Low-Comotovation Rail System Configuration Management

Document Control

|  |  |
| --- | --- |
| Team | Low-Comotovation |
| Document Status | Version 1 |
| Date of Issue | 4/20/17 |

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Doc. Version | Author | Date | Description / Change |
| 1 | Michael Ghaben | 4/20/17 | Document Creation |
|  |  |  |  |

Distribution List

|  |  |
| --- | --- |
| Name | Role |
| Demetri Khoury | Train Model |
| Max Reno | Track Controller |
| Andrew Lendacky | Train Controller |
| Michael Ghaben | Track Model |
| Christen Reinbeck | CTC Office |
| Zach Scheider | MBO |

1. **Version Control System**

The version control system utilized for this project consisted of a publicly available GitHub. The associated Travis-CI server was integrated to ensure consistent builds and regression testing.

**Repository**

[**https://github.com/michaelghaben/ece1186**](https://github.com/michaelghaben/ece1186)

**Commit History**

Individual branch commit history is available via GitHub.

**Issues**

GitHub issues as well as Javadoc-style comments were used for tracking bugs. To view the GitHub issues, visit: <https://github.com/michaelghaben/ece1186/issues>

To view the Javadocs-style generated bugs and todo lists, visit:

**Static Analysis**

To better understand the software evolution and track development, we utilized Doxygen.

1. **Continuous Integration**

Detailed build history and test status of each individual branch pushed to the remote branches under the Maven build system as well as the accompanying regression test results are available at:

<https://travis-ci.org/michaelghaben/ece1186>

Travis is free to open source projects, with commercial licensing available.

To view branch-wise build history, click on the “branches” tab. To view chronological build history, click on the “build history” tab. These build results are reflected on GitHub.

1. **Building and Development Environment**

The build system consists of Apache Maven. Unless explicitly overridden, unit tests are run prior to build completion. To build using Maven, first visit the associated website (<https://maven.apache.org)> and install for the desired operating system. From a terminal supporting Maven, run “mvn package”. This will run unit tests and checkstyles prior to building the jar. The build parameters used for Maven may be found and configured in the accompanying pom.xml file in the root directory of the project. Note that Apache Maven is provided under a permissive commercial license, allowing for the continued integration of the build system in commercial projects.

Alternatively, the preferred build system consists of running the run.sh shell script. This is supported on POSIX compliant systems. On windows, this may be accomplished through tools such as Cygwin or MinGW. This bash script addresses a number of issues encountered through the development process such as the removal of operating system specific files and prior build files. This was found highly beneficial, as it ensured that Java could only reference the most recent class files, preventing potentially build-breaking old build information from reaching the programmer for debugging. At the end of the bash script, it will then automatically open the most recently built jar file.