**

Software Design Document – Track Controller

Version 1.0

March 22, 2012

PAAC Demonstration System

|  |  |  |
| --- | --- | --- |
| List Of Revisions | | |
| Date | Name | Description |
| 3/22/2012 | Sean Cardello | Initial creation of SDD. |

# Table of Contents

[List Of Revisions 2](#_Toc320193431)

[Table of Contents 3](#_Toc320193432)

[1 Introduction 5](#_Toc320193433)

[1.1 Purpose 5](#_Toc320193434)

[1.2 Scope 5](#_Toc320193435)

[1.3 Definitions and Abbreviations 5](#_Toc320193436)

[2 Reference 5](#_Toc320193437)

[3 Decomposition Description 5](#_Toc320193438)

[3.1 Module Decomposition 5](#_Toc320193439)

[3.1.1 Track Controller 5](#_Toc320193440)

[3.1.2 Switcher 5](#_Toc320193441)

[3.2 Current Process Decomposition 6](#_Toc320193442)

[3.2.1 Communication Process 6](#_Toc320193443)

[3.2.2 Controller Process 6](#_Toc320193444)

[3.2.3 Switch Process 6](#_Toc320193445)

[3.3 Data Decomposition 6](#_Toc320193446)

[3.3.1 Track Graph 6](#_Toc320193447)

[3.3.2 Track Status 6](#_Toc320193448)

[3.3.3 Authority 6](#_Toc320193449)

[3.3.4 Signal State 6](#_Toc320193450)

[3.3.5 Switch State 6](#_Toc320193451)

[4 Dependency Description 6](#_Toc320193452)

[5 Interface Description 7](#_Toc320193453)

[5.1 Module Interaction 7](#_Toc320193454)

[5.1.1 Normal Communication 7](#_Toc320193455)

[5.1.2 Close Track Block Sequence 7](#_Toc320193456)

[5.1.3 Failure Simulation 7](#_Toc320193457)

[5.2 Current Process Interface 7](#_Toc320193458)

[5.2.1 Communication Process 7](#_Toc320193459)

[5.2.2 Controller Process 7](#_Toc320193460)

[5.2.3 Switch Process 7](#_Toc320193461)

[6 Detailed Design 7](#_Toc320193462)

[6.1 Module Detailed Design 7](#_Toc320193463)

[6.1.1 Track Controller 7](#_Toc320193464)

[6.1.2 Switcher 10](#_Toc320193465)

[6.2 Data Detailed Design 11](#_Toc320193466)

[6.2.1 Track Graph 11](#_Toc320193467)

[6.2.2 Track Status 11](#_Toc320193468)

[6.2.3 Authority 12](#_Toc320193469)

[6.2.4 Signal State 12](#_Toc320193470)

[6.2.5 Switch State 12](#_Toc320193471)

# Introduction

## Purpose

The purpose of this document is to describe the design of the track controller component of the transit system simulation project. This document intention is to illustrate the overall functionality of the track controller.

## Scope

The scope of this document includes track controller use cases, class diagram, and sequence diagrams as it pertains to the transit system simulation. The functionality addressed in this design was laid out by the subsystem software requirements specification (see Reference 3).

## Definitions and Abbreviations

**Authority** – how far in distance the train is permitted to travel

**Block** – a section of a railway line

**Region** – the contiguous blocks of track between 2 switches

**CTC** – Centralized Traffic Control

**GUI** – graphical user interface

# Reference

1. IEEE-1016 Software Design Document
2. NSC-009 PAAC Bid Package
3. Bazinga Track Controller Subsystem Software Requirements Specification
4. Bazinga Transit System Software Design Document

# Decomposition Description

## Module Decomposition

### Track Controller

The track controller module is composed of 2 main data structures: a Track Graph and a Switcher. It will use these structures to correctly map trains to the specified route in a safety critical manner. It will decode suggestions sent by the CTC and determine whether those instructions are safe to execute by knowing the current status of the track determined track blocks.

### Switcher

The switcher is a component of the track controller that will use the Track Graph to handle track connectivity. It will only be concerned with its specified switch (1 per track controller/switcher) and the regions that it touches.

## Current Process Decomposition

### Communication Process

The Communication process will be responsible for all of the communication between the track controller and the track block/train and the CTC Office GUI. It will process the commands received by each and decode them as necessary for the Controller process.

### Controller Process

The Controller process will be constantly monitoring track status once authority is sent to the train. It will continue to set signals as appropriate and watch for any failure modes that could occur during operation. It will continuously check the train’s movement and ensure that it is abiding by the commands sent.

### Switch Process

The Switch process will be controlled by the switcher who is dependent upon the track controller process. It will be responsible for block switching and track graph updating. It will also monitor the switches for any failures and ensure safe operation.

## Data Decomposition

### Track Graph

The Track Graph is a graph data structure that will store track blocks in a way that connectivity can be determined between any two blocks. This will be useful in switching and closing of track.

### Track Status

The Track Status structure will be used to encapsulate the necessary properties of a track block such as: train presence, broken rail, circuit failure, power failure, signal state and switch state.

### Authority

The Authority structure will be used to encapsulate the data needed by the track controller and train sent by the CTC. These parameters include: speed limit, authority, and destination.

### Signal State

The Signal State enumeration will be used to keep the state of a particular track block’s signal lights. There are four signals in total: RED (stop), YELLOW (proceed with caution/slow down), GREEN (proceed at current speed/acceleration), SUPERGREEN (full speed ahead as fast as possible).

### Switch State

The Switch State enumeration will be used to keep the state of a particular switch. A switch will either be: THROWN (open or closed), or THROWING (in the process of changing).

# Dependency Description

The inter-module, inter-process and data dependencies are described in the UML class diagram below.

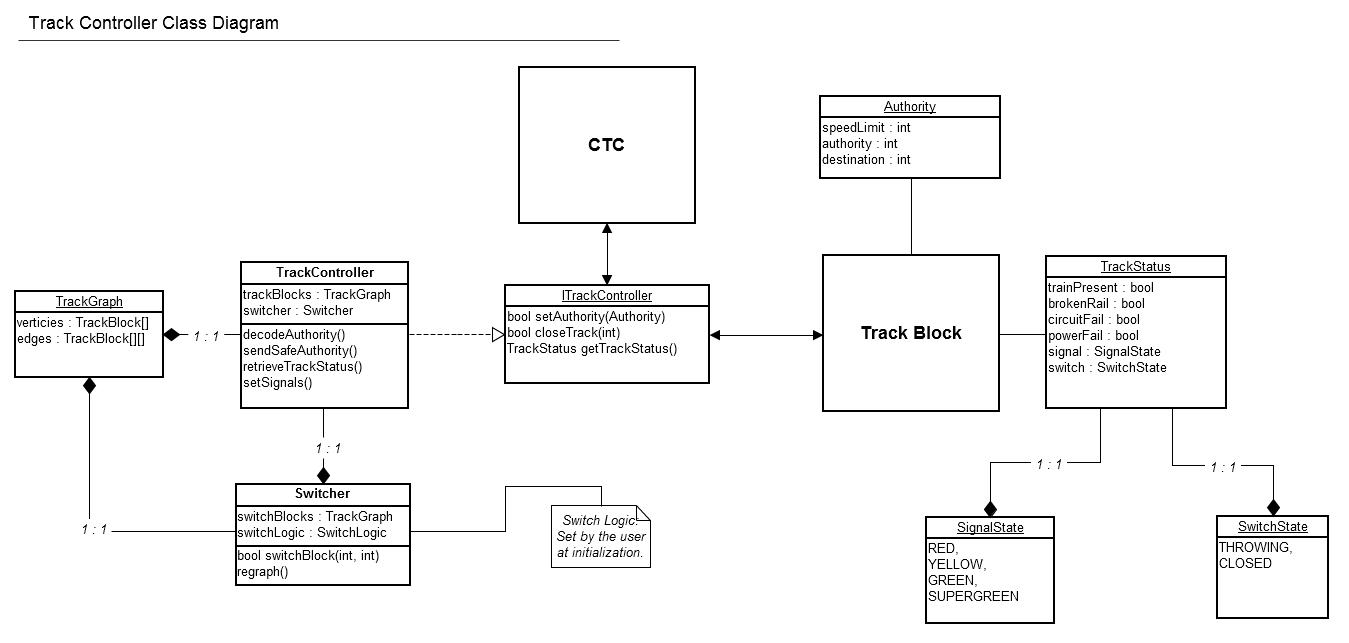


Figure - Class diagram

# Interface Description

## Module Communication

All track controller module communication will occur in the Communication process outlined above. It will interface with the CTC and track blocks as required for operation. It will also be responsible for monitoring the status of track blocks and passing any necessary information to the track controller for processing. See the sequence diagrams below for a detailed design.

# Detailed Design

## Module Detailed Design

### Track Controller

The Track Controller module is the main component within the Wayside Controller part of this system. It will be responsible for decoding and determining safe operating authority based on issued suggestions and current track status. The uses cases that the track controller is responsible for are illustrated below.

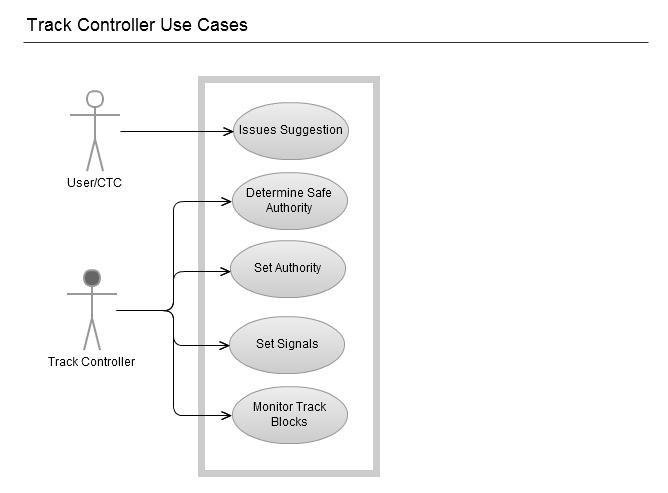


Figure - track controller use cases

The UML sequence diagram below shows the inter-module/process interfacing between the track controller and switcher under normal operating conditions.

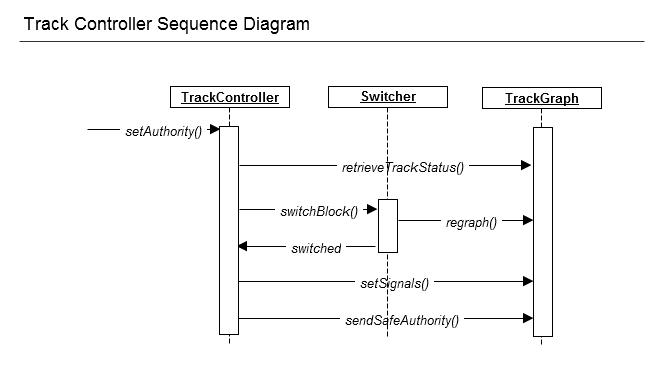


Figure 3 - normal sequence

The UML sequence diagram below shows the inter-module/process interfacing when a close block command is sent from the CTC.

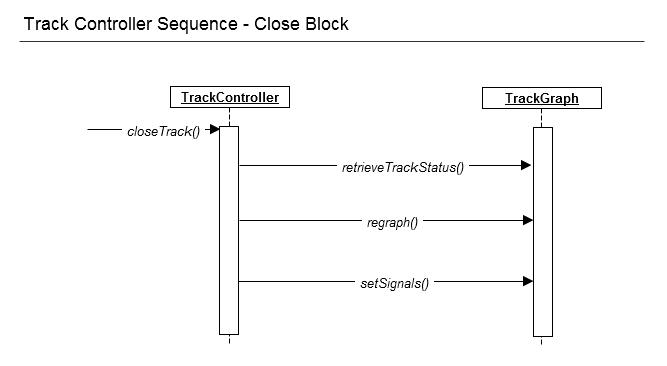


Figure - close block

Since this process is a simulation of a transit system, failure modes need to be “faked” from the CTC office in order to create them. Below is the UML sequence diagram of the interactions that would occur in such an event.

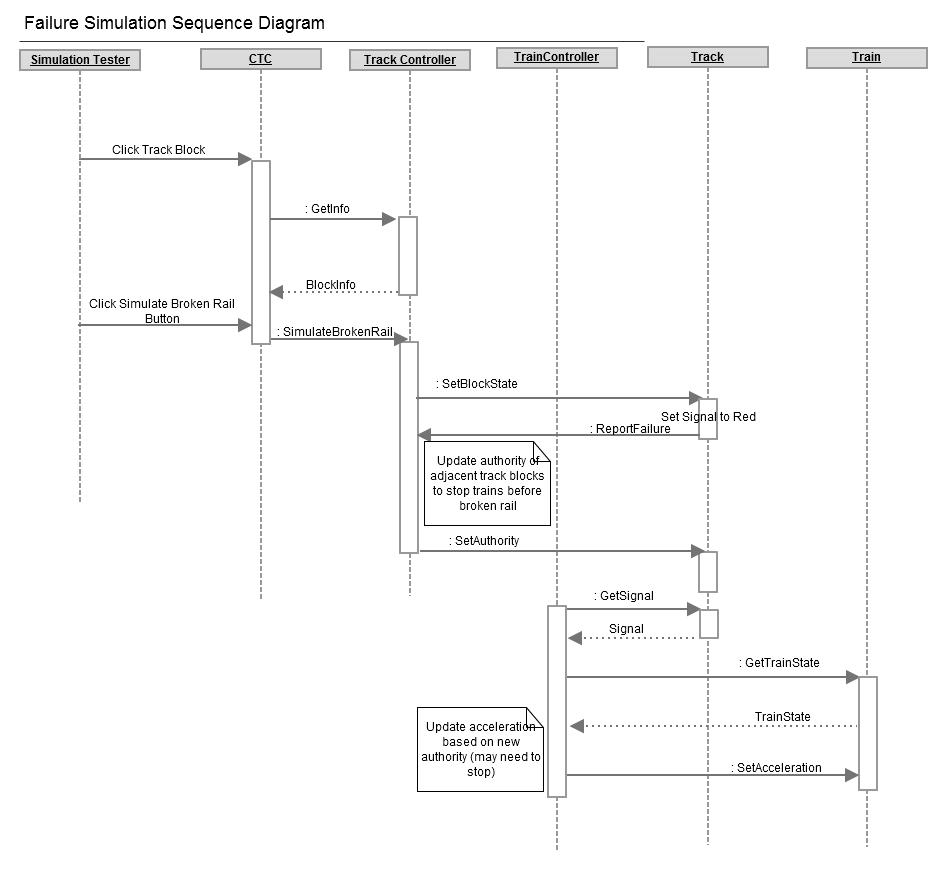


Figure 5 - Failure simulation illustration

### Switcher

The switcher is responsible for the switching and monitoring of the switch blocks that change track connectivity. This process is extremely important because a train cannot be switched to the same track as an oncoming train as this would be catastrophic. Thus, this component is safety critical in nature and will need to abide by the Boolean logic given to it by the CTC at initialization. The use case diagram for this component is shown below. The sequence diagram for normal operation can be seen in Figure 2 above.

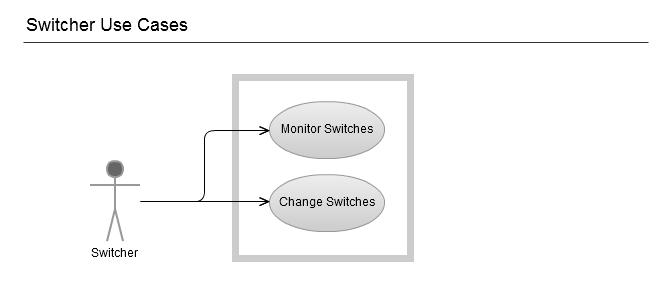


Figure - switcher use cases

## Data Detailed Design

### Track Graph

The Track Graph structure will be used as the class diagram below describes. It will hold an array of vertices which will correspond/map to an array of edges that it is connected to. A track block is only considered “connected” if and only if it is adjacent to the given vertex and in the direction of travel. This allows the switcher to simplify its functions by updating the graph as connectivity is changed. It also allows the track controller to easily determine which blocks are connected and thus where the trains are going to be at any particular time.

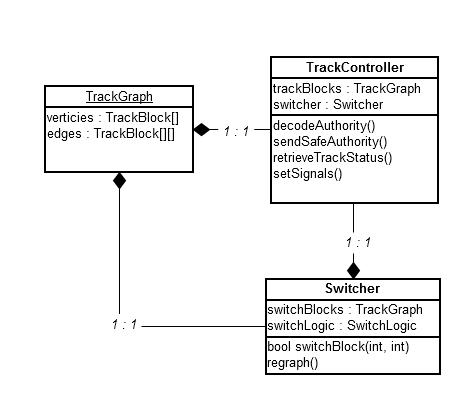


Figure - track graph data structure

### Track Status

The Track Status structure will be used to encapsulate all of the properties needed by the Track Controller to determine safe operation. Every track block will have an associated Track Status that will be updated by the Track Controller (Signal State/Failure modes), the Switcher (Switch State), and the Train Controller (presence).

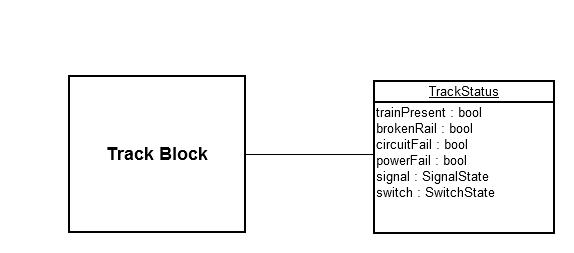


Figure - Track Status Data Structure

### Authority

The Authority structure is associated with a track block only in the sense of communication with the train controller. The properties within this arrangement are sent to the track controller by the CTC and then modified by it based of safety considerations. Once it is determined to be safe and accurate, it is passed to the track block for train controller “pickup”.

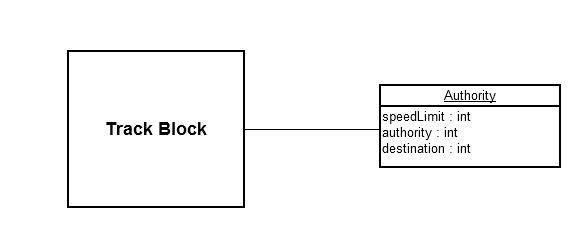


Figure - Authority data structure

### Signal State

The Signal State structure is an element of Track Status that will be updated by the Track Controller according to advised authority. It will also be useful to the CTC in displaying active and inactive track blocks for illustration.

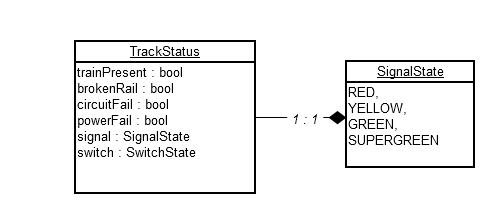


Figure - signal state data structure

### Switch State

The Switch State object shown below will be used to ensure transit safety. A train will not be allowed to cross a switch unless it is in the THROWN state which will be determined by the Switching process.

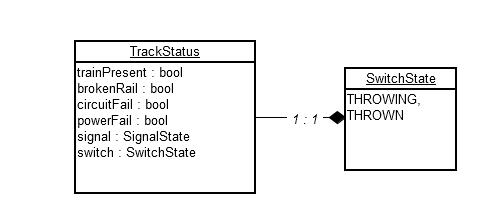


Figure - switch state data structure