text divider

Project Plan

for

NASA Maestro Format Test Tool

**Version 1.0**

**Prepared by:**

**NASA TEAM 1**

**University of Maryland Global Campus**

**23 February 2020**

**Authors**

* Alberto Bonfiglio
* Kenya Foster
* Beatrice Oluwabuyi
* Jacquetta Reid
* Rick Stuart
* Tiezheng Yuan

**Revision History**

|  |  |  |
| --- | --- | --- |
| Date | Reason for Update | Version |
| 02/23/20 | Initial Document | 1.0 |
| 03/15/20 | Updating based on suggestions | 2.0 |

**Table of Contents**

[**1.**](#_heading=h.gjdgxs) Introduction 4

[1.1.](#_heading=h.30j0zll) Statement of Need 4

[1.2.](#_heading=h.1fob9te) Project Summary 5

[1.2.1.](#_heading=h.3znysh7) Purpose 5

[1.2.2.](#_heading=h.2et92p0) Scope 5

[1.2.3.](#_heading=h.tyjcwt) Assumptions and Constraints 5

[1.2.4.](#_heading=h.1t3h5sf) Project Deliverables 6

[1.3.](#_heading=h.4d34og8) Definitions and Abbreviations 6

[**2.**](#_heading=h.17dp8vu) References 7

[**3.**](#_heading=h.3rdcrjn) Project Organization 8

[3.1.](#_heading=h.26in1rg) External Structure 8

[3.2.](#_heading=h.lnxbz9) Internal Structure 9

[3.2.1.](#_heading=h.35nkun2) Roles and Responsibilities 9

[**4.**](#_heading=h.1ksv4uv) Managerial Process Plan 10

[4.1.](#_heading=h.44sinio) Project Tasks and Activities 11

[4.1.1.](#_heading=h.2jxsxqh) Major Tasks 11

[4.1.2.](#_heading=h.z337ya) Minor Tasks 11

[4.2.](#_heading=h.3j2qqm3) Schedule 11

[4.2.1.](#_heading=h.1y810tw) Deadlines 11

[4.2.2.](#_heading=h.4i7ojhp) Schedule Allocation 12

[4.3.](#_heading=h.2xcytpi) Resource Allocation 13

[4.4.](#_heading=h.1ci93xb) Risk Management 14

[**5.**](#_heading=h.3whwml4) Technical Process Plan 15

[5.1.](#_heading=h.2bn6wsx) Process Model 15

[5.2.](#_heading=h.qsh70q) Methods, Tools, and Techniques 17

[5.2.1.](#_heading=h.3as4poj) Development Collaboration 17

[5.2.2.](#_heading=h.1pxezwc) Software Architecture 17

# Introduction

## Background

National Aeronautics and Space Administration (NASA) is an independent agency of the United States Federal Government responsible for the civilian space program, as well as aeronautics and aerospace research. NASA was established in 1958, succeeding in the National Advisory Committee for Aeronautics. NASA responsibility, purpose and path are embedded in the agency’s mission and vision statements as shown below:

Vision: “We reach for new heights and reveal the unknown for the benefit of humankind.”

Mission: “Drive advances in science, technology, aeronautics, and space exploration to enhance knowledge, education, innovation, economic vitality, and stewardship of Earth.”

NASA’s Vision leads to a future with an American-made launch capability supporting cutting-edge science, technology, and human exploration with strong technology and aeronautics programs. Its mission statement outlines NASA’s fundamental purpose and role in bringing that Vision to life. In its continuous commitment to the fulfillment of its vision and to enhance the efficiency and effectiveness of its spacewalks, the agency decided to update its Maestro software suite, a tool used to create procedures for spacewalks (EVAs), through “NASA Maestro Format Test” project. This project plan is designed to guide the execution of the project by the project team.

## Statement of Need

The customer, National Aeronautics and Space Administration (NASA), needs some updates to the Maestro software suite, which is a tool used at NASA to create procedures for spacewalks known as Extra-Vehicular Activities (EVAs) or spacewalks undertaken outside of the host vehicle. In general, these procedures include two astronauts working outside the space station, a robotics operator inside, and other actions taken by Mission Control on the ground. Maestro currently takes the procedure data and can render it in many file types (HTML, DOCX, specific XML types) as well as different formats (EVA format, IVA format, etc.). The software facilitates basic uses, but NASA wants to extend the software’s use cases. Before that, NASA needs additional testing and format functionality.

The customer would like for Team One to focus on extending existing testing and validation functionality and to lay the groundwork for moving the project to Gitlab.

This project will provide the following values for NASA Maestro software:

* Create a mechanism to perform an "Is this valid Microsoft Word document" test. The mechanism must be able to be run on a non-Windows system without MS Word installed.
* Create a way to generate a screenshot of an MS Word document to compare new outputs with previously-stored ones as additional validation. This must be able to be run on a non-Windows system without MS Word installed.
* Create additional end-to-end test examples, runnable by the CI/CD framework, and automatically included in the documentation with links allowing to run in online example environments such as Runkit.
* Build an API documentation site (on each successful merge) that provides the ability to access documentation for each version.
* Pending the customer publishing some additional functionality to the Maestro Repository, create a test that builds new XML, applies a customer-supplied style sheet to it, creates a screenshot, and compares the output to an original screenshot

## Project Summary

### Purpose

This software shall be designed to ensure that .docx files created by Maestro are valid and then compare expected formatting of the provided Yet Another Makeup Language (YAML) to a desired or expected output as a picture file. YAML is the filetype currently used to store EVA templates (An example is available for download in the references section). The software will maintain a record of the actual output and compare it to both the new expected output and the actual output to maintain a record of changes in Yet Another Makeup Language (YAML) formatting. An example of the YAML used by the NASA Maestro software is available in the references section.

### Scope

This product shall be developed to test Maestro and to provide additional capability following the completion of document conversion. This shall include the ability of the program to discover and highlight changes in the document formatting, and to display those changes to Maestro developers. Additionally, the automation of testing, build, and potential deployment shall be incorporated by transferring the repository to a NASA hosted Gitlab from its current git location on NASA’s GitHub repository.

### Assumptions and Constraints

* The software shall be system independent and specifically designed to run without the use of Microsoft Windows or Microsoft Word.
* One challenge created by the potential use of Linux is the requirement to support the Arial font which is not provided within GNU Linux. The use of created container images should alleviate this concern.
* The software testing suite (currently Mocha) needs to instantiate the docker container providing the platform-agnostic document testing and formatting (explained below). Preliminary research indicates that this is possible with NodeJS and commander.js but additional research is needed.
  + 1. Project Deliverables

NASA currently has five priorities that were specified in the initial customer meeting. They want Team One to:

1. Create a test that builds new XML, applies a customer-supplied style sheet to it, creates a screenshot, and compares the output to an original screenshot
2. Create a mechanism to perform an "Is this valid MS Word document" test. This must be able to be run on a non-Windows system without MS Word installed.
3. Create a way to generate a screenshot of an MS Word document (perhaps Word-to-PDF then use preexisting PDF-to-screenshot). This must be able to be run on a non-Windows system without MS Word installed.
4. Create end-to-end test examples that are both run by CI but are also automatically included in the documentation with links to run in online example environment (e.g. Runkit)
5. Build API docs site (on each successful merge) that handles docs for each version (e.g. dropdown that allows showing docs for v1.0, 2.0, etc.)
6. Mirror Maestro project on gitlab.com and make all testing/etc. work

These NASA requirements have been broken down into eight deliverable products templated below:

|  |  |
| --- | --- |
| Deliverable | Summary |
| Maestro Integration Documentation | Documentation tracking any changes made to the Maestro Software to integrate the testing and demonstration software. |
| Software Documentation | Development documentation detailing the codebase. |
| Testing Software | Software that tracks changes in the formatting done by Maestro and ensures that the user desired formatting matches the final product created by Maestro. |
| Move to Gitlab | Move NASA Maestro and team developed software to Gitlab. The DevOps team is providing most of the work. |
| Testing through Runkit | Convert existing test cases to run automatically using Runkit. |
| API Versioning | Complete a system through Gitlab that tracks API changes. |
| XML Maestro Testing | Once XML output is finalized within Maestro, conduct testing to ensure accurate production. |
| Maestro Integration Documentation | Documentation tracking any changes made to the Maestro Software to integrate the testing and demonstration software. |

## Definitions and Abbreviations

|  |  |
| --- | --- |
| Word | Definition |
| Extravehicular Activity | Any activity that is done by an astronaut or cosmonaut outside a spacecraft beyond the Earth's appreciable atmosphere. |
| Intravehicular Activity | Any activity that is done by an astronaut or cosmonaut inside a spacecraft beyond the Earth's appreciable atmosphere. |
| NASA MAESTRO | An open-source program released by [NASA](https://en.wikipedia.org/wiki/NASA) to facilitate the creation of EVA and IVA procedures. |
| YAML | A [human-readable](https://en.wikipedia.org/wiki/Human-readable) [data-serialization language](https://en.wikipedia.org/wiki/Serialization). It is commonly used for [configuration files](https://en.wikipedia.org/wiki/Configuration_file) and in applications where data is being stored or transmitted. |
| .docx | A DOCX file is a document created by Microsoft Word or another [word processing](https://techterms.com/definition/wordprocessor) program, such as OpenOffice Writer or Apple Pages. It contains formatted text but may also include images, drawn objects, and other document elements. |
| Vector | A point that has a definite position on the x- and y-axis of the work plane and determines the direction of the path; further, each path may have various properties including values for stroke color, shape, curve, thickness, and fill. |
| GNU | An operating system that is [free software](https://www.gnu.org/philosophy/free-sw.html). The GNU operating system consists of GNU packages (programs specifically released by the GNU Project) as well as free software released by third parties. |
| YAML | A [human-readable](https://en.wikipedia.org/wiki/Human-readable) [data-serialization language](https://en.wikipedia.org/wiki/Serialization). It is commonly used for [configuration files](https://en.wikipedia.org/wiki/Configuration_file) and in applications where data is being stored or transmitted. |
| SDLC | The Software development lifecycle is a process used by the software industry to design, develop and test high-quality software. |

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| NASA | National Aeronautics and Space Administration |
| HUD | Heads-Up Display |
| EVA | Extravehicular Activity |
| IVA | Intravehicular Activity |
| HTML | Hypertext Markup Language |
| DOCX | Microsoft Word Document Format |
| XML | Extensible Markup Language |
| API | Application Programming Interface |
| CICD | Continuous Integration and Continuous Deployment |
| PM | Project Manager |
| CICD | Continuous integration and continuous delivery |
| PDF | Portable Document Format |
| REST | Representational State Transfer |
| UI | User Interface |

# References

* Sample EVA procedure available at <https://www.nasa.gov/centers/johnson/pdf/539922main_EVA_134_F_A.pdf>.
* The current NASA Maestro source code is available at <https://github.com/xOPERATIONS/maestro>.
* Kanban link at <https://github.com/xOPERATIONS/maestro/projects/2>.
* STS-134 EVA YAML example is available at <https://gitlab.com/xOPERATIONS/sts-134>.

# Project Organization

This is the organization chart of the company

## External Structure

The external structure of the project consists of direct communication by the PM with the customers at NASA and with the DEV/ARCH team within the UMGC framework.

Project Manager

Rick Stuart

Development Team

Alberto Bonfiglio

Tiezheng Yuan

Requirements Analyst Team

Jacquetta Reid

Beatrice Oluwabuyi

Kenya Foster

Software Testing

Alberto Bonfiglio

Tiezheng Yuan

Jacquetta Reid

DEVOPS

Anthony Lockhart

## Internal Structure

The internal structure consists of members from NASA Team 1, representing the development team from NASA.

The roles of the internal personnel are defined below.

### Roles and Responsibilities

The project team consists of the following roles and responsibilities. Those roles highlighted below are the members' primary role. Each member contributes to additional roles as required by the project.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Responsibility | Allocation |
| Alberto Bonfiglio | System Developer/Engineer | Requirements gathering, System design, Development, and Integration | 200 hours |
| Beatrice Oluwabuyi | System Developer/Tester | Development and Testing | 200 hours |
| Jacquetta Reid | System Developer/Tester | Development and Testing | 200 hours |
| Kenya Foster | System Analyst/Tester | Definition, Testing, and Training | 200 hours |
| Rick Stuart | Project Manager/Scrum Master | Requirements gathering, budget, and final approval authority | 200 hours |
| Tiezheng Yuan | System Developer/Engineer | Requirements gathering, System design, Development and Integration | 200 hours |

Responsibilities Using RACI Model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Name | Project Sponsor | Project Manager | Requirement Analyst | Application Development | Software Tester |
| Stage A | Requirement Gathering | C | A | R | I | I |
| Stage B | Project Documentation | C | A | R | R | I |
| Stage C | Initiate Project | C | A | R | R | R |
| - C01 | Project Development | I | R | R | A | R |
| - C02 | Project Testing | I | R | R | C | A |
| Stage D | Design solution | I | A | R | R | R |

#### Project Manager

Rick Stuart, the Project Manager (PM) for this project, acts as a liaison between the teams and all the stakeholders, the PM will communicate clearly and effectively any needs or concerns from the teams to affected stakeholders and vice versa. The PM coordinates with all team members to ensure the deliverables are completed on time and following all specifications. Furthermore, the PM organizes all group chat sessions, phase reports, and student-professor communications. As Scrum Master, Rick is responsible for improving interactions between the development team and the organization in other to maximize the productivity of the Scrum team. He arranges and facilitates the team’s meetings – daily Scrum, planning sessions, sprint retrospective, and other team activities, as the facilitator. He manages the process for how information is exchanged.

#### Systems Analyst

The software requirements analysts, Beatrice Oluwabuyi, Kenya Foster and Jacquetta Reid are responsible for evaluating the customer’s needs and converting them into specific software requirements. Furthermore, the analyst oversees coordinating all necessary technical documentation in the project, including the Software Requirements Specification (SRS). By working closely with all the stakeholders, the analyst can establish the baseline of the project requirements with enough data expertise for the project plan, software development, and testing.

#### System Developer/Engineer

The software developers, Alberto Bonfiglio and Tiezheng Yuan, responsibilities include the use of software development languages and tools to write, optimize, and maintain computer software for the project system. The developers shall exercise and follow the Agile Scrum framework to plan, design, build and deploy a relevant map/web-based software application. The final deliverables shall meet or exceed the customer’s requirements and expectations.

#### System Tester

Alberto Bonfiglio, Tiezheng Yuan, and Jacquetta Reid are responsible for reviewing software requirements and preparing appropriate test case scenarios. The test cases must be executed fully and iteratively to ensure software usability. By analyzing test results, proper reports are produced for the software development team to help eliminate errors, bugs, and other defects that can detract from the overall user experience of the software.

# Managerial Process Plan

The managerial process plan emphasizes the work schedule and task management needed to meet the agreed-upon project goals. This section defines the project's schedule, the focus of development, how resources are allocated, and the method of development.

## Project Tasks and Activities

Project tasks, which are a single piece of work, or units of related work, that must be completed to satisfy the Maestro project deliverable or the requirements of the deliverable, are broken down in this section. The breakdown of the work of the project into manageable components ensures that time, resources, and cost estimates are easily determined. Breaking down the deliverables into tasks makes the project manager's job easier because the work is subdivided into small units that are easily assigned to one team member or a group of team members. The rest of section four highlighted and described the project tasks breakdown.

This project has

### Major Tasks

These are project tasks that must be undertaken to satisfy the overall project requirements. Each task is accomplished by a set deadline and contributes toward work-related objectives.

* Documentation for project deliverables: project plan, requirements specification, use cases, and Scrum backlogs.
* Requirements elicitation and gathering
* Data and process modeling
* User interface design
* Software testing

### Minor Tasks

These are project tasks that help to aid in the overall project development and major tasks. All tasks are useful in keeping project organization, keeping each team member abreast of project details, and keeping the project on schedule.

* Weekly recurring tasks include status reports, status discord meetings, project management schedule and updates
* Functional group stand-ups meetings
* Source code management

## Schedule

The following section details both the work breakdown and project schedule for NASA Team 1.

### Deadlines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Task Name | Duration | Due | Resource Names |
| Documentation  (Milestone 1) | Project plan, Requirement Specifications | 15 | 02/23/2020 | Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan |
| Design and Analysis  (Milestone 2) | Use cases, software architecture, data modeling, process modeling | 10 | 03/07/2020 | Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan |
| Development  (Milestone 3) | The Scrum cycle including development, coding work, and testing | 20 | 04/05/2020 | Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan |
| Final Phase  (Milestone 4) | Implementation and project delivery | 10 | 04/26/2020 | Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan |

### Schedule Allocation

The table below illustrates the project schedule broken down by sprint.

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint | Start Date | End Date | Deliverables |
| 1 | 10 FEB 2020 | 23 FEB 2020 | * Project Plan * SRS - Completed by all members * Create a testing framework * Develop and prototype file comparison solutions |
| 2 | 24 FEB 2020 | 08 MAR 2020 | * Develop API specification * .docx file checking and conversion * Project Plan and SRS refinement * Parallel test development with .docx. conversion |
| 3 | 09 MAR 2020 | 22 MAR 2020 | * Maestro integration * Image Comparison * API document finalization * Demonstration creation * Test .docx conversion |
| 4 | 23 MAR 2020 | 05 APR 2020 | * Testing Software integration * Gitlab integration * Complete development documentation |

## Resource Allocation

Listed are all resources that will be used to complete the project. This project plan will allocate the resources described below to fulfill the project. All listed resources are considered based on their availability, project time and budget.

|  |  |
| --- | --- |
| Resources | Allocation and Source |
| Project Team | * Roles of Team Members: All team members make contributions throughout the project that matches existing skill sets. * Skills of Team Members: Skills of each member shall be activated and further polished. After the project is delivered, additional knowledge or skills shall be added to team members. |
| Customer Support | * Requirements gathering: Requirements are done through bi-weekly conference calls. Initial meetings focus on overall requirements and refining customer needs. Once prototypes are available, meetings focus on refining the exact product and ensuring the interface matches customer expectations. * Contextual Documents: Documents are open to all team members. Team members edit sections of the project documents that correspond to their role descriptions. Developers focus on program documentation. Testers focus on defining use cases and documenting written tests. Analysts focus on the project plan and ensuring software remains up to date throughout development. |
| Software Tools | * Professional Software: All software and code will be available to all team members. Any data files are accessible by all team members. |

## Risk Management

The purpose of the risk management plan is to identify, analyze, and prioritize risk factors. The plan will also include specific risk mitigation strategies. The probability of a risk occurring is graded on a Low - High scale with 1 being the lowest and 5 the highest probability of the risk occurring.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Insignificant** | **Negligible** | **Moderate** | **Extensive** | **Significant** |
| **Delay** | The feature requires additional focus within the sprint. | Feature testing not completed during the sprint. | Feature not completed during the sprint and pushed to the next sprint. | Multiple features not completed during the sprint and pushed to the next sprint. | Feature(s) not completed during project development. |
| **Capabilities** | Backend performance targets are not reached. | Minor development requests are modified to reduce the development burden. | Feature ships with minor bugs due to incomplete testing. | Features are not complete enough for production software. | Feature not completed during project development. |
| **Developer Time** | A minor increase in developer time required. | The developer has more than one hour of additional work to do. | The developer must shift focus for a sprint to a single issue. | Developers are shifted to handle the issue. | Developers will not have enough time to complete work. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Chance** | **Frequency** | **Probability** |
| **Almost Certain** | Is expected to occur | This occurs in most of the development. | > 95% |
| **Likely** | Will most likely occur. | Occurs in seven of ten development projects experienced by the team. | > 65% |
| **Possible** | Might occur at some point. | Has occurred in development projects experienced by the team. | > 35% |
| **Unlikely** | Could occur at some point. | Has been experienced by the development team but not often. | < 35% |
| **Rare** | It only occurs in exceptional circumstances. | Has occurred to a member of the development team or is an identified risk. | < 10% |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Insignificant** | **Negligible** | **Moderate** | **Extensive** | **Significant** |
| **Almost Certain** | 6 | 7 | 8 | 9 | 10 |
| **Likely** | 5 | 6 | 7 | 8 | 9 |
| **Possible** | 4 | 5 | 6 | 7 | 8 |
| **Unlikely** | 3 | 4 | 5 | 6 | 7 |
| **Rare** | 2 | 3 | 4 | 5 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Mitigation | Risk Level |
| Scope Creep | The scope for this project expands over its initial boundaries. New requirements are added. | Productive initial customer meetings produced a solid list of requirements with a good understanding of both parties. | 7  (Likely/  Moderate) |
| Testing/  Debugging | Testing and debugging requirements extend features development time and complicate development. | The testing protocol is developed before development and testing are completed in parallel with development. | 7  (Likely/  Moderate) |
| Image Comparison | Image comparison cannot be added with an open-source library and must be developed in house. | Open-source libraries will be used as much as possible even if they must be modified. | 7  (Unlikely/  Significant) |
| Image Comparison Performance | The number of pixels that must be compared is extensive and can require too much time with performance hardware. | Balance comparison performance with accuracy to meet the users’ needs while minimizing run time. | 5  (Possible/  Moderate) |
| Outside Time Concerns | Students can have outside influences that require their time. | All team members will participate in all phases of development and documentation to allow for changing individual schedules. | 6  (Likely/  Negligible) |

# Technical Process Plan

The technical process plan will explain the process model being utilized as well as the tools, techniques, and methods to be used in the development of the software. The team utilizes Scrum as a development framework and VS Code as the primary development suite. Other developments and collaboration tools are highlighted in this section.

## Process Model

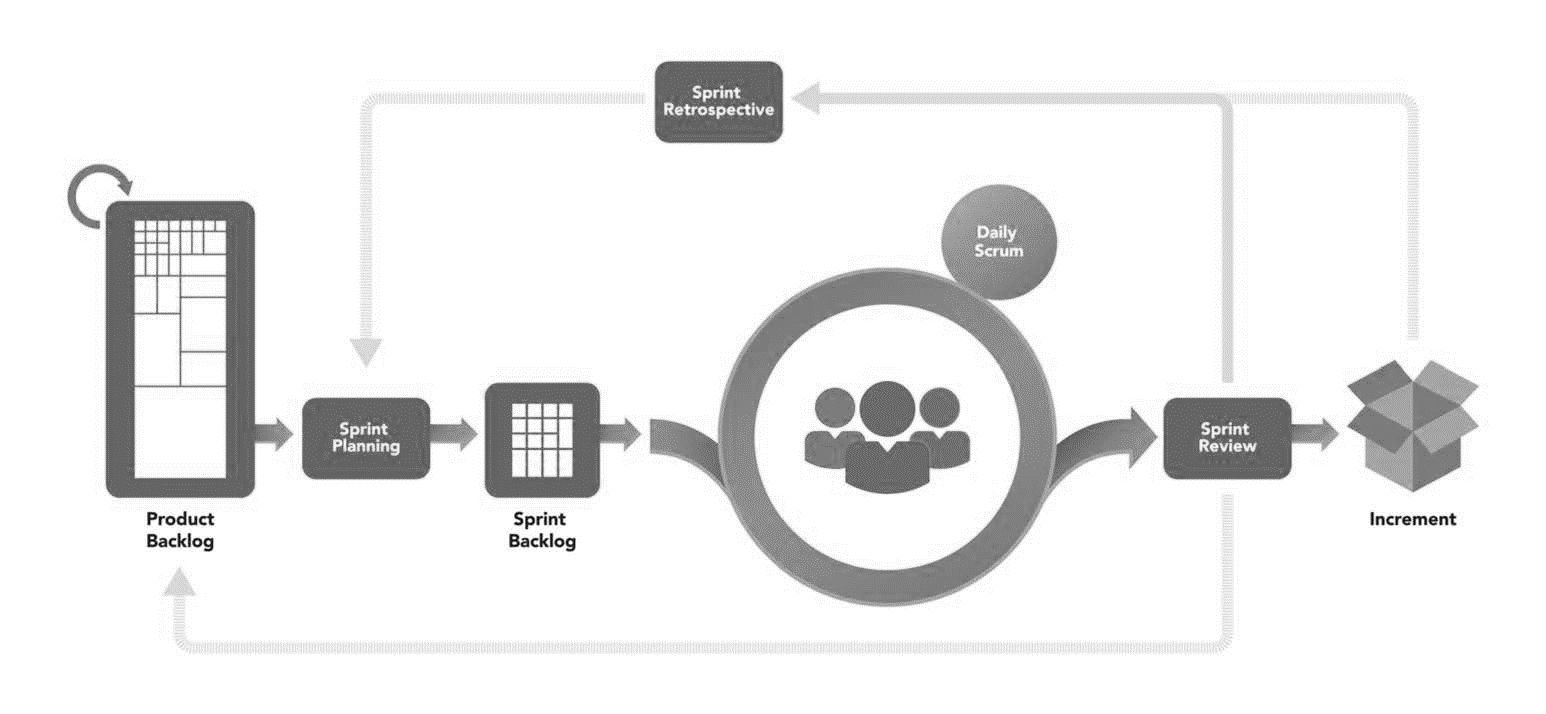
NASA Team I will employ the Scrum method. Scrum is an iterative and incremental software development method derived from Agile methodology. Agile development is “an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it”[[1]](#footnote-1)[[2]](#footnote-2). Scrum is a specific framework within Agile development that was first defined as "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal" as opposed to a "traditional, sequential approach" in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka in the "New Product Development Game". [[3]](#footnote-3) [[4]](#footnote-4)

The iterations in SCRUM are called Sprints. They are usually around 2-4 weeks in duration. The goal of each sprint is to be able to deliver to the customer a fully tested, production-ready, and shippable product. The emphasis is on the "shippable product" rather than the number of features. The product requirements are assembled in a "backlog" of user stories. Each sprint, a handful are selected to be included in the sprint.

Communication is vital to the SCRUM software development lifecycle (SDLC). To that effect, several meetings, and their maximum duration, are planned and coordinated throughout the sprint from the daily stand-up (to keep the entire team informed of work done, and potential hiccups) to the retrospective meeting at the end of the sprint (to evaluate the sprint and improve the team performance).

Another characteristic of SCRUM is the small self-organizing team. Usually, the team consists of six to nine people and three distinct roles:

* The product owner is responsible for what is delivered and represents the stakeholder’s interests and requirements
* The scrum master is a facilitator and addresses and removes impediments to the team.
* The development team (sometimes called the delivery team) is responsible for the actual delivery of the shippable product (sprint target).

*Figure 1: Sprint process of the scrum method [[5]](#footnote-5)*

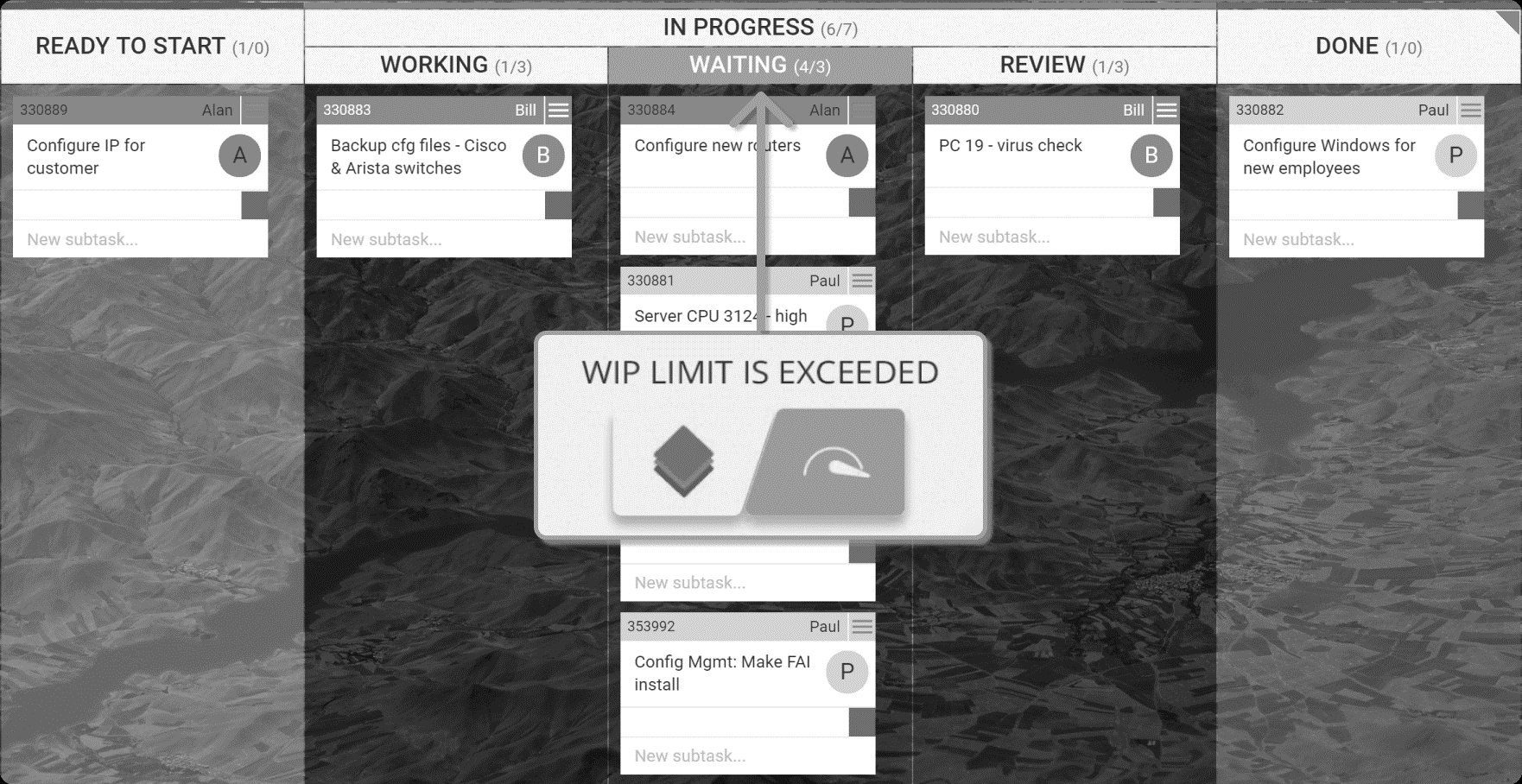
As illustrated in Fig. 1, A sprint in Scrum is restricted to limited work and duration. For NASA maestro software, each sprint shall include design, development, testing, and implementation. At the end of each sprint, the team will gather feedback from customers and managers and incorporate the feedback into the next iteration.

## Methods, Tools, and Techniques

To establish a uniform standard within the project team, the following tools will be utilized during project development.

### Development Collaboration

* Issue Tracking: Glow will be used initially for will be used to track project stories, issues, and tasks using a Kanban-style board[[6]](#footnote-6) of four lanes: Backlog, To Do, Doing, and Done. Glow integrates well with VS Code[[7]](#footnote-7) thus minimizing the need to switch back and forth between tools. Once the project is migrated to Gitlab the native Kanban boards will be used to keep all project assets in one basket.



*Figure B: Kanban example[[8]](#footnote-8)*

* Project Documentation: Google Docs is utilized to create project documents.
* Team Communication: Discord is used for messaging and voice conferencing and includes a voice channel for group conference calls and channels for general conversations, one for DevOps, UI Development, Testing, and Backend Development.
* Customer Communication: For typical communication, email is used. FreeConferences.com is used for voice conferencing.

### Software Architecture

This section describes the tools used for design and development to include planning, coding, and testing software.

#### Design and Analysis

This section details the tools used, analyses business requirements and to design the software to be developed. These include basic prototyping and documentation:

* Prototyping: Pencil will be used to create UIs of applications before actual coding work development starts.
* Technical documentation: Microsoft Word and Google Doc will be used to create and track technical documents, respectively.

#### Development

This section details the tools used for development and include versioning control, front and backend development and the overall development environment:

* Version control: Git will be used to control versions. GitHub will be used to host and track source codes. Eventually, the project will be moved to Gitlab as per customer request. Gitlab allows version tracking in addition to providing the CICD framework.
* Front-end development: Express.js is used to create a simple local web-based UI. The project does not require additional modifications to the Front-end
* Development environments: Visual Studio Code.
* Programming: The application uses JavaScript, Electron, and Node.js as the standard framework. Visual Studio Code (VSCode) will be the team’s default code editor and IDE. The additional functionality shall be developed in JavaScript and Node.js and hosted in a Docker Linux Container.

#### Software Verification and Validation

The whole process of software testing is categorized into 4 levels: unit testing, integration testing, system testing, and acceptance testing.

* Code Reviewing: Peer developers check source code to ensure the code is developed to published standards and follow established conventions.
* Unit Testing: white-box testing is done by developers and testers. The framework used is Mocha. The development shall follow a test-driven development methodology with the unit test cases and methods written before the actual application code is developed. Ideally, the code coverage shall be at least 90%.
* Functional Testing: This testing verifies that the applications developed to meet the requirements.
* Automated Regression Testing: Selenium will be used for regression testing to ensure newly developed code does not break existing functionality.
* Acceptance Testing: Acceptance testing will be performed by the customer representatives before final delivery

## Communication Model

This section details normal communication between customers, management, and the development team. Communication between all parties is handled using email and Zoom.[[9]](#footnote-9)

Weekly update emails highlighting progress within sprints and overall progress. Additionally, development teams are submitted in these emails.

Bi-weekly, at the end of a sprint, a meeting is held with the customer, management, and the development team. These meetings highlight prototypes and development during the sprint. Additionally, the development team supplies the customer with the focus for the next sprint and change can be submitted, by the customer, for integration in the sprint backlog.

## Project Change Plan

The customer suggests changes to the project plan during the bi-weekly post sprint meeting. The change is recorded initially as a verbal agreement. Following the completion of the meeting, the change is written into a change agreement that is confirmed by the customer through email.

Changes are integrated into the sprint backlog and prioritized based on the customers’ priority for the feature. Features changes are accepted and integrated as able and features that are not completed are fully documented for follow-on development.

# References

*Agile 101*. (2020). Retrieved from Agile Alliance: https://www.agilealliance.org/agile101/

*DEFINING TASKS AND ACTIVITIES - PROJECT MANAGEMENT*. (n.d.). Retrieved from Wisdom Jobs: https://www.wisdomjobs.com/e-university/project-management-tutorial-6/defining-tasks-and-activities-11622.html

Dennis, A. (2012). *Systems Analysis and Design.* Hoboken: John Wiley & Sons Inc.

K. Schwader, J. S. (2017, November). *The Scrum Guide.* Retrieved from Scrumguides.org: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

*Kanban Explained for Beginners*. (n.d.). Retrieved from Kanbanize: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban/

*Manifesto for Agile Software Development*. (2001). Retrieved from Agile Manifesto: http://agilemanifesto.org/

P. A. Permana, S. S. (2015). Scrum Method Implementation in a Software Development Project Management. *International Journal of Advanced Computer Science and Applications*, 6-9.

*What is a Scrum Master? The Role and Responsibilities*. (n.d.). Retrieved from Visual Paradigm: https://www.visual-paradigm.com/scrum/what-is-scrum-master/

*What is Scrum: An Introduction to the Scrum Framework.* (2018, April 4). Retrieved from Scrum.org: https://www.scrum.org/resources/what-scrum-introduction-scrum-framework

1. (Agile 101, 2020) [↑](#footnote-ref-1)
2. The Agile Manifesto and 12 Principles can be found at <http://agilemanifesto.org/>. [↑](#footnote-ref-2)
3. (K. Schwader, 2017) [↑](#footnote-ref-3)
4. (Dennis, 2012) [↑](#footnote-ref-4)
5. (What is Scrum: An Introduction to the Scrum Framework, 2018) [↑](#footnote-ref-5)
6. Kanban is a workflow management method designed to help you visualize your work, maximize efficiency and be agile. (Kanban Explained for Beginners, n.d.) [↑](#footnote-ref-6)
7. Visual Studio Code is a text editor with some Integrated Development Environment features available from Microsoft. <https://code.visualstudio.com/> [↑](#footnote-ref-7)
8. (Kanban Explained for Beginners, n.d.) [↑](#footnote-ref-8)
9. <https://zoom.us/> [↑](#footnote-ref-9)