Maestro Test Software

FEBRUARY 20

NASA Team 1

Authors

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

Revision History

Name	Date	Reason for Update	Version
Initial Document	02/20/20		1.0

Table of Contents

1. Introduction

1.1. Statement of Need

The customers of this project who work at NASA create procedures for spacewalks (EVAs). Typically, these procedures include two astronauts working outside the space station, plus 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 customers would like to add a new format for a Heads-Up Display (HUD). This would display just one actor's steps (EVA procedures include multiple people working in parallel) and would need to fit that display in a small screen size which may go inside an astronaut's helmet. Getting the basic format can mostly be achieved by copying existing output types.
This project will provide the following values for NASA Maestro software:

- 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
- 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.

- 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.
- 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)
- 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.)

1.2. Project Summary

1.2.1. Purpose

This software will be designed to ensure that .docx files created by Maestro are valid and then compare expected formatting of the provided YAML to a desired or expected output as a picture file. 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 YAML formatting.

1.2.2. Scope

This product will be developed to test Maestro and provide additional capability following the completion of document conversion. This includes the ability of the program to discover and highlight changes in the formatting and to display those changes made by Maestro developers.

1.2.3. Assumptions and Constraints

- The software needs to be system independent 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 typeface which is not provided within GNU Linux. The use of created container images should alleviate this concern.

1.2.4. Project Deliverables

1.3. Definitions and Abbreviations

Word	Definition

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

2. 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.

3. Project Organization

3.1. External Structure

[TODO]

3.2. 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.

3.2.1. Roles and Responsibilities

The project team consists of the following roles and responsibilities.

Name	Role	Responsibility	Allocation
Alberto Bonfiglio	System Developer/Engineer	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	Requirements gathering, budget, and final approval authority	200 hours
Tiezheng Yuan	System Developer/Engineer	Requirements gathering, System design, Development and Integration	200 hours

3.2.1.1. Project Manager

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 in accordance with all specifications. Furthermore, the PM organizes all group chat sessions, phase reports, and student-professor communications.

3.2.1.2. Systems Analyst

The software requirements analyst is 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.

3.2.1.3. System Developer/Engineer

The work responsibilities of system developers 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.

3.2.1.4. System Tester

Testers 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.

4. Managerial Process Plan

The managerial process plan emphasizes the work schedule and task management needed to meet the agreed-upon project goals.

4.1. Project Tasks and Activities

4.1.1. Major Tasks

- 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

4.1.2. Minor Tasks

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

4.2. Schedule

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

4.2.1. Deadlines

Task	Task Name	Duration	Start	Finish	Resource Names
Documentation	Project plan, Requirement Specifications	15	02/03/2020	02/23/2020	Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan
Design and Analysis	Use cases, software architecture, data modeling, process modeling	10	02/23/2020	03/07/2020	Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan
Development	The Scrum cycle including development, coding work, and testing	20	03/08/2020	04/05/2020	Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta

					Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan
Final phase	Implementation and project delivery	10	04/06/2020	04/26/2020	Alberto Bonfiglio, Beatrice Oluwabuyi, Jacquetta Reid, Kenya Foster, Rick Stuart, Tiezheng Yuan

4.2.2. Schedule Allocation

The figure below illustrates the project schedule in a Gantt chart view. This chart highlights the dependency mapping between tasks. The chart shows the major project milestones with delivery due dates for each project deliverable.
[TODO]

4.3. 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 shall make indispensable contributions to the project. ● 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 elicitation should be done with cooperation with customers ● Contextual Documents: All technological documents with regards to the design and implementation of the system or business logic on spacewalking will be open to all team members.
Software Tools	● Professional Software: All software should be available to all team members. Any data files should be accessed by other members.

4.4. 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 1 -5 scale with 1 being the lowest and 5 the highest probability of the risk occurring.

Description	Impact	Mitigation	Risk and Probability
Delay	Milestones are delayed, or project delivery is overdue.	[TODO]	[TODO]
Inadequate Scrum backlog and review	The scrum will be used in the project life cycle. Inadequate clarification of	[TODO]	[TODO]

	scopes, goals, or another management plan may lead to poorly productive in each Scrum cycle.		
Poor integration	This project is required to deliver five different testing components. If configuration and integration are done poorly, the system may suffer from unplanned downtime.	[TODO]	[TODO]
Poor access to end-to-end testing	Some end-to-end testing examples should be created in this project. Customers will access those testing cases. If poor access controls occur, the significance of this project will decline.	[TODO]	[TODO]
Scope creep	The scope for this project expands over its initial boundaries. New requirements	[TODO]	[TODO]

	are added.		
Invisible information	Project documents and progress are invisible to all team members.	[TODO]	[TODO]

5. 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.

5.1. Process Model

NASA Team I will employ the Scrum method. Scrum is an iterative and incremental software development method derived from the agile methodology. Scrum 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". [1] [2]

[1] (Schwaber & Sutherland, 2017)
[2] (Dennis, 2012)

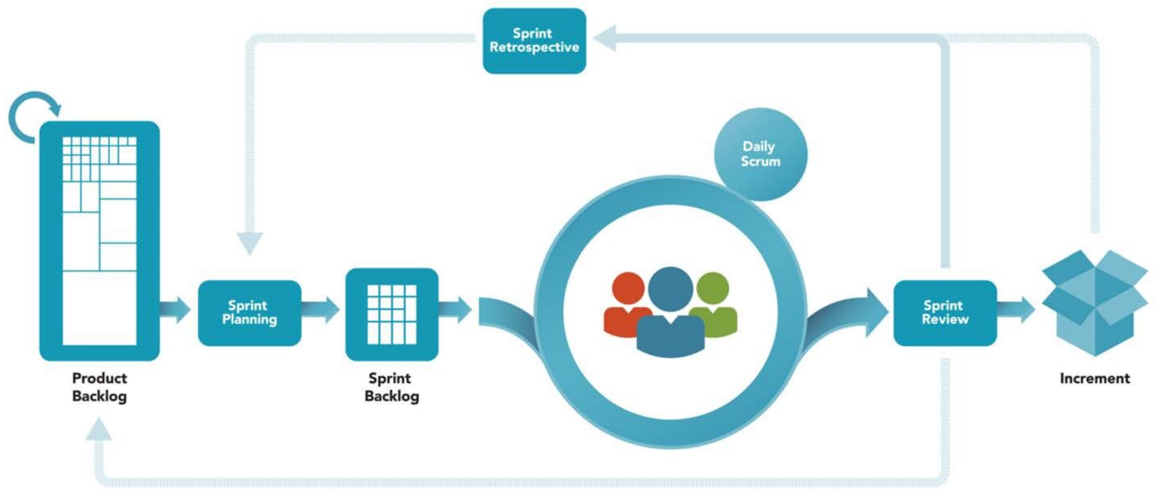

Figure 1: Sprint process of the scrum method [3]

In Scrum, the end-to-end testing work into 5 pieces. So, it is easier for team members to map out what needs to be done and achieve some pieces of work. It is easier to plan for short periods of time because less work is involved. Scrum divides NASA Team 1 into three smaller teams. It is easier for small teams to focus on a single goal.

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.

5.2. Methods, Tools, and Techniques

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

5.2.1. Development Collaboration

- Issue Tracking: [TODO]
- Project Documentation: Google Docs is utilized to create project documents.

[3] (What is Scrum: An Introduction to the Scrum Framework, 2018)

- Team Communication: Discord is used for messaging and voice conferencing.
- Customer Communication: For typical communication, email is used. FreeConferences.com is used for voice conferencing.

5.2.2. Software Architecture

5.2.2.1. Design and Analysis

- Prototyping: Pencil will be used to create UIs of applications before actual coding work development starts.
- Entity-relationship, context, and class diagrams: ArgoUML will be utilized to create those diagrams if necessary.
- Technical documentation: Microsoft Word and Google Doc will be used to create and track technical documents, respectively.

5.2.2.2. Development

- 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.
- Relational Database Management: N/A
- Front-end development: Java server is used to create a simple local web-based UI. The project does not require addition or modifications to the Front-end
- Development environments:

- Programming: The application uses JavaScript, Electron, and NodeJs as the standard framework. The additional functionality shall be developed in Javascript and nodejs and hosted in a docker linux container.

5.2.2.3. 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 follows established conventions.
- Unit Testing: JUnit white-box testing is done by developers and testers. This test will be used to create unit tests to test all single methods or objects of the code.
- 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.

5.2.3. Quality Assurance

[TODO]

References

Dennis, A. (2012). *Systems Analysis and Design*. (4th, Ed.) Hoboken: John Wiley & Sons, Inc.

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

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

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