Software Requirements Specification

Rocket View 3000 v0.0.1

March 19, 2017

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to document the requirements for the RocketView 3000 (RV3K) v0.0.1 software system. This document is a reference for software scope and development. It will explain RV3K's purpose, features, interfaces and system constraints, as well as the scope of use.

Functional requirements were elicited during conversations between February 7th, 2017, and March 19, 2017 with Jamey Sharp and Andrew Greenberg at Portland State University. [1]

This document is (very loosely) based on IEEE SA - 29148-2011 Software Requirements Specification (SRS) guidelines [2].

1.2 Scope

RV3K v0.0.1 is a PSAS rocket telemetry display module that allows local and remote users to monitor PSAS vehicle flights using a networked computer that supports modern web-browsers (2017). The goal is to create data visualization features that will allow users to better understand what is presented. This includes, but is not limited to, 3D maps and video.

1.3 Product overview

1.3.1 System purpose

The three main purposes of RV3K are:

- State visualization and state transparency
  - To display internal vehicle and mission state information
  - To display external vehicle and mission state information
- Publicity

  To display telemetry data and mission state to viewers who are no domain experts
- Logistics
  - To coordinate recovery
    - To locate individual ground crew
    - To locate rocket parts for recovery

1.3.2 Product perspective

- RV3K is a cross-platform, software intensive system with a browser-based user interface
- The RV3K user interface is composed of web browser modules and widgets, server modules, and internetworking capabilities. As such, RV3K requires hardware that supports modern browsers (see §2.1.2).

1.3.3 Objectives and success criteria

- System can be invoked cross-platform via command
- RV3K UI can be invoked from the web browsers listed in §2.1.2
- RV3K correctly performs calculations on incoming data
- Upon failure, RV3K will enforce halting over displaying incorrect data
- RV3K displays telemetry information provided in the data stream
- RV3K displays GPS coordinates of the recovery crew

1.3.4 Limitations

RV3K, nor its creators shall be held responsible for death or injury resulting directly or indirectly from its use. Users of RV3K assume full responsibility for accidents, damages, injury, death, expenses, and/or humiliation incurred while using RV3K software system. This limitation is covered by the Capstone contract statement: "No mission critical or phase based projects"
`http://wiki.cs.pdx.edu/capstone/sponsors.html`

Scientific accuracy The accuracy of calculations for vehicle location, attitude, trajectory, and any other internal and external state are limited by sensor states, signal strength, connectivity, and other factors of complexity. RV3K shall provide approximate telemetry data about these states, but it shall not guarantee scientific accuracy or precision.

1.4 Definitions

cross-platform: An application that runs on Linux, macOS and Windows operating systems (2017).

1.4.1 Abbreviations used

RV3K	Rocket View 3000 v0.0.1 Telemetry Viewer
APRS	Automatic Packet Reporting System
PSAS	Portland State Aerospace Society
UI	User Interface
URL	Uniform Resource Locator

References

[1] Sharp, J.; Greenberg, A. RV3K Elicitation. 2017.

[2] 29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering. 2012; `https://standards.ieee.org/findstds/standard/29148-2011.html`.

2 Specific requirements

2.1 External interfaces

2.1.1 Operating systems

The system shall support the latest versions of Linux, macOS and Windows as of 2017.

2.1.2 Web browsers

- The system shall fully support Chromium and Firefox.
- The system should provide reasonable support for mobile browsers.
- The system should provide reasonable support for Internet explorer.

2.1.3 Incoming data streams

- Ground Video
- Recovery Team APRS Packets
- Rocket Telemetry
- Rocket Video

2.1.4 Outgoing data streams

- Telemetry Interface (see §2.2.1)

2.1.5 Network connections

- The system shall support receiving launch data streams over provided local networks (at launch site), and over the internet.

2.2 Functional Requirements

2.2.1 System must serve the user interface as a web page

- The web UI must be accessible on localhost and from the network
- The UI must include the following elements
  - Earth Frame View display
  - Vehicle Attitude (local tangent) display
  - Telemetry data
  - Video Views
    - Video stream sources TBD

2.2.2 System shall process and display video streams

- System shall define a video format
- System shall take as input video streams of the defined format
- System shall output the video in the UI
- Codec: TBD
- Source: TBD

2.2.3 System must interpret the telemetry data

- RocketView 3000 must interpret data from all relevant telemetry feeds.
- The system must output the interpreted data in a meaningful form for the UI described in §3.2.1
  - data interpretation shall describe internal and external states of the vehicle, before, during, and after flight.
  - Some data classifications are described in `https://github.com/psas/psas_packet`, but these classifations are likely to change.
- Packet rate details TBD
- Packet contents TBD

2.2.4 System must perform calculations on telemetry data

- The system shall accept Vehicle Attitude and Vehicle Trajectory data.
- The system shall translate Vehicle Attitude and Vehicle Trajectory data into object or data structure representations that will be used by the UI.

2.2.5 System shall interpret APRS data

- RV3K shall receive and interpret APRS data from recovery teams.
- RV3K shall output this data as input for the UI described in §2.2.1.

2.2.6 System should interpret rocket location

- RV3K should isolate GPS data from the incoming data stream or extrapolate location from otherwise available data.
- The system should output the interpreted data as input for the UI described in §2.2.1
- This functionality should support logistics as covered in §1.3.1

2.3 Extra-functional requirements

2.3.1 Performance requirements

- System shall perform real time operations within the constraints of the host hardware and network system.
- Imported libraries shall not hinder performance beyond a negligible amount
- Graphics processing shall not hinder performance beyond a reasonable amount
- Target latency should be near 10 milliseconds

Initial: ______

2.3.2 Reliability requirements

- System shall provide proper handling of packet loss and/or loss of connectivity
- System should provide meaningful indicators to the user when there is loss of connectivity and/or packet loss
- System shall have methods to ensure reasonably quick restart after crashing
  - After such a crash, system shall restart to the last good state

Initial: ______

2.3.3 Security

- Since RV3K is a one-way, standalone system, security is up to the discretion of users, and the parties who stream content to public or private networks.
- RV3K shall be easily modifiable in such a way that it could support end-to-end encryption, or other security enhancements, but this will not be a feature of version 0.0.1.

Initial: ______

2.3.4 Usability requirements

- The RV3K UI should be intuitive and easy to understand. Every pixel should have a purpose.
- User controls and settings should function responsively and be clearly labeled.
- User interface documentation should be easily invoked and/or accessible to the user.

Initial: ______

Accessibility

- Accessibility features will not be required, however, UI design should consider visual coherence, and use alt-tags whenever possible.
- Color schemes shall not be confusing to persons with color deficient vision.

Initial: ______

2.3.5 Maintainability

Code shall be well commented, well structured, and easy for an undergrad CS student to work on and understand.

- No magic: code shall maintain readability by avoiding convoluted or overly-clever blocks
- Code should follow best practices for whichever languages and frameworks are used
- Code should avoid obscure and/or overly complicated languages and frameworks

Initial: ______

2.3.6 Licensing requirements

Licensing requirements are covered in the license agreement document and copyright notice at the beginning of this document.

Initial: ______

3 Verification and validation requirements

3.1 Unit testing

The unit tests shall:

- Ensure that the web UI is accessible on localhost
- Check that the Earth Frame View display is functioning properly
- Check that the Vehicle Attitude display is functioning properly
- Check that the Video Streams are functioning properly
- Ensure that the Telemetry data isn't being misinterpreted
- Check that the translated Vehicle Attitude and Vehicle Trajectory data is correct
- Ensure that the APRS data isn't being misinterpreted
- Provide meaningful messages for each failure mode

3.2 Continuous integration

The development process for RV3K will use a GitHub repository together with a continuous integration tool (such as TravisCI).

3.3 Replay testing

The validation process for RV3K will use recorded and/or simulated data to improve the relevance of test results.

3.4 Runtime error handling & assertions

- RV3K will employ assertions and/or exception handling to manage error states during run-time.
- In the event of an critical error, RV3K shall notify the user of the error and prefer halting rather than allow continued operation that may present erroneous or incorrect data.
- The system will restart to its most recent state after a crash occurs.

3.5 Logging and instrumentation

RV3K will maintain plain text logs to facilitate debugging and diagnostic analysis.

4 Use cases

4.1 Use Case: Spectators

4.1.1 Primary Actor: Spectator

Flow of Events:

1. PSAS crew members start RV3K on a server, either local or remote
2. User opens their browser and navigates to the URL of the telemetry viewer, given to them by PSAS crew members.
3. User monitors and enjoys the PSAS launch via RV3K either locally at the launch site, or anywhere on the globe if they can get network access to the RV3K telemetry server.

4.2 Use Case: Flight Director and PSAS Crew

4.2.1 Primary Actor: Flight Director

Flow of Events:

1. PSAS crew members configure RV3K for launch
2. PSAS crew tests configuration and makes necessary modifications
3. PSAS crew pushes configuration to repository
4. PSAS crew invokes RV3K and opens it in browser tab(s)
5. PSAS crew members distribute URL of RV3K to spectators, both local and remote
6. PSAS Flight Director monitors launch flight and recovery crew locations
7. RV3K receives GPS coordinates from the Recovery Crews using APRS and updates the map with their locations throughout the launch
8. PSAS Flight Director changes view of EFV to birds-eye view to coordinate recovery
9. Flight Director communicates the location of the rocket to the Recovery Crews

5 Appendices

5.1 Assumptions and dependencies

5.1.1 RV3K is part of a bigger system

The RV3K module is one small piece in all the systems that are used during a PSAS launch. It is assumed that RV3K shall be used in tandem with launchcontrol and other software systems.

5.1.2 Language and compiler dependencies

It is assumed that the user is comfortable following directions to install necessary language and/or compiler software as part of the setup for operating RV3K. These instructions shall be clearly documented.

5.1.3 Version control

It is assumed that the user is comfortable updating software using GitHub and/or version control software. This includes updating a remote repository with configurations and modifications, and cloning, forking, and installing software from a GitHub repository.

5.2 Links to social media

The RV3K module should provide links to PSAS social media. This includes Facebook, Twitter, or any other sites relevant to a given launch or phase of PSAS activity. It is assumed that these profiles are maintained separately from RV3K. RV3K shall be easily configurable to change such links when needed (see §2.3.5).