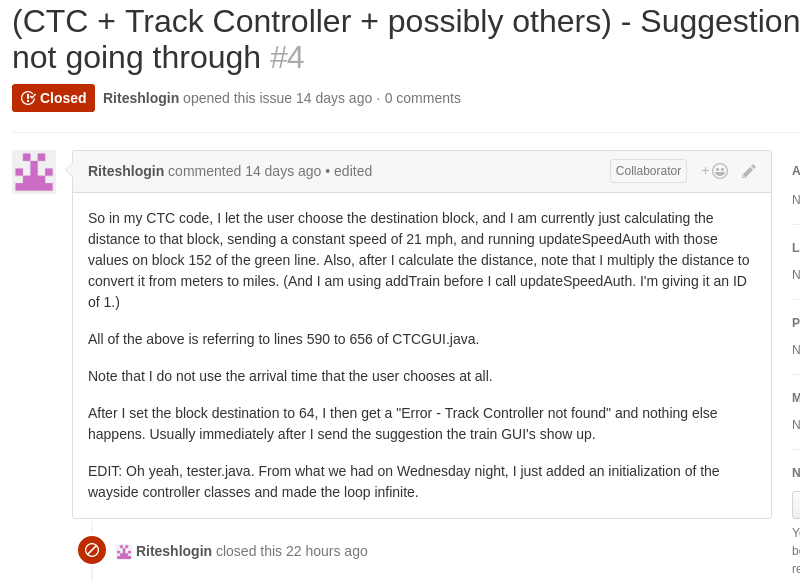
First the programmer will work on important parts of their module before working on any defects to ensure that the modules all have some functionality, even if it does not work 100%. After they have all the functionality, then they will take the furthest reaching and most critical defects first. Then they will work on the defect they are sure that it is completely fixed. Once fixed, they will mark that they finished the defect on the GitHub page.

## Tool Used

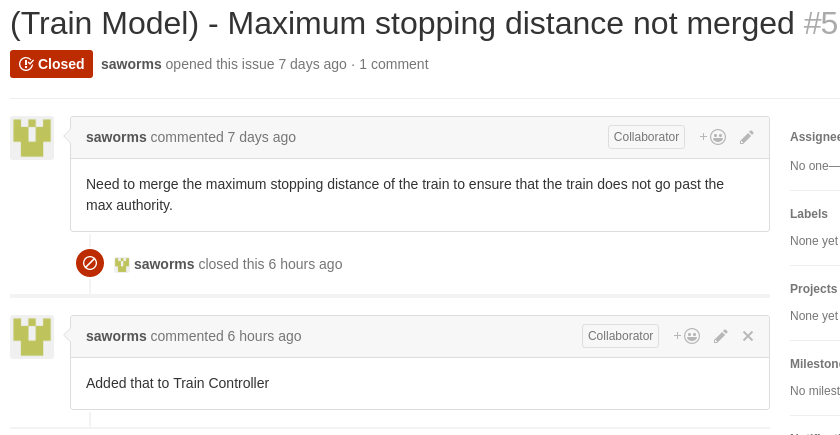
We used the GitHub integrated issue tracking system. This allows any member of the group to dynamically add and change defects and their statuses.

# Bugs

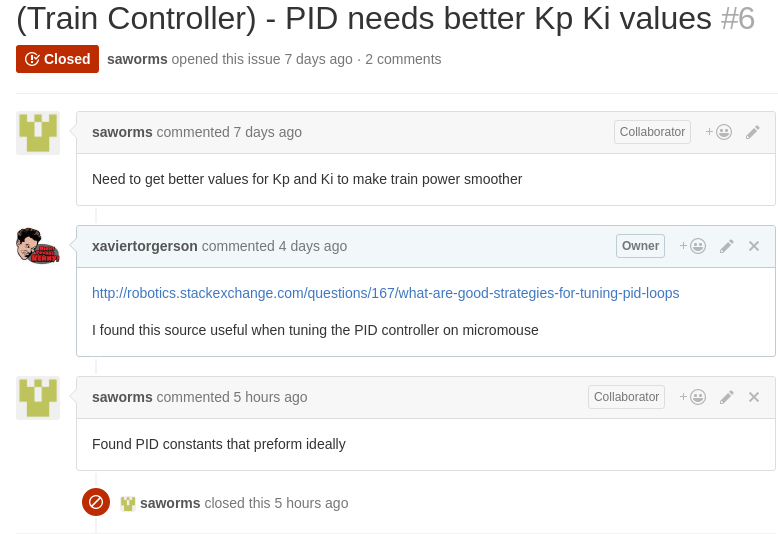
* 1. (CTC + Track Model) - Testing linked list
  2. (CTC + Track Controller + possibly others) - Suggestion not going through



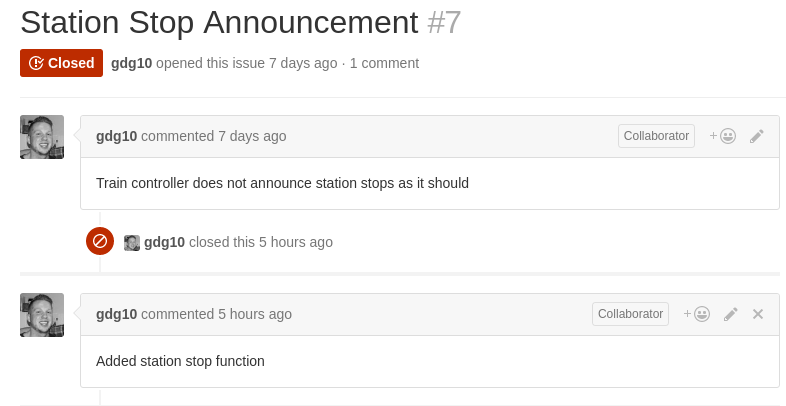
* 1. (Train Model) - Maximum stopping distance not merged



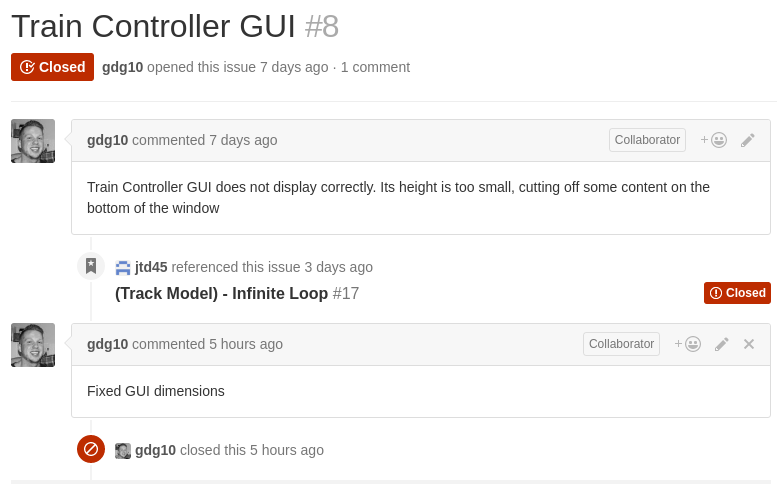
* 1. (Train Controller) - PID needs better Kp Ki values

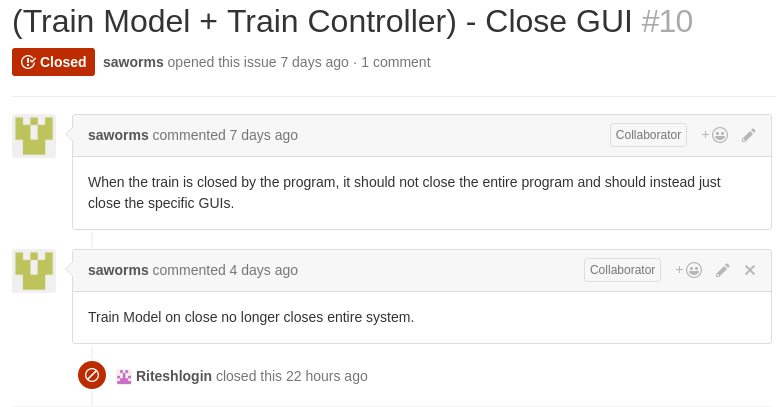
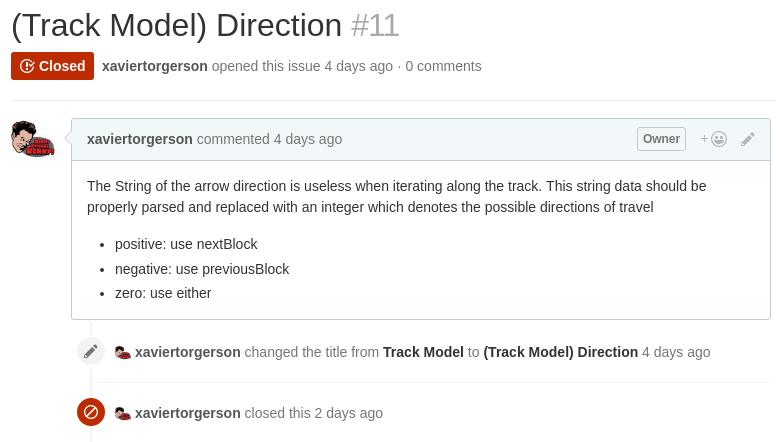


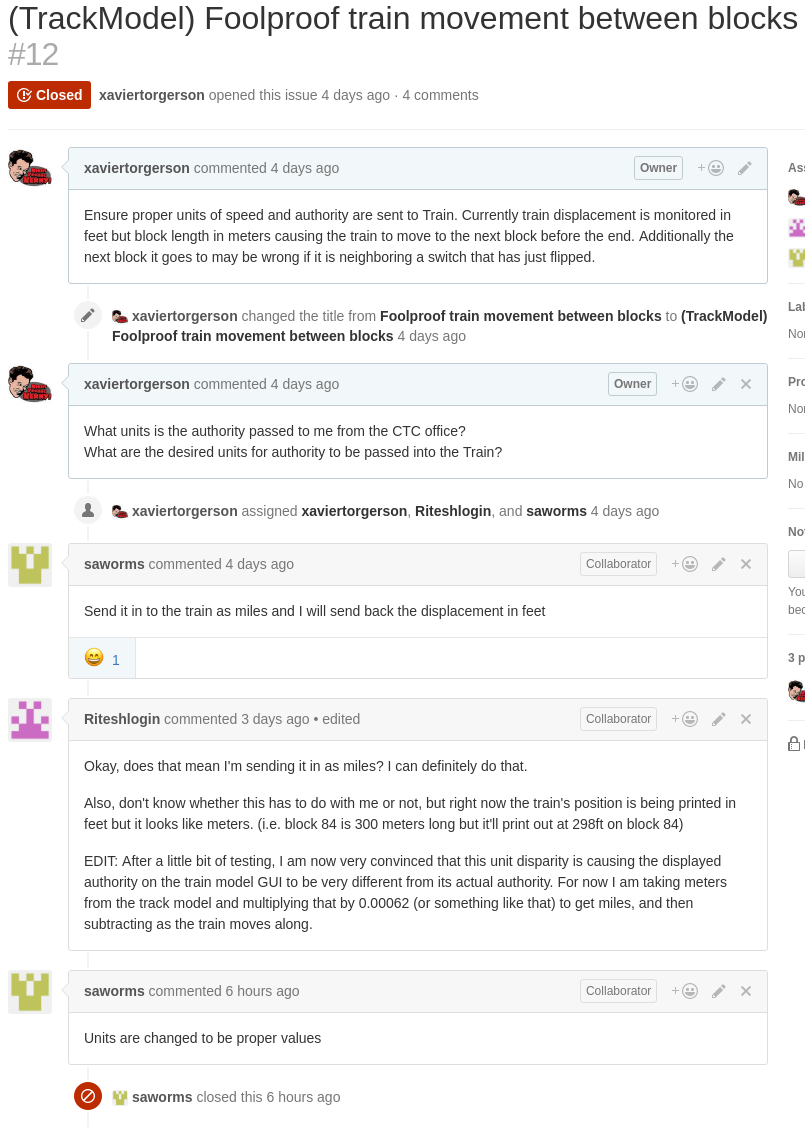
* 1. (Train Controller) - Station Stop Announcement



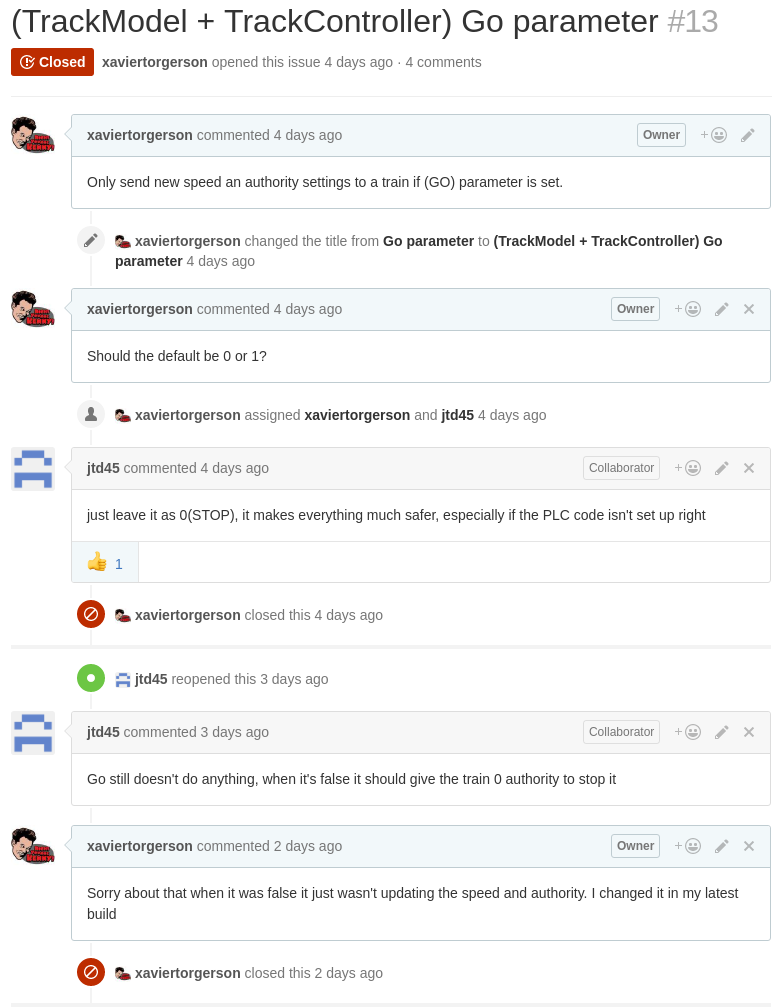
* 1. (Train Controller) - Train Controller GUI



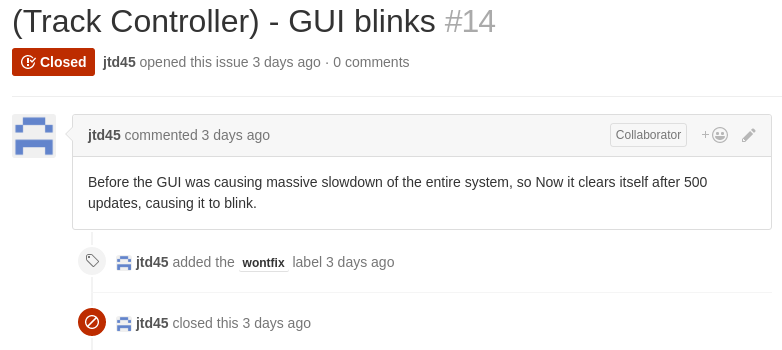
* 1. (Train Model + Train Controller) - Close GUI
  2. (Track Model) - Direction
  3. (Track Model) - Foolproof train movement between blocks



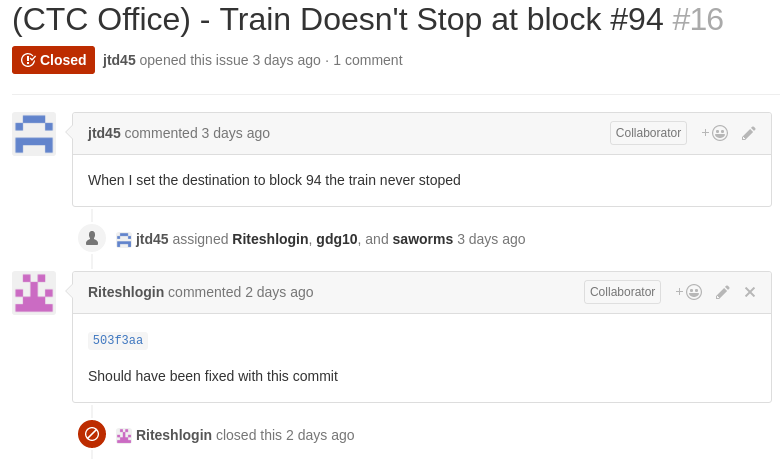
* 1. (Track Model + Track Controller) - Go parameter



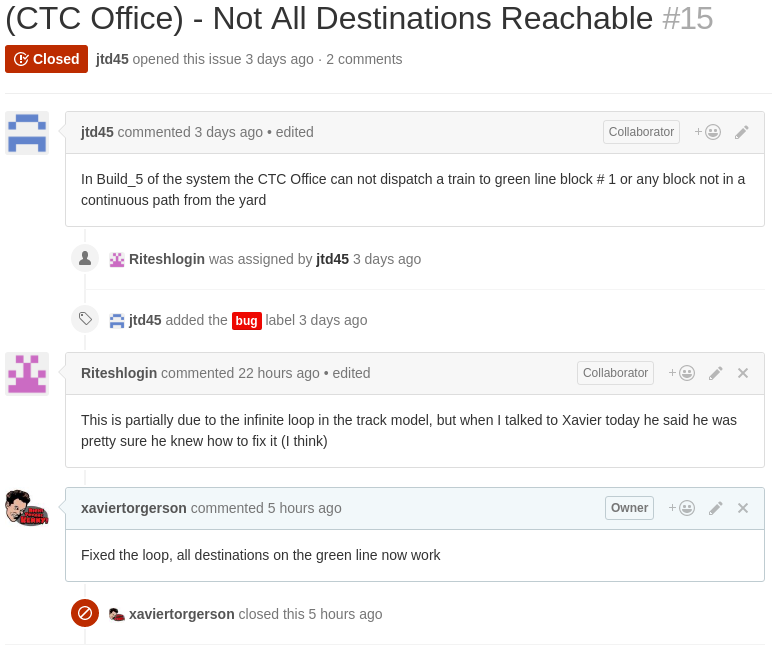
* 1. (Track Controller) - GUI blinks



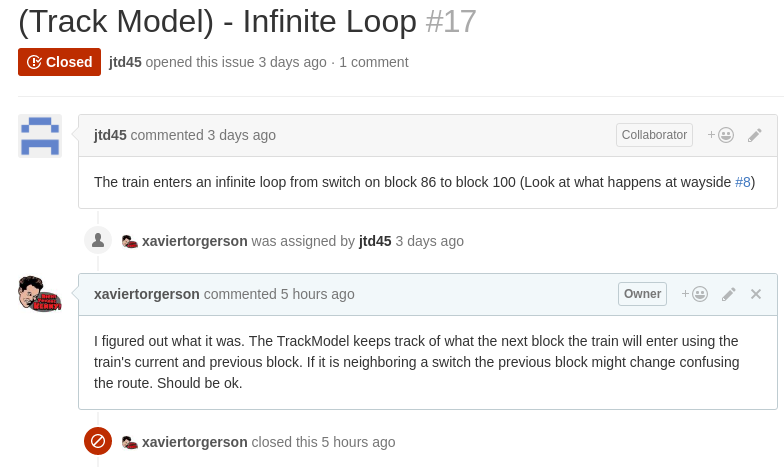
* 1. (CTC Office) - Train doesn’t stop at block #94

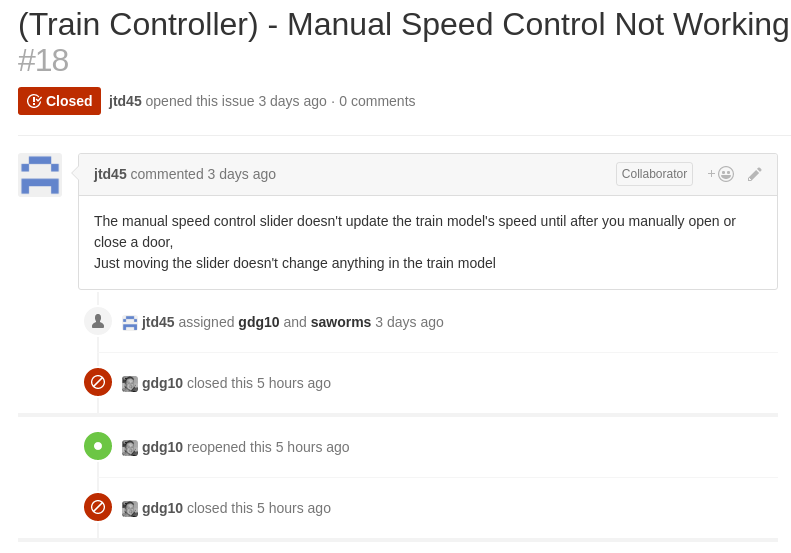


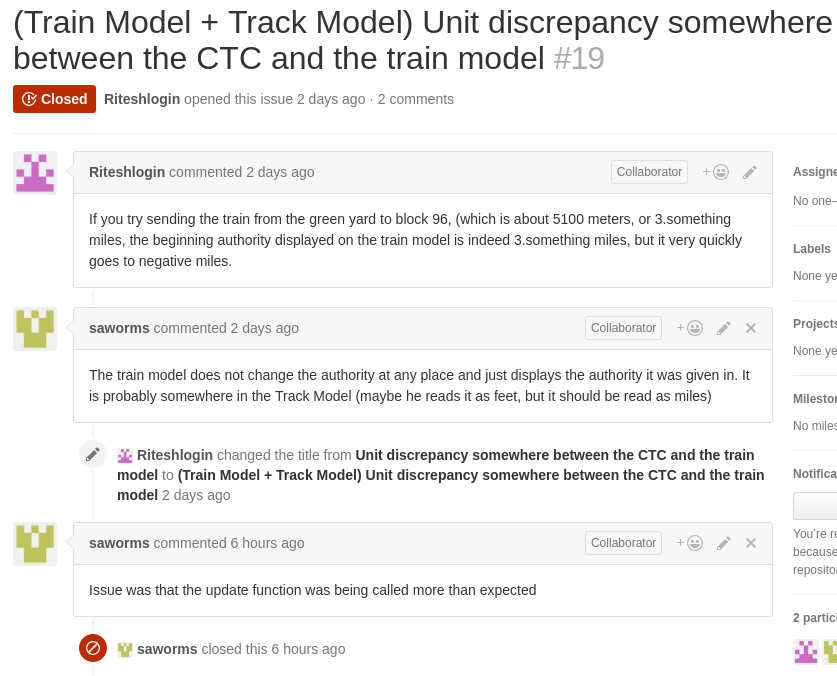
* 1. (CTC Office) - Not all destinations reachable



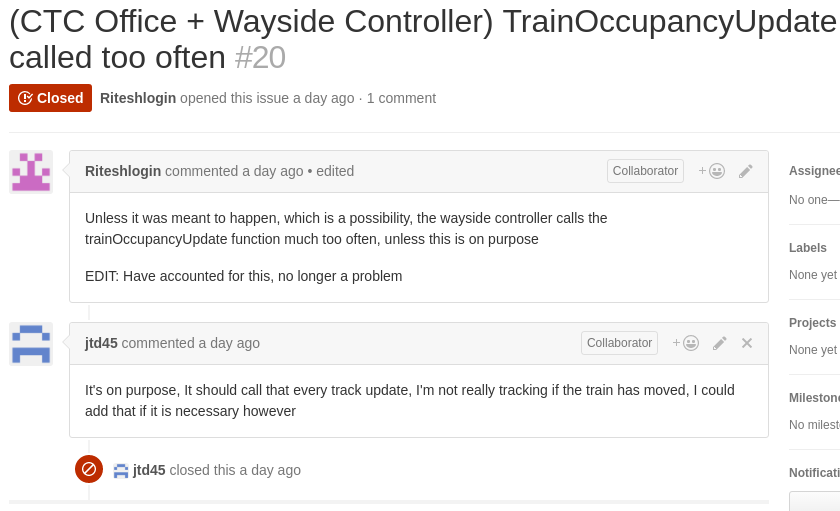
* 1. (Track Model) - Infinite Loop



* 1. (Train Controller) - Manual speed control not working
  2. (Train Model + Track Model) - Unit discrepancy somewhere between the CTC and train model)



* 1. (CTC Office + Wayside Controller) - TrainOccupancyUpdate called too often



# Bugs per Week

|  |  |
| --- | --- |
| Week | Bugs open |
| Week 12 | 0 |
| Week 13 | 1 |
| Week 14 | 16 |
| Week 15 | 0 |