Software Requirements Specification for Operations Data Analysis and Management System

Version 1.3

Prepared by Jesus Mendoza, Maximillion Orozco, Nathan Lee, and Richard Balbuena

Project Sponsor: Boeing

Created: November 1st, 2019
# Table of Contents

**Revision History**

1. **Introduction**
   1.1 Purpose
   1.2 Intended Audience and Reading Suggestions
   1.3 Product Scope
   1.4 Definitions, Acronyms, and Abbreviations
   1.5 References

2. **Overall Description**
   2.1 Product Perspective
   2.2 Product Functions
   2.3 User Classes and Characteristics
   2.4 Operating Environment
   2.5 Design and Implementation Constraints
   2.6 User Documentation
   2.7 Assumptions and Dependencies
   2.8 Apportioning of Requirements

3. **External Interface Requirements**
   3.1 User Interfaces
   3.2 Hardware Interfaces
   3.3 Software Interfaces
   3.4 Communications Interfaces

4. **Requirements Specification**
   4.1 Functional Requirements
   4.2 External Interface Requirements
   4.3 Logical Database Requirements
   4.4 Design Constraints

5. **Other Nonfunctional Requirements**
   5.1 Performance Requirements
   5.2 Safety Requirements
   5.3 Security Requirements
   5.4 Software Quality Attributes
   5.5 Business Rules

6. **Other Requirements**

3
4
5
6
7
8
9
10
11
12
13
14
15
# Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>All members</td>
<td>11/1/19</td>
<td>Created initial draft</td>
<td>1.0</td>
</tr>
<tr>
<td>All members</td>
<td>11/15/19</td>
<td>Updated document for second draft</td>
<td>1.1</td>
</tr>
<tr>
<td>Richard Balbuena</td>
<td>11/21/19</td>
<td>Format document, ensure cohesiveness</td>
<td>1.2</td>
</tr>
<tr>
<td>Jesus Mendoza</td>
<td>11/26/19</td>
<td>More formatting, added table of contents</td>
<td>1.3</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Purpose

1.2 Intended Audience and Reading Suggestions

This document is intended for:

- The customers to ensure that each of the requirements that were requested were successfully implemented
- For developers and testers to familiarize themselves with what the modules do and their expected behavior of it as well understand the structure of the project for their own personal use

1.3 Product Scope

The main objective of ODAS is to take in structured and unstructured data in the form of spacecraft telemetry and applications logs, and analyze this operational data. This is accomplished through modules that include:

- A database system to store the data sets
- A web based user interface that performs the query and reporting functions
- A report generator to generate the reports as well as export them in different exportable formats
- Graphing and machine learning tools to perform analytics and learn more about what the data is about

1.4 Definitions, Acronyms, and Abbreviations

ODAS—Operations Data Analysis and Management System
API—Application programming interface
COSMOS—Comprehensive Open-architecture Solution for Mission Operations Systems
NOS3—Nasa Operational Simulation for Small Satellites
GUI—Graphic User Interface
1.5 References

Documentation for technologies used:

Django: <https://docs.djangoproject.com/en/2.2/>

React.js: <https://reactjs.org/docs/getting-started.html>

Plotly.js: <https://plot.ly/javascript/>

2. Overall Description

As more and more satellites are launched into orbit, it is important to track the status of each one. We’re developing ODAS, with guidance from our Boeing liaisons, to function as a central hub which focuses on monitoring the Health and Status of your satellites. ODAS does so by analyzing the Health and Status Telemetry for each of your satellites, and providing powerful analytical tools, via a user-friendly Web interface, that allows any user or organization to monitor the state of their satellites.

From the beginning we designed ODAS to be an Easy-to-Use system that deals with all of the dirty work for you, ranging from Fault Detection using Machine Learning to scalable storage solutions to manage millions of Telemetry data points.

2.1 Product Perspective

Initially, ODAS was being developed as an application that would utilize another existing software system called COSMOS. Essentially, we were going to use COSMOS to handle the telemetry processing, then deliver the pre-processed data to our software. This was because we would be assuming that our end-user would be using a COSMOS environment and accumulate data in that proprietary format. So, in that case, COSMOS would provide the functionality for us to analyze the data.

Yet, as we continued to develop our project, we realized that it would be more beneficial for us to generalize our dataset sources to allow us to handle and process telemetry that may be provided by any user or organization. Soon, we will have data that will allow us to interpret and generalize a file format for which the data may be provided as. In such case, we will most likely not require the use of COSMOS, since the application to handle data interpretation would be developed by our team, in-house.

Currently, ODAS provides is planned to include several unique features which are not available on other software applications. In actuality, there exist many software suites that provide functionality to parse and
review telemetry, in addition to sending command telemetry to satellites, such as COSMOS. However, the features COSMOS and similar applications lack, is the ability the perform predictive analysis on the incoming telemetry data. In fact, these software applications are more commonly known as C2 Systems (Command and Control Systems), since they only provide functionality to communicate with a satellite and display/graph the telemetry received.

### 2.2 Product Functions

Contrary to COSMOS, ODAS focuses on being a central analytical hub that allows a user or organization to manage their telemetry by providing storage solutions and data analytical features. These data analytics will consist of the capabilities to predict satellite failures, highlight potentially damaged/abnormal components. Moreover, our analytical processes would allow a user to provide historical data of a failed satellite, and ODAS would help narrow down then potential cause or component failure which may have led to the failure of the satellite. These are key distinct features, not provided in other software applications.

In addition to predictive analysis, ODAS will provide features for managing a user’s uploaded set of telemetry. Our application will accommodate for storage expandability and accessibility, allowing users to reliable access their data via our website, as well upload their telemetry files.

Another large branch of our software application’s focus is data interpretability. Meaning that ODAS will provide many features that allow users to view graphical representations and analysis of their telemetry. For example, through our web interface, users will be able to view graphs of their telemetry measurements. These graphs provide the capability to capture an image of a certain time period or further inspect the rate of change for the telemetry data points plotted. Also, users will be able to create custom graphs that query only the data the wish to plot.

Another aspect of data interpretability that ODAS will provide is report generation and alert notifications. Since users will be providing their email upon sign up, our application will be able to send them relevant information regarding their satellites status in case our predictive models determines an imminent concern. As for generating reports, the user will have the capability to generate reports either on data which they can select, or query for, and/or the entire digested telemetry file. These reports will include any relevant graphs that were generated based on the telemetry as well as warnings information regarding the health and status of their satellites and its components. These are the main features that will be provided by ODAS.
2.3 User Classes and Characteristics

The end user expected to be using ODAS is someone with knowledge about a particular satellite’s telemetry. This would be the optimal user since this would allow them to take full advantage of all of ODAS’s features. For instance, if someone from a particular organization was tasked to use ODAS to narrow down the possible failure points that led to the demise of a satellite using its historical telemetry, and if this individual was unfamiliar with the expected behaviour of this satellite, then ODAS may only provide a general overview of a predicted fault. On the other hand, someone with knowledge about such satellite, could use ODAS by first querying for specific telemetry data points, and in this case ODAS may be able to more accurately isolate a potential fault.

Nonetheless, many of ODAS’s features are designed so that anyone, with at least some knowledge about satellite telemetry (perhaps who works at an organization dealing with satellites and telemetry), could utilize all of the other aspects of ODAS. Namely, features like generating reports, plotting graphs, uploading and managing stored telemetry, as well as some predictive analysis.

2.4 Operating Environment

As for the user’s operating environment, since all of the functionality provided by ODAS will be done so via a web interface, the only required state for the user would be to have a reliable internet connection with a modern browser.

In terms of the environment for which ODAS will be running, there are many steps we have taking into consideration to allow for our application to scale and demand grows. Regarding the environment, ODAS will be running a Python Django backend server that will be connected to remote MySql server, as well as a React.js frontend that will manage the visual aids and the actual web interface. To allow for scalability, and to simplify the deployment process, we are starting to implement Docker into our project.

Containerization provides so great solution for us, especially for our development process. As we were and continue to develop ODAS, we came across some difficulties for reproducing our environments. This led to some difficulties during our development since some code that was run on one machine, had some difficulties running on others. For example, a common problem we faced was installing all of the dependent packages required to run the backend server our our local machines during testing. Docker allows us to containerize our services and ensure that our requirements are always met and can be deployed and run on any environment where Docker is installed. This allows us to avoid the common software development problem of “well it works on my machine”.

2.5 Design and Implementation Constraints

While developing ODAS, we face a few challenges to ensure that the development process is working well. For one thing, we must ensure that our development environments are at least somewhat similar. To simplify the fact of installing dependencies and packages that are required to run each other’s code, we
create environments and dependency lists that will automatically install them when necessary, such as `pipenv`.

Despite this, we are choosing to evolve into using other products, such as containerization with Docker, to ensure that our environments do in fact run the same, so that we can run someone else’s module and ensure that it will function properly and continue to work on our own module that may depend on us using another team member’s code / environment setup.

Here is a brief list of constraints and software that we rely on:

- Docker
- Django
- Node.js
- React.js
- Plotly.js
- MySql
- Python

These will have to be installed by our team of developers to ensure that each module functions properly. Therefore, there is a minimum hardware constraint to ensure that everyone has a system capable of downloading, installing, and executing code and modules reliant on these software.

### 2.6 User Documentation

For this project, we plan on developing a simple user guide to show the users how to access each feature provided by ODAS as well as documentation regarding the implementation which they could read.

### 2.7 Assumptions and Dependencies

Currently, our project includes some assumptions that we rely on. For instance, we are currently unaware of how the data will be formatted, but it must be a parsable format. In the case where it was delivered as binary data, if the format for which it was stored is proprietary, then we would at least need access to an application that could encode / decode the data in such format. In turn, we would be unable to provide a generalized solution that would allow multiple users / organizations to use ODAS, since it would be developed on the basis of interpreting this proprietarily formatted telemetry.

Also, we are assuming that we will be provide will semi-real telemetry data, that does not consist of manipulated or randomly generated telemetry. This is an important factor for the success of ODAS; Because in order to develop accurate predictive machine learning models that can accurately narrow down potential faults based on historical telemetry, it is essential that we train our machine learning algorithms to learn how satellites are likely to fail, rather than train it on telemetry that was manufactured.

In the case that we are unable to collect and use realistic and undoctored data, we would not be confident that ODAS’s predictive models would be representative to a satellites behaviour. Meaning that our
algorithms would most likely be over-fit to our specific dataset and would not extrapolate well when presented with new data. Essentially, rendering our fault predicting capabilities severely handicapped.

2.8 Apportioning of Requirements

Due to the lack of data, we had decided to split up the development out ODAS into two stages (for two semesters). In the first stage, we will be and are focusing on developing the infrastructure that will house our project. In other words, we are developing the website, the interface, the API, and setting up all of the remote services, like remote databases and virtual machines, that will focus on how ODAS functions.

Then, during the second stage, we will focus on Data Analysis, since this is a major component of ODAS. By doing the infrastructure during stage 1, we can easily isolate the machine learning from the actual content delivery and user interactions with our website. As we anticipate collecting data in the coming months, that is when we will begin dedicating time to develop predictive models and testing various machine learning algorithms, as well as optimizing them for optimal performance and the best results for ODAS.

3. External Interface Requirements

3.1 User Interfaces

The user will enter in the (satellite) data as input and will be able to provide or collect data from COSMOS or a database. The user will have the ability via the interface to manage and query data in the database over a user-specified time period or a relative time period, create a query to search unstructured data, generate and upload a dataset, develop and save a query with a label for later use or as a source of data for plots or exports, monitor ODAS system performance, plot data in graph form, perform ODAS system administration and generate, display, retrieve and export reports in common ASCII formats (comma-separated, space-separated, tab-limited, JSON, XML).

In accordance with the Americans with Disabilities Act Title II Regulations, Subpart F, Clause 35.177, an online manual for operation of any GUIs will be provided and easily accessible.
3.2 Hardware Interfaces
This system shall be compatible with Internet operating systems. We will also be utilizing cloud services as well to offer remote access to data and the database for the user. As the application must be run over the internet, additional hardware required to connect the user to the internet may be required for the system.

3.3 Software Interfaces
Python Version 3, latest version found at https://www.python.org/downloads/source/,
- Used for main back-end architecture and functionality.
Javascript Plotly Version 4.2.1, latest version found at https://pypi.org/project/plotly/,
- Used for user-interface functionality, including Node and React.
- Used for graphing functions in the user-interface.
Django Version 2.2.6, latest version found at https://www.djangoproject.com/download/,
- Python toolkit used for back-end HTTP responses and back-end website architecture.
MySQL Version 8.0.12, latest version found at https://www.mysql.com/downloads/
- Used for back-end database management, storage, and retrieval.
Operations Data and Management System API, accessed by JSON Fetch methods.
Ball Aerospace COSMOS 4, latest version found at https://cosmosrb.com/docs/installation/

3.4 Communications Interfaces
This system shall use the JSON format for APIs, HTTP protocol for email and internet communication, TCP/IP protocol suite for intranet communication. CORS will be enabled for communication with the API.
4. Requirements Specification

4.1 Functional Requirements

4.1.1 Data Storage Solution
   4.1.1.1 The application shall have a database that is capable of scaling to large data sets

4.1.2 API for External Applications
   4.1.2.1 The application should allow the capability of ingesting, retrieving and generating reports using an API
   4.1.2.2 The application should be able to export data into common ASCII formats including CSV, Space Separated, Tab-Delimited, JSON, and XML.

4.1.3 Web Based User Interface
   4.1.3.1 The application shall have a web based interface to perform various functions
      4.1.3.1.1 The application should be able to create a query for a list of one or more telemetry points over a user specified time period
      4.1.3.1.2 The application should be able to create a query for a list of one or more telemetry points over a relative time period
      4.1.3.1.3 The application should be able to create a query to search unstructured data
      4.1.3.1.4 The application should be able to develop and save a query with a label for later use or as a source of data for plots or exports
      4.1.3.1.5 The application should be able export data to a file in one or more available formats
      4.1.3.1.6 The application should be able to plot data for one or more several telemetry points in one of several plot formats
      4.1.3.1.7 The application should be able to format telemetry plots
      4.1.3.1.8 The application should be able to Monitor ODAS system performance
      4.1.3.1.9 The application should be able to perform ODAS system administration

4.1.4 Report Generator
   4.1.4.1 The application shall have a report generating function to create reports from ODAS data based on specific or saved queries
   4.1.4.2 The application shall be able to send reports using email as a one time execution or a recurring basis such as daily, weekly or monthly
   4.1.4.3 The application shall be able to send reports to a file server as well
4.1.5 Data Search
   4.1.5.1 The application shall have a query interface that allows for easy searches on data
   and displaying the results as well as filter queries

4.1.6 Data Analytics
   4.1.5.1 The application shall have data analytics tool suites that allow the user to exploit
   the data using the ODAS system to make predictive fault predictions or predictive fault
data discovery

4.2 External Interface Requirements

Two-Line Element (TLE) file - Compressed data that is formatted into two-lines. The first line provides
orbital data such as Satellite Number Timestamp, and temperature of satellite instruments. The second
line provides data such as Satellite Number, Inclination in Degrees, Motion, and Revolution Number.
Command files and Log files are input into the ODAS system.

Telemetry file information is stored into the database and is used by the various modules that ODAS has
such as the report generator, the data analytics tools and the graphing tools.

Files are sent out from the report generator through either emails or to a file server where the user can
view their reports at their leisure. The emails are sent as text and the files are stored on the file server in
the .txt format.

Other things to be input are user credentials that are required on the site in order to access the file server
for the users to see all their files.

4.3 Logical Database Requirements

This section specifies the logical requirements for any information that is to be placed into a
database.

This may include:
   ● Types of information used by various functions
   ● Frequency of use
   ● Accessing capabilities
   ● Data entities and their relationships
   ● Integrity constraints
   ● Data retention requirements
If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.

The database schema consists of three tables: Measurement, Satellite, and Components. Measurements and components are tied to a specific satellite based on their satellite ID field and keep track of various important measurements and components that are relevant to the satellites. The database is being hosted online on a free tier of Amazon’s Relational Database System (RDS).

4.4 Design Constraints

We do not have any real data to use for developing and testing our database to date so we have had to use made up sample data while we wait for our liaison to provide us with data. This slows down the development process as we cannot create the machine learning module when we do not know what kind of data we are working with.

Functions that receive data are still not finalized in the sense of how we will be receiving the data, i.e the format in which it is received such as CSV, TLE, tab delimited, etc.
5. Other Nonfunctional Requirements

5.1 Performance Requirements

Data will in large part be kept on the server side of the application. Thousands of data points are processed, analyzed and plotted on the server which are then sent to the front end UI. Since most of the computation will be done on a server and not locally, machine specifications and hardware matter little. The speed of data gathering and processing depend in majority on the internet connection specific terminals will have at the time of use.

This application began with scalability in mind. Django web servers are known to be easily scalable, which is part of the reason for its popularity. Since Django allows multiple servers for multiple modules it is able to distribute and balance workloads well. Django combined with a well designed MySQL database and ReactJS front end allows for an efficient and highly scalable application.

5.2 Safety Requirements

There are no safety precautions that need to be taken while using the application. No risk of physical harm is involved in its use.

5.3 Security Requirements

Using the application itself has no security risk. However when handling the application it is essential that the data be kept private. Data being used and analyzed is data that is directly related to satellites currently in commission. The satellite data being collected and stored is rightfully owned by Boeing’s customers and owners of their respective equipment. Each customer is entitled to any data related to its corresponding satellite. Redistribution of the data can result in legal recourse against the offenders. Precautions must be taken to avoid accidental leaks of the data. Those that have direct access to satellite data may be asked to sign a Non-Disclosure Agreement (NDA) to protect the interests of all parties involved.
5.4 Software Quality Attributes

Due to the use of containers, the application can be easily maintained and is highly portable. For this particular case, Docker was used to build a containerized environment and keep development and maintenance efficient. All application servers will be deployed on a web server hosted by a free tier of AWS. This allows the application to be as available as AWS continues to be.

The application is being designed to be as easy to use for the user as possible. The UI is kept simple with its functions remaining as obvious as possible. Where the operations are not clear, there will be documentation provided on the website itself describing the application.

5.5 Business Rules

As mentioned in the previous section regarding security requirements, users should practice caution while using the application. The application will have user authentication allowing for users to save certain configurations specific to their uses. Authentication will also allow for regulation of who accesses the data being collected and processed since in many cases the data being used may be classified. Users must be aware of the consequences of the exposure of the data that they are handling.

6. Other Requirements

There are some additional requirements needed with regards to data collections and usage. Since we will be receiving data from a private company, it is required that we sign Non-Disclosure Agreements which will ensure that we cannot disclose any particular information which can describe or identify the data. However, we can discuss what we do and perform with the data, just not the contents of it.