Maintainability Plan

ECE 492 - Spring 2015

Abstract

This document outlines the language and software choices of VSCADA and the plans of maintaining the software.

Revision 1.1
Sam Cesario

Table of Contents

Summary

This document outlines how the VSCADA team plans to address the general maintainability requirements. This maintenance plan is software-specific. The software must be maintainable over the 5-year life of the LFEV system. The plan answers the following questions listed in the Statement of Work (SOW).

1. What is the design of the system API and how will this design support ongoing reliable operation, maintenance and expansion?

Application Programming Interface or API is defined as a set of requirements that govern how one application can talk to another. The VSCADA team's system API will be designed to meet requirement S021 located in the VSCADA Preliminary Design Report. Requirement S021 states, "Good, open-source API and SDK required with all necessary documentation. VSCADA software shall be a suite of applications built to a unified API with common data formats, protocols, [and] look and feel." When first designing the software for our Lafayette Electric Vehicle we plan to write a skeleton of our low level API. We will analyze the requirements listed in the PDR and roll them into our API. Coding done between teams will rely heavily on this agreed upon API so that no time is wasted writing incompatible code. The API will be extensive so as to support all required operations.

2. How is system configuration maintained? Will the system auto detect hardware configuration changes or will configuration maintenance be required? If the latter, what is the consequence of misconfiguration?

System configuration will be contained in configuration files (text files) that will be easily editable by both the system developers and the end-user. The system will use another configuration file to keep track of hardware that is connected. If sensors which are in the configuration file are detected as missing at startup, the user will be notified and given the choice of shutting down the system or running without drive mode. If any sensors are misconfigured, errors will be logged for later viewing and the user notified with a message.

3. What tool chain will be used? Is the tool suite up-to-date and actively supported? Is the tool suite mature enough to have stable functionality? Evidence must be provided to support assertions.

Popular, open-source, widely supported software will be chosen for the project along with a language with standard libraries.

The embedded linux computer will have a 32-bit linux operating system installed. This system, most likely ubuntu, will be widely supported and have the features of extendability and ease to configure.

Systemd will be used as a system management platform for daemon creation as it has most major linux distributions support.[1]

The GCC compiler system created by the GNU Project, will be used as it supports programming languages such as C++ and Java, and the GNU Project regularly updates and maintains the compiler.[2]

[1] http://www.freedesktop.org/wiki/Software/systemd/
[2] https://gcc.gnu.org/releases.html

4. What third party software will be incorporated into the system? How will this be maintained, upgraded, or patched during the life of the system.

SocketCAN, an open source CAN drivers along with a network stack was developed by Volkswagen Research. SocketCAN is included in the Linux Kernel 2.6.25 and up and is well documented on the linux kernel website.[3]

RRDtool is a round robin database tool that handles time series data in a circular buffer. RRDtool is sponsored by various users and the tool has been declared as an industry standard for time-series data logging and graphing.[4]

QT Creator is a cross platform and user interface framework that will be used to create the user interfaces for the VSCADA system. QT partners with top names in the technology industry such as ARM, and offer extensive support.[5]

The distributed revision control tool for software, Mercurial will be used to maintain order among the software as it is implemented on many web based hosting services.[6]

The web based hosting service BitBucket will be used to manage the mercurial repositories making them available anywhere.[7] Bitbucket is owned by the enterprise software company Atlassian with no sign of dropping support.

The programming language Python will be implemented, and version 3 will be used as to keep the software as current as possible.[8] Also the programming languages C and C++ will be used as they both are widely supported. The Structured Query Language (SQL) will be used for a relational database management system, as SQL has a wide range of support and implementation.

Copies of the software used in the system, with versions the same as those actually used in the system, will be archived on the project website for possible future use.

5. How are requirements in GPR007 met?

Software Maintainability

- Software must be developed according to an explicit Software Maintainability Plan.
  - This Maintainability Plan will be updated on 2/13/2015 and will be used as the guideline for coding. The Maintainability Plan will be finalized on 2/25/2015.
- All software source code must be maintained under configuration control. Release snapshots must be archived on the project website.
  - This requirement will be met later during software development.
- The system must start from cold power-up and boot to full operational status without

[3] https://www.kernel.org/doc/Documentation/networking/can.txt
[4] http://oss.oetiker.ch/rrdtool/
[5] http://www.qt.io/about-us/
[6] http://mercurial.selenic.com/
[7] https://bitbucket.org/
[8] https://www.python.org/about/

requiring user interaction beyond enabling power and safety procedures.
    - This will be achieved per requirement S001.
- Any PC software must be packaged for installation with a SETUP.EXE, RPM, “make install” or equivalent installer allowing it to be installed easily on any compatible computer.
    - This will be achieved as an extension to requirement S021.
- Configuration parameters, calibration factors, preferences, and options shall not be hard-coded within the software source code. It shall be possible to alter these various factors without recompiling software or physically disassembling hardware. Altered configuration parameters must be persistent through power cycling and reboots. The system must have a function to initialize itself with sane (factory default) configuration content if requested.
    - This will be achieved per requirement S016 and S019.
- All data and configuration files must be in a generally supported format (e.g. XML) or the format required by a mature and well supported application (e.g. MySQL database files, Berkeley db, etc...). The use of custom formatted ASCII or binary files for configuration or data storage is not permitted. Files shall be accessible either through removable media or network file transfer or both.
    - This will be achieved per requirement S016 and through use of the Interface Control Document (ICD).
- The use of removable media (thumb drives, flash media cards) is permitted for configuration parameters, offline storage, access, and backup. If removable storage is used for configuration, the system must have the inherent capability to operate without media, and to initialize blank media with sane configuration content.
    - This will be achieved per requirement S016.
- Enumerated devices, such as USB, must be automatically discovered by the system and assigned correct port designations such that the system operates correctly after re-enumeration without any interaction or re-programming by the user. Port designations may not be “hard coded” in the software or firmware.
    - This will be achieved per requirement S017.
- Any cell phone software must be packaged and available from an online “App store” for easy installation on any compatible phone without requiring special alterations of the phone such as SDK installation or jailbreaking.
    - A mobile app will be considered after week 9.

Hardware Maintainability

- Maintainability requirements for hardware must be demonstrated in the ATP both by analysis and by Inspection. The use of MIL-HDBK-472 (N1) and MIL-STD-470B, ISO/IEC 25000:2005, or other equivalent techniques are encouraged for the analysis.
    - This will be achieved as suggested in the Requirement and Test matrix.
- In the hardware maintainability analysis you should assume a stock of recommended spare parts. The list of these spare parts should be included in the ATP. The Users Manual should include a section giving simple troubleshooting procedures. The

Maintenance Manual should have more elaborate diagnosis and troubleshooting resources.
    - The analysis of spare parts will be included in the final version of the ATP after extensive research on different embedded and communication systems. The budget is included in the PDR. The Troubleshooting and Maintenance Manual will be included in the User Manual.
- Should there be a failure, the system wide Mean Time To Repair (MTTR) must be less than 1 week over the system lifetime. MTTR applies to both hardware and software.
    - The recovery methods will be tested per T002. The actual testing methods and the time required will be analyzed in the CDR.
- In addition, a maintainability inspection shall be conducted during ATP where a novice using procedures included in the User Manual demonstrates the diagnosis and repair of a likely failure, and an expert using resources included in the Maintenance Manual demonstrates the diagnosis and repair of an UN-likely failure.
    - This will be further developed in the User Manual.