Software Requirements Specification

for

Spacewalk Task Generator

Version 1.0

Prepared by Group 3

University of Maryland University College

September 23, 2018

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 2

3. External Interface Requirements 3

3.1 User Interfaces Overview 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

4. System Features/Modules 3

4.1 System Feature/Module 1 3

4.2 System Feature/Module 2 (and so on) 4

5. Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 4

5.3 Security Requirements 4

5.4 Software Quality Attributes 5

Appendix A: Glossary 5

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Jose  DelaCruz-Ramirez | 09/23/18 | Initial draft | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

NASA spacewalks or Extravehicular activity (EVA) checklists are a list of procedures typically performed by 2 astronauts working outside the space station with the support of a ground crew and a robotics operator inside the space station.

These procedures are generated months in advanced in an electronic document containing a table of steps, images, dependencies and responsible personnel, usually referred as EV1 or EV2.

The proposed solution will allow the document creator to create the EVA checklist in YAML (YAML Ain’t Markup Language) YAML format, run a conversion tool and generate the document in the desired format, utilizing a Node Package Manager (NPM) command.

The initial version of the NPM package will allow the creator the generation of documents from a single YAML file, future versions of the package will enable the implementation of shared YAML tasks and embedded images within the tasks, but these requirements will be out of scope for version 1.

## Intended Audience and Reading Suggestions

The document is intended for developers, project managers, program managers, testers, NASA EVA checklist creators and SWEN 670 project group.

## Product Scope

The application will be a Node Package Manager (NPM) solution to be executed as a terminal or console application as either a scheduled task or manually executed, it will receive a YAML file path and an output path as arguments. If the YAML file contains images those images must be located at the same path as the YAML file. The NPM package will generate an HTML file with the proper checklist and leverage the use of Pandoc tool to convert the HTML file into a Word document both to be saved in the desired output argument.

## References

Ben-Kiki, O., Evans, C., & Net, I. d. (2009, 10 01). *YAML Ain’t Markup Language (YAML™) Version 1.2*. Retrieved from YAML.org: http://yaml.org/spec/1.2/spec.html

Montalvo, J. (2018, 09 21). *Spacewalk*. Retrieved from Github: https://github.com/jamesmontalvo3/spacewalk

NASA. (2011, 07 28). *Flight Data Files*. Retrieved from Johnson Space Center: https://www.nasa.gov/centers/johnson/news/flightdatafiles/foia\_archive.html

# Overall Description

## Product Perspective

The product is a self-contained package developed as a JavaScript NPM package that allows EVA creators to save the checklist steps as YAML files and pass the saved “.yaml” or “.yml” file as an argument and get the document, this process can be re-run as many times as desired. If changes are made to the YAML file, the document generated should reflect those changes with the appropriate updates.

## Product Functions

* No user interface required, the application should be able to run from the command line by typing the name.
* Received a YAML file as input containing the steps, dependencies, images and astronaut performing the step.
* Output a Word document that can be adjusted if needed based on the data that came from the YAML file.

## User Classes and Characteristics

The tool is intended for users with basic knowledge of command line execution. Basic knowledge of Command Prompt or PowerShell (if Windows) or Unix shell (Linux/Mac) is needed for usage.

Basic knowledge of Node.js and NPM is required for the installation and configuration of the tool.

## Operating Environment

The task generator tool is cross platform and can operate on any operating system with Node.js 8.12 or higher and NPM 6x or higher (installed with Node.js)

## Design and Implementation Constraints

* Multiple YAML as inputs are a desired functionality but will not account for initial release due to time constraints.
* Task dependencies are going to be flagged from the YAML file, the tool will not validate dependencies, but rather generate the file “as is”. The user will have to be responsible to review and adjust the YAML file if improper data is found.
* If the same file is submitted as parameter the tool will generate a new document, if the same argument is provided then the document will be overwritten.
* Images have to be generated in the position specified in the YAML but for release 1 they may have to be imported at the end of the document due to time constraints.
* 3rd party tools for document generation will be used to build the output as long as they are MIT or Apache licensed.

## User Documentation

Written documentation will be provided from the Github repository page (Montalvo, 2018).

## Assumptions and Dependencies

YAML files will be saved in separate GitLabs repository for historic maintenance and tracking. The application will not be saving anything in a database.

No internet connection or internet interaction is expected by the application.

A document processing application will be installed in the computer to open the generated file. A browser can be used as an alternate if the user opens the generated HTML file.

Node and NPM will be installed from the computer to be executing the application.

No specific operating system is enforced to be utilized by the application as long as NodeJS is installed.

# External Interface Requirements

## User Interfaces Overview

*The task generator command-line interface will allows the creator to use it from the Unix and DOS command lines, and any other environment where Node.js is supported to supply a file with the list of steps and a path to save the output.*

## Hardware Interfaces

No Mouse interaction is needed to execute the app, the keyboard is required to execute the commands, interaction with touchscreens will require the ability to type to the command-line.

# System Features/Modules

## Help

|  |  |
| --- | --- |
| C:\> eva-checklist --help | C:\> eva-checklist -h |

4.1.1 Description and Priority

Give the user the different arguments that can be passed to the tool. This same output must be returned when the user provides invalid parameters.

4.1.2 Stimulus/Response Sequences

The user should be able to stay in the terminal and continue the execution of any other terminal command after the help is displayed. The terminal must not be paused after the help is displayed to maintain the standard convention of command prompt executions.

4.1.3 Functional Requirements

REQ-1: Display the basic call of the tool in a single line in such a way that the user can copy/paste the output and adjust the arguments.

REQ-2: Explained each argument that can be passed to the tool and specify whether they are optional or required.

## Generate EVA checklist

|  |
| --- |
| C:\> eva-checklist --input pma3-shields-and-connections.yml --output |

4.1.1 Description and Priority

Execute the application main functionality.

4.1.2 Stimulus/Response Sequences

The user should be able to receive notification of the application state while the application goes through the different stages of the conversion.

4.1.3 Functional Requirements

REQ-1: input parameter is required

REQ-1.1: Input parameter must be a valid file path.

REQ-1.2: Input parameter file must be YAML-file formatted with extension “.yml” or “.yaml”

REQ-2: Output parameter is optional

REQ-2.1: If output parameter is provided then generated files will be saved in the directory provided.

REQ-2.2: If the directory provided in the output doesn’t exist, the application will attempt to create it.

REQ-2.3: if access is denied or errors creating the directory happens, notify the user and exit.

REQ-2.4: The output file name must match the YAML file name with a different extension.

REQ-2.5: The HTML file generated as checklist must be kept.

REQ-3.1: If the YAML contains references to images they must exist in the same directory as the YAML file.

REQ-3.2: The final output file must be a Word document.

# Nonfunctional Requirements

## Security Requirements

Node.JS and NPM are Open Source solutions that must be kept properly updated through the Nodejs.org website to maintain their vulnerabilities patched.