Wonderfresh

PAAC North Shore Extension

Test Plan

Version 1.0

March 16, 2017

Rail Control Systems

Prepared for

COE1186 - Software Engineering

Instructor: Joseph A. Profeta III, Ph.D.

Spring 2017

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 3/15/17 | Version 1.0 | Rail Control Systems | First version of the Test Plan |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

This Test Plan document has been reviewed, accepted, and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Austin Bagnato | Train Controller |  |
|  | Kevin Carr | Track Model |  |
|  | Sarah Higbee | CTC Office |  |
|  | Angela Hoeltje | Moving Block Overlay |  |
|  | Camille Holland | Train Model |  |
|  | Kayla Walker | Communication Controller, Track Circuit and Train Antenna |  |

**Test Items**

1. Total System

Due to the loss of a group member testing the whole system after scrambling to redesign to six modules instead of seven will not show accurate operation of individual modules. The group will try to integrate around this new issue knowing that our final system will not meet the original requirements of the system.

1. CTC
   1. Major Methods
      1. Dispatching
      2. Sending speed/authority through Communication
      3. Open/closing block
      4. Flipping switch
      5. Changing mode
      6. Loading schedule
      7. Receiving instructions from MBO
      8. Updating displays
   2. Algorithmic Functionality
      1. Checking block occupancy and ensure one train per block
      2. Animating movement on map
      3. Calculating throughput
2. Communication
   1. Major Methods
      1. Sending and receiving data to and from CTC and Track Controller
      2. Track Circuit putting information on the block
      3. Track Circuit encoding the data
      4. Train Antenna decoding the encoded information
      5. Train Antenna sending vital information to Train Model
   2. Algorithmic Functionality
      1. Fixing the encoded message if there is one bit in err
      2. Testing if information received from modules is equal to NULL
3. Track Model
   1. Major Methods
      1. Importing Track
      2. Simulating Failures
      3. Carrying authority and set point speed between Wayside and track circuit.
      4. All methods contained in the public block to access block information
      5. Changing the switch states
   2. Algorithmic Functionality
      1. Simulating temperature.
      2. Organizing track model into functional data structure.
      3. Simulating people at train stations.
4. Train Model
   1. Major Methods
      1. Activating emergency brake
      2. Syncing train attributes with Train Controller
   2. Algorithmic Functionality
      1. Approaching target temperature
      2. Approaching target speed
5. Train Controller
   1. Major Methods
      1. Setting a speed
      2. Receive beacon information
      3. Turning on and off lights
      4. Opening and closing both left and right doors
      5. Stopping for stations
      6. Calculate power commands
   2. Algorithmic Functionality
      1. Ensure safe speed
      2. Ensure stopped before opening doors
      3. Ensure correct side doors are open
6. Moving Block Overlay
   1. Major Methods
      1. Communication with CTC
      2. Communication with Communications module
      3. Receiving valid input from Scheduler
      4. Creating drivers schedule
      5. Creating trains schedules
   2. Algorithmic Functionality
      1. Calculate safe stopping distance

**Software Risk Issues**

1. Total System

Integration is the biggest issue here. As previously stated lacking an entire module integral to our original design increases the likely hood of this issue.

1. CTC
2. Communication
   1. Train Antenna must fix encoded data if there is one bit in err; else, information will be wrongly decoded
   2. Train Antenna must send data to Train Model; else, vital error occurs
3. Track Model
   1. Correct Excel data file must be chosen and in the correct format to run properly.
   2. Blocks must contain information regarding next block, or train model will experience issues.
4. Train Model
   1. Brake failure must always activate emergency brake, regardless of command from Train Controller, until brakes are fixed.
5. Train Controller
   1. Safety critical features must be implemented correctly (doors, emergency brake, and speed regulation)
6. Moving Block Overlay
   1. Failure to communicate with CTC, or communications will result in the system leaving moving block overlay mode until communications are restored.
   2. Must send signals for trains to break if they are too close, less than calculated safe stopping distance.

**Features to be Tested**

1. CTC
2. Communication
   1. Sending/Receiving data from CTC and Track Controller
   2. Indicator lights on each subsection of the Communication Controller UI
   3. Putting information on the block
   4. Encoding/Decoding data between the Track Circuit and Train Antenna
   5. Train Antenna sends vital information to Train Model
3. Track Model
   1. Importing track model.
   2. Testing temperature and track heater algorithm.
   3. Check that CTC is notified of block failures.
4. Train Model
   1. Emergency brake activation
   2. Acceleration of train in response to power command
   3. Acceleration of train in response to brakes
5. Train Controller
   1. Setting a speed
   2. Setting beacon info
   3. Opening/Closing doors
   4. Calculating power commands
   5. Stopping train with emergency brake
6. Moving Block Overlay
   1. Communication with CTC & Communications module
   2. Receiving valid input from Scheduler
   3. Creating drivers schedule
   4. Creating trains schedules
   5. Calculate safe stopping distance

**Approach**

The system will be tested using both junit testing and manually running tests and viewing their outputs.

**Item Pass/Fail Criteria**

A module must pass all tests or is considered failing.

**Specific Test Cases**

1. CTC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | **Requirements** | **Method** | **Inputs** | **Outputs** |
| Sarah |  |  |  |  |

1. Communication Controller

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | **Requirements** | **Method** | **Inputs** | **Outputs** |
| Kayla | Receiving information from CTC and Track Controller | incomingCTCOutingTrackCtrl(), incomingTrackCtrlOutgoingCTC() | Valid incoming speed, authority, and train ID from CTC | Blue indicator light flashes on "Incoming CTC" section on the UI |
|  |  |  | Invalid incoming speed, authority, and/or train ID from CTC | Red and blue indicator lights flash on "Incoming CTC" section on the UI |
|  |  |  | Valid incoming block ID, lights status, occupancy status, railway crossing status, and switch status from Track Controller | Blue indicator light flashes on "Incoming Track Controller" section on the UI |
|  |  |  | Invalid incoming block ID, lights status, occupancy status, railway crossing status, and/or switch status from Track Controller | Red and blue indicator lights flash on "Incoming Track Controller" section on the UI |
|  | Sending information to CTC and Track Controller | sendInfoToCTC(), sendInfoToTrackCtrl() | Valid block ID, lights status, occupancy status, railway crossing status, and switch status to CTC | Either the real CTC module or the mock CTC module verifies information was received |
|  |  |  | Invalid block ID, lights status, occupancy status, railway crossing status, and/or switch status to CTC | Red indicator light flashes on "Outgoing CTC" section on the UI |
|  |  |  | Valid speed, authority, and block ID to Track Controller for both lines | Either the real Track Controller module or the mock Track Controller module verifies information was received |
|  |  |  | Valid speed, authority, and block ID to Track Controller for one line | Assume data is the same for both lines; either the real Track Controller module or the mock Track Controller module verifies information was received |
|  |  |  | Invalid speed, authority, and/or block ID to Track Controller | Red indicator light flashes on "Outgoing Track Controller" section on the UI |
|  | Track Circuit receives data, encodes it, and puts it on the current block | incomingTrackModelOutgoingTrackCircuit(), encodeData(), updateBlockInformation() | Valid speed, authority, and train ID received from Track Model | Blue indicator light flashes on "Track Circuit" section on the UI |
|  |  |  | Invalid speed, authority, and/or train ID received from Track Model | Blue and red indicator lights flash on "Track Circuit" section on the UI |
|  |  |  | Valid data was received and is put on current block | Block is updated with new information |
|  |  |  | Invalid data was received and was not put on current block | Block information remains unchanged |
|  | Train Antenna decodes encoded data and sends it to Train Model | decodeData(), sendInfoToTrainModel() | Encoded data was received with 0 bits in err | Blue indicator light flashes on "Train Antenna" section on the UI; data is decoded and sent to Train Model |
|  |  |  | Encoded data was received with one bit in err | Blue indicator light flashes on "Train Antenna" section on the UI; encoded data is fixed, so there are no bits in err; data is then decoded and sent to Train Model |
|  |  |  | Encoded data was received with more than one bits in err | Blue indicator light flashes on "Train Antenna" section on the UI; data is decoded and sent to Train Model |
|  |  |  | Encoded data is equal to NULL | Blue and red indicator lights flash on "Train Antenna" section on the UI; vital error occurs -> let Train Model know vital error occurred |

1. Track Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | **Requirements** | **Method** | **Inputs** | **Outputs** |
| Kevin | Importing Track | ExcelToJavaGraph() | Properly formatted Excel Data File | Track Model (Lines, Sections, Blocks, Length, Grade, Speed Limit, Infrastructure, Switch Block, Arrow Direction ) |
|  |  |  | Improperly formatted Excel Data File | Error warning to message panel that the file isn't formatted correctly. |
|  |  |  | Not an Excel File | Error warning to message panel that the file isn't of the right type. |
| Kevin | Cause a Failure | SetFailure(Block brokeBlock, boolean break, boolean power, boolean circuit) | Check box of "Break Line", hit Update. | Box stays checked and updates in system. Signal sent to Wayside controller about block failure. |
|  |  |  | Check box of "Power Failure" , hit Update. | Box stays checked and updates in system. Signal sent to Wayside controller about block failure. |
|  |  |  | Check box of "Circuit Failure", hit Update. | Box stays checked and updates in system. Signal sent to Wayside controller about block failure. |
|  |  |  | Check multiple boxes of the above failure modes and hit Update. | Boxes stay checked and update in system. Signal sent to Wayside controller about block failure. |
| Kevin | Change Temperature | SetTemperature(int temp) | Set temperature box between 0 and 32°F. | Heater should turn on at any given block. |
|  |  |  | Set temperature box between 33 and 39°F when previous temperature was 32°F or below. | Heater should remain turned on at any given block. |
|  |  |  | Set temperature box between 33°F and 100°F when previous temperature was above 32°F. | Heater should either turn off or remain off. |

1. Train Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | **Requirements** | **Method** | **Inputs** | **Outputs** |
| Camille | Emergency Brake Activation | setEBrake(1) | Emergency brake set to on | Train stops, without exceeding braking rate |
|  | Acceleration in response to power command | setSpeed() | Set point speed is greater than current speed | Train asymptotically approaches set point speed, whilst not exceeding acceleration limit |
|  |  |  | Set point speed is equal to current speed | NA |
|  |  |  | Set point speed is lesser than current speed | Service brakes are activated until speed is lesser than set point speed, then train asymptotically approaches set point speed |
|  | Acceleration in response to brakes | setEBrake(), setServiceBrake(), setSpeed | Service brake activated | Train decelerates steadily at correct rate |
|  |  |  | Emergency brake activated | Train decelerates steadily at correct rate |

1. Train Controller

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | **Requirements** | **Method** | **Inputs** | **Outputs** |
| Austin | Setting a speed | setSpeedAndAuthority(), setSetSpeed() | Safe speed | The speed is successfully set to the input in the TrainController and PowerCalculator classes |
|  |  |  | Speed exceeding limit | Display notification that speed was invalid, speed remains the same |
|  |  |  | Invalid speed | Display notification that speed was invalid, speed remains the same |
|  | Setting beacon info | setBeaconInfo(),  setLights() | Valid beacon information indicating entering tunnel at time between 6PM and 8AM | Beacon information is set and lights remain on |
|  |  |  | Valid beacon information indicating entering tunnel at time between 8AM and 6PM | Beacon information is set and lights are set to on |
|  |  |  | Valid beacon information indicating leaving tunnel at time between 6PM and 8AM | Beacon information is set and lights remain on |
|  |  |  | Valid beacon information indicating leaving tunnel at time between 8AM and 6PM | Beacon information is set and lights are set to off |
|  |  |  | Invalid beacon information | Beacon information is not set |
|  | Opening/closing left doors | setLeftDoors(),  setBeaconInfo() | Speed is 0, and beacon information indicates station to left | Left doors are set to open or closed |
|  |  |  | Speed is 0, and beacon information indicates station to right | Left doors remain closed |
|  |  |  | Speed is greater than 0, and beacon information indicates station to left | Left doors remain closed |
|  |  |  | Speed is greater than 0, and beacon information indicates station to right | Left doors remain closed |
|  | Opening/closing right doors | setRightDoors(), setBeaconInfo() | Speed is 0, and beacon information indicates station to right | Right doors are set to open or closed |
|  |  |  | Speed is 0, and beacon information indicates station to left | Right doors remain closed |
|  |  |  | Speed is greater than 0, and beacon information indicates station to right | Right doors remain closed |
|  |  |  | Speed is greater than 0, and beacon information indicates station to left | Right doors remain closed |
|  | Calculating power commands | The power calculator is a thread that calculates the power within its run() method | Valid speed | Power is calculated and sent to train |
|  |  |  | Invalid speed | Continues at previous speed |
|  | Stopping train with emergency brake | engageEmergencyBrake() |  | Train comes to complete stop |

1. Moving Block Overlay

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | **Requirements** | **Method** | **Inputs** | **Outputs** |
| Angela | Communication with CTC and Communications module | setDispatchedTrain(), setUpdateSpeedAuthority() | Valid trainID, speed, and authority. | Message sent with correct information |
|  |  |  | Any input as a String | Error message & train info not sent. |
|  |  |  | Any input as a negative integer | Error message & train info not sent. |
|  |  |  | TrainID as a decimal number | Error message & train info not sent. |
|  | Receiving valid input from scheduler | getData() | Number of Drivers as a valid number (1-50) | Accepted |
|  |  |  | Any input as a String | Error message, and scheduler input erased. |
|  |  |  | Any input as a negative number | Error message, and scheduler input erased |
|  |  |  | Any input as a decimal | Error message, and scheduler input erased |
|  |  |  | Number of Drivers entered as integer less than or equal to 0, or greater than 50 | Error message, and scheduler input erased |
|  | Creating drivers schedule | getDriverSchedule() | Number of drivers as integer between 1 and 50 | Driver's schedule shows correct number of driver IDs all with five 8.5 hour shifts including meal break after 4 hours |
|  | Creating trains schedule | GetTrainSchedule() | String for line color, and percent of drivers to be used. | Train schedule shows correct time around line and utilizes correct percentage of drivers.  Schedule takes into consideration lunch breaks taken by drivers at yard. |
|  | Calculate safe stopping distance | GetBrakeDistance() | Braking constant, maxSpeed | Prints calculation to terminal, line by line for inspection to check math, and process. |