**CS672-2405A-01**

Systems Engineering Management Plan

Ajitpal Grewal

November 10th, 2024

Week 5: Executive Summary

Dr. Pollack

# 2. Executive Summary

This Systems Engineering Management Plan encompasses all System Engineering Processes, Specialty Engineering, System Deployment, and Product Support methods involved in developing software that controls the actuators and telemetry data processing subsystems within a satellite. The software will adjust the satellite's trajectory accordingly and transfer recorded data to a ground central server. Components of this software include real-time operating systems and data processing algorithms. These components will be integrated to handle precise satellite positioning and accurate data manipulation (Chang, 2024). This project has 6-month development phases within its timeline, which include all testing and personnel training both pre- and post-deployment. The software will be designed such that it is compatible with pre-existing hardware to mitigate risks and minimize cost. Furthermore, the software must be resilient in the face of extreme aerospace environments. Using Agile sprint methodology, the system will support reliability, maintainability, safety, and security while adhering to *FAA* policies*.* GitLab, UML diagrams, Jira, and Functional Flow Block Diagrams ensure smooth handoffs throughout all phases of the CI/CD pipeline, starting with the Critical Design Review Board and concluding with the Safety and Mission Success Review Board. Integrated Logistics Support Engineers in collaboration with Senior Requirements Engineers will oversee piezoelectric actuator hardware and Linux-based Real-Time Operating System integration, installation, and checkout. Support Engineers shall aid the consumer throughout the project life cycle, applying relevant modifications, supervising updates, and ensuring the system is disposed of safely.

# 3. Document History

|  |  |
| --- | --- |
| **Date** | **Updates** |
| 10/13/2024 | Initial Release: Document Shell, Executive Summary |
| 10/20/2024 | Week 2: Purpose, Document Overview, System Overview, Project Schedule |
| 10/27/2024 | Week 3: Organization, Decision Making Process, SE Model, SE Processes |
| 11/3/2024 | Week 4: Specialty Engineering, System Deployment, Product Support |
| 11/10/2024 | Week 5: Executive Summary |

Table of Contents

[2. Executive Summary 2](#_Toc182162085)

[3. Document History 3](#_Toc182162086)

[5. Introduction 6](#_Toc182162087)

[a. Purpose 6](#_Toc182162088)

[b. Document Overview 6](#_Toc182162089)

[c. System Overview 7](#_Toc182162090)

[d. Project Schedule 8](#_Toc182162091)

[6. System Engineering Process 9](#_Toc182162092)

[a. Project Organization 9](#_Toc182162093)

[b. Environments 9](#_Toc182162094)

[c. Decision-Making Process 10](#_Toc182162095)

[d. System Engineering Model 11](#_Toc182162096)

[e. System Engineering Processes 12](#_Toc182162097)

[i. Configuration Management 12](#_Toc182162098)

[ii. Requirements Engineering 13](#_Toc182162099)

[iii. Functional Analysis 13](#_Toc182162100)

[iv. Design Processes 14](#_Toc182162101)

[v. Development Processes 14](#_Toc182162102)

[f. Verification 16](#_Toc182162103)

[g. Validation 16](#_Toc182162104)

[7. Specialty Engineering 17](#_Toc182162105)

[8. System Deployment 19](#_Toc182162106)

[a. Site preparation 19](#_Toc182162107)

[b. System installation 19](#_Toc182162108)

[c. System checkout 20](#_Toc182162109)

[d. User training 21](#_Toc182162110)

[e. Support engineer training 21](#_Toc182162111)

[9. Product Support 22](#_Toc182162112)

[a. Maintenance 22](#_Toc182162113)

[b. Logistics support 22](#_Toc182162114)

[c. Disposal 23](#_Toc182162115)

[10. References 24](#_Toc182162116)

# 5. Introduction

## a. Purpose

This Systems Engineering Management Plan outlines a blueprint for the development and deployment of satellite software capable of manipulating satellite trajectory through coordination of actuators. The software is also responsible for the processing and transmission of telemetry data collected by the satellite to a ground central server (Chang, 2024). The following information identifies the key software components of pre- and post- deployment that enable it to resiliently operate in various aerospace environments. The plan will ensure all software is successfully integrated with existing hardware through configuration management, functional analysis, verification, and validation while meeting project deadlines.

## b. Document Overview

The *System Engineering Process* includes Project Organization, Project Environments, the Decision-Making Process, the System Engineering Model, Project Verification, and Project Validation. Project Organization details the coordination methods of a multidisciplinary team and specifies individual responsibilities. All team members’ duties should align with mission objectives. Project Environments include the integration of hardware and software components within a satellite utilizing real-time operating systems (RTOS) and their ability to transmit sensitive data across vast distances. The Decision-Making Process ensures each phase of the project is resource efficient, reducing cost and mitigating risks (Blanchard, 2014). The System Engineering Model emphasizes techniques of Agile methodology for iterative and reactive development at each phase of the project life cycle (Paliwal, 2023). Further subsections of the System Engineering Process include Configuration Management, Requirements Engineering, Functional Analysis, Design Processes, and Development Processes. Configuration Management ensures successful implementation of the system throughout the entire system life cycle while allowing traceability between baseline configurations when appropriate changes are made. Requirements Engineering utilizes a top-down approach to impose guidelines for successful behavior of components within a system. Functional Analysis decomposes those behaviors and details their specific objectives. The Design and Development Processes incorporate modeling to ensure successful integration of hardware and software components in their respective environments. The Verification and Validation processes are checking/testing stages that confirm the system has met its essential requirements and is fully functional (Blanchard, 2014).

## c. System Overview

This Systems Engineering Project supports the advancement of technology in the aerospace sector. Relevant companies may safely and securely communicate their satellite observations using TCP/IP protocols without the risk of intrusion (Chang, 2024). The Linux-Based Real-Time Operating System implements several algorithms to ensure synchronous and parallel data processing while avoiding bottlenecks (Stallings, 2018). This document ensures the overall system adheres to consumer obligations, abides by FAA regulations, and is fully operational in extreme aerospace environments. The system should also be flexible such that it allows regular updates, granting longevity for high-intensity missions.

## d. Project Schedule

|  |  |
| --- | --- |
| **Project Phase** | **Date** |
| *Start of Documentation* | 10/13/2024 |
| *Alpha Test/Release* | 4/13/2025 |
| *Beta Test/Release* | 10/13/2025 |
| *Final Release* | 4/13/2026 |
| *Annual Update Version 1* | 4/13/2027 |
| *Annual Update Version 2* | 4/13/2028 |
| *Annual Update Version 3* | 4/13/2029 |

**Alpha Test/Release:** The first major step of the satellite software life cycle. Initial product goals for the internal development team are to debug the system in a controlled environment, applying patches.

**Beta Test/Release:** After internal tests the system is ready for consumer feedback. Trusted aerospace companies are granted system access, allowing performance and usability checks.

**Final Release:** This stage includes the stable release candidates of the satellite communication software. If final testing finds no issues within the system, the project is ready for deployment.

**Annual Updates:** Algorithms and communication protocols will be updated to align with mission objectives.

(Tucker, 2024).

# 6. System Engineering Process

## a. Project Organization

Operating in Silicon Valley, California, project headquarters is home to over 50 working teams consisting of astronauts, software/hardware engineers, team leaders specializing in multiple facets of the system engineering process (ex: Senior Project Requirements Engineer), and information technology/security specialists. Cross-functional communication between these multidisciplinary groups allows facilitation of directives and aligns specialist responsibilities with the mission statement. All mission objectives are carried out on-site, ensuring maximum security. The facility provides substantial resources for its contractors such as research and development laboratories with appropriate technologies and conference rooms.

## b. Environments

The project features a Linux-Based Real-Time Operating System that is to be embedded within a satellite. This RTOS integrates efficient software and hardware architectures that provide state of the art parallel processing capabilities using algorithms (ex: Round Robin, First-In-First-Out) that prioritize tasks to promote integrity of the system (Stallings, 2018). The following tools support these mechanisms:

***MATLAB, PyTorch:*** Within Visual Studio Code using Python, MATLAB offers extensive libraries engineers may utilize to compute and visualize complex models regarding satellite telemetry data (Arora, 2024). PyTorch implements machine learning to detect and predict movements of debris, and aids in satellite image processing (Stewart, 2022). Together, these tools improve actuator functionality by automating calculations for Telemetry Data Analysts before the information is handed off to appropriate Systems Engineers (Deiss, 2023).

***GitLab:*** Gitlab utilizes repositories to organize historical records, such as previous codes during Continuous Integration/Continuous Deployment (CI/CD) Pipelines, to streamline handoffs between software developers and mechanical engineers during each phase of the project life cycle (Reddy, 2024).

***JIRA:*** JIRA may collaborate with Structured Query Language (SQL) for debugging during field configurations and identifies permissions/workflows for relevant teams. By characterizing issue types and highlighting challenges for each phase of the project life cycle, the handoff process is efficiently coordinated amongst Verification and Validation (V&V) teams (Vartika, 2020).

## c. Decision-Making Process

***Trade Studies:*** Requirements engineers evaluate positive and negative results of interactions between project alternatives and possible states of nature using Decision and Compliance Matrices. Through the cardinal and quantitative expression of a Decision Matrix, a systems engineer may select the alternative that aligns with project requirements while maximizing profit, minimizing cost, and mitigating risks (Blanchard, 2014). Key Decision Point parameters during trade studies of satellite software deployment includes maintaining a $250 million budget for system design across a 6-month deployment timeline (Deiss, 2023).

***Control Boards:*** The *Critical Design Control/Review Board* is responsible for ensuring that the Concept of Operations (including coding practices, hardware framework, and subsystem integration) meets its mission requirements while considering constraints. The *Launch Readiness Control/Review Board* is responsible for testing, verifying, and validating the final design of the project prior to deployment. The *Post-Launch Assessment Control/Review Board* evaluates post-deployment activity of the system (Deiss, 2023). All indicators are documented to improve future missions.

***Working Groups & Ethical Considerations:*** Multidisciplinary working groups consisting of Quality Assurance teams, Requirements Engineering teams, and Information Security teams collaborate to ensure the project maintains strict adherence with guidelines set by relevant government agencies. The Streamlined Launch and Reentry Licensing Requirements Final Rule set by the *Federal Aviation Administration* regulates operations for all types of space vehicles (Federal Aviation Administration, 2021). The satellite software must comply with FAA policies to prevent interference and catastrophes.

*International Traffic in Arms Regulations* govern the transfer of sensitive information regarding defense-associated satellite software. This includes all component systems, such as actuators, involved in data exports. An organization that decides to implement these satellite services must comply with ITAR guidelines if the system is classified as a defense service (Blake, 2013).

## d. System Engineering Model

***Agile Methodology:*** This iterative, incremental project management framework allows system engineers to split system life cycle phases into smaller modules. Users can then improve upon each sprint with regular feedback loops. The flexible and adaptive capabilities of this model uphold consumer satisfaction because testing and development processes occur synchronously. Stages of this cycle include requirement gathering, design, development (coding), testing, deployment, and review. Scrum incorporates agile methodologies for complex, frequently changing systems such as satellite software development (Paliwal, 2013).

***Waterfall & “Vee” Process Models:*** The Waterfall model consists of 6 steps consisting of requirements analysis, specifications, design, implementation, test, and maintenance. A modification of this model divides hardware and software development into different phases, and each phase is fully completed until final product release. The downside of this model is that it only tests the system after implementation, therefore ignoring iterative reconstruction. It is challenging to use this model when requirements of the product are constantly changing, as is the case with satellite software development (Blanchard, 2014).

The “Vee” model begins with user needs and ends with a user-validated system. Steps of this model include defining system requirements, allocating system functions to subsystems, detailing and designing components, verification of components, verification of subsystems, and full system operation and verification. The downsides of this model are comparable to the Waterfall model in that iterative feedback is disregarded, and components of the system are not integrated prior to development (Blanchard, 2014).

## e. System Engineering Processes

### Configuration Management

The project utilizes Git for initializing repositories, staging changes, and committing changes. It aligns with the iterative Agile methodology, allowing for changes during each phase of the system life cycle. This distributed version control system simplifies traceability between handoffs for developers by logging all changes within its folders. Open-source characteristics streamline collaboration among developers during codebase modifications (Akt, 2024).

System configuration management involves identification and establishment of baselines for critical components (interfaces), version control and documentation of updates to artifacts, component change control to approve change requests, configuration auditing to formally review the technical correctness of modifications, and reports using release notes or configuration guides to provide accurate and up-to-date configuration data for stakeholders (Goel, 2018). This architecture will support the Continuous Integration/Continuous Development pipeline.

### Requirements Engineering

The Critical Design/Control Review Board determines system requirements based on consumer needs. The contracting company will provide a document outlining its mission endeavors and the board will dissect the project into its relevant phases prior to development. An updated version of this document detailing the critical component design concepts and project schedule is then relayed back to the buyer. Any adjustments will be made as the project life cycle progresses based on consumer feedback, upholding transparency.

### Functional Analysis

Based on requirements, mission critical components will be prioritized at the top level. These include actuator control interfaces, data transmission protocols, and telemetry data automation. Using a top-down hierarchy of functional flow block diagrams, high level components will be broken down into second and third-level functions. At these levels, actuator relevant codes are developed, data transmission protocols (TCP/IP) are implemented, and appropriate tools (PyTorch) that automate telemetry data are integrated with the system (Blanchard, 2014).

### Design Processes

*Actuator hardware* must be designed to survive in the extreme conditions of aerospace. Hardware engineers utilize CAD tools to simulate the effectiveness of *piezoelectric smart-material actuators* in such harsh environments. These actuators manipulate satellite telemetry when given electric charges (Allegranza, 2014). Software will dictate when these charges propagate.

*Software* design includes all relevant languages and applications used in actuator manipulation and data processing techniques, such as Python, MATLAB, and PyTorch. Using Functional Flow Block Diagrams, engineers can graphically represent the interaction and sequencing between the software (Blanchard, 2014). In terms of telemetry data processing, the diagrams depict the flow of information from reception, to filtering, to storage (Natural Resources Canada, 2024). Regarding actuator manipulation, engineers can represent the flow of commands from reception to execution.

### Development Processes

#### Software Development

Sprints within Agile methodologies enforce adaptability of the code during changing requirements of the system. Python logs within Git enable convenient data transfer and manipulation. Code readability is simplified using Unified Modeling Language Diagrams. UML diagrams allow software engineers to visualize the structure and interactions between subparts of a system. Data Analysts and Software Engineers can coordinate their findings with these diagrams, and efficiently store and process relevant telemetric data based on their relationships. Using classes and objects, teams can standardize languages across multiple disciplines, streamlining communication during resolution of issues throughout the project life cycle (Miro, 2024).

#### Hardware Development

The iterative hardware design cycle is composed of the payload & mission, actuator configuration, the command and data handling subsystem, the communications subsystem, the attitude determination & control system, the power sheet, the mass properties & distribution sheet, the thermal sheet, the model state sheet, the database sheet, and cost model. Physical models of subsystems are used to test key components prior to system integration. Using light and durable materials, hardware engineers can create resistant actuators (Mcinnes, 2001).

#### System Integration

A resilient system successfully integrates its hardware and software components. The combination of these components involves the use of fieldbus systems in coherence with Transmission Control Protocol/Internet Protocol. The Linux-Based Operating System using TCP/IP, and the fieldbus, is responsible for initiating actuator movement while transmitting processed data to a central ground server. These components must operate synchronously to coordinate communication cycles and allocate bandwidth (Schmeding, 2017).

#### Build Management

Jenkins supports the CI/CD pipeline as it allows developers to build, test, and deploy software. Using plugins, Jenkins may automate tasks sent by Git. Interaction of these tools in coordination with the objects and classes of the functional flow block diagrams creates a flexible open-source environment for both hardware and software engineers to collaborate their ideas (Khatiyan, 2021).

## f. Verification

Verification involves testing and results analysis/documentation. The Launch Readiness Control/Review Board establishes testing schedules and multiple beta deployments prior to final release. The board verifies that post-system integration models comply with regulatory standards set by the FAA and ITAR guidelines. Each beta test is analyzed and documented within a repository. These results include bugs, corrective course of actions, and version updates (Deiss, 2023).

## g. Validation

The Launch Readiness Control/Review board must also recognize stakeholder requirements. In discussions with the contracting company and the Critical Design Control/Review Board, the team aligns the mission requirements document with results from verification tests in the real world. If parameters indicate success, the system is ready for deployment. The Post-Launch Assessment Control/Review board evaluates system efficiency and receives feedback from the consumer. Every 6 months, the board documents weak points within the system, creating possibilities for annual improvements (Deiss, 2023).

# 7. Specialty Engineering

Software and hardware must be developed and integrated such that the system is able to survive in the extreme conditions of aerospace. The following key areas must be considered for successful pre- and post-deployment:

*Reliability:* A specialty engineer collaborates with specialists with multiple backgrounds to ensure reliability of the software/hardware integration. These backgrounds include orbital debris, space asset protection, and human space integration specialists (Deiss, 2023). These groups will conduct a Failure Mode, Effects, and Criticality Analysis (FMECA) across all development phases to ensure the system adheres to Technical Performance Measures Once these TPMs are defined, systems engineers will conduct further functional analyses, determine requirements allocation, identify causes of failure, determine effects of failure, identify failure detection means, rate failure mode severity/frequency/probability, and provide recommendations for system improvement (Blanchard, 2014).

*Maintainability:* This design characteristic regards ease, economy, safety, and accuracy of maintenance functions. This system should not delineate from system requirements throughout its 6-month deployment cycle. Maintenance should not be required prior to scheduled upgrades, and the maintenance cost should not exceed a $50 million budget.Identification of mission needs is the primary concern of maintainability. Specialty engineers regularly modify the software development architecture iteratively, based on emerging mission objectives. The Predictive Interactive Groundstation Interface supports constant upgrades (patches) and maintenance logs that assist in future design frameworks by locating recurring weaknesses (NASA, 2018). To minimize downtime, and ensure adequate corrective/preventative maintenance, all system restorations including fault detections, diagnostics, and component repair must be documented for future occurrences. Field technicians inspect, detect, and prevent possibilities of failures through software patches every 6 months (Blanchard, 2014).

*Safety:* A safety/hazard analysis is conducted every 2 months. These analyses include hazard description, hazard causes, identification of hazard outcomes, and hazard classification. Hazard conditions of software deployment within hardware in aerospace include acceleration and motion, radiation, pressure, extreme heat, and vibration. These hazards are categorized based on their effects. Negligible hazards (personnel error) may be disregarded. Catastrophic hazards such as equipment faults may cause complete system failure (Blanchard, 2014). To support integrity of the system, redundant components and automatic shutdown features are implemented and activated upon detection of errors. Detected errors will trigger immediate data backups to ground central servers to preserve the maximum amount of information collected.

*Security:* The aerospace industry is prone to data breaches and cyberattacks from malicious entities. To mitigate this risk, all operations regarding the system are conducted on-site with appropriate access controls. Such controls include Multi-Factor Authentication using fingerprints and voice recognition. Role-Based Access Controls will segregate disseminated information regarding the project on a need-to-know basis. Audits are conducted weekly to detect vulnerabilities in system security, and Intrusion Detection Systems integrated with dual firewall zones will prevent unauthorized access (Maymi, 2019).

# 8. System Deployment

## a. Site preparation

The Integrated Logistics Support engineer is responsible for verification of assets involved in transportation of system-related components to a preliminary site. For successful project transition, the receiving site is equipped to store, assemble, integrate, install, and maintain the system. A site survey is conducted early in the project life cycle alongside determining project requirements. The survey will include training for operators and maintainers, supporting final-product implementation. This system requires multiple transitional sites within the receiving site such as storage, instructional, and disposal site environments. A successful product transition mitigates electronic wiring damage, shock or stress damage, heat warping, and moisture overloads. Appropriate hooks and crates are secured to maximize ease and safety of loading and unloading components of hardware such as the piezoelectric actuators. The Linux-Based Real-Time Operating System is transported digitally (Deiss, 2023).

## b. System installation

End product transition involves quantification of Measures of Effectiveness to ensure the system is ready for installation. These MOEs regard safety, maintainability and reliability. Documents containing operations manuals and installation sequences are issued to systems engineers, post-validation of the product. Operational scenarios provide actions, stimuli, and other relevant information for engineers to consider multiple possibilities during system interaction. Such actions that are supported by the Predictive Interactive Groundstation Interface include data transmission, software patches, and mission planning (Deiss, 2023). Field Technicians will coordinate with Senior Project Requirements Engineers to arrange a data plan (allocate bandwidth), assemble the system, configure the satellite gateway, configure the sensor network, and connect to the system (Licor, 2024). The critical software device drivers controlling hardware functions include the CMOS Camera, Analog Converter, Real-Time Clock, Pulse Propulsion Unit, and Temperature Sensor (Dabrowski, 2005).

## c. System checkout

Post-installation, Quality Assurance Engineers collaborate with a Senior Requirements Engineer to review system function and identify hazards. Such reviews are Critical Design Reviews, Mission Definition/Readiness Reviews, Operational Readiness Reviews, System Integration Reviews, and Safety and Mission Success Reviews. These checks confirm customer requirements are met, and all documents are updated. Post-integration of software and hardware, the system is initialized, tested (hardware-in-the-loop), and verified multiple times to observe deviations from expected results. The Critical Design Control/Review Board and Launch Readiness Control/Review Board confirm the coherence of design schematics and layout with interconnection of Government-Supplied Equipment. Upon analysis of test-data, the system is either rolled back to previous phases, or is labeled as ready for deployment (Deiss, 2024).

## d. User training

End product documents include as-built design drawings, close out photos and training manuals (fabrication, operations) . Upon installation, users are provided with 6-week informational on-site training sessions for components involving human interactions (human-in-the-loop testing). Skilled users are granted access to testing facilities and gain familiarity with relevant tools regarding their specialization such as MATLAB, Pytorch, and the Predictive Interactive Groundstation Interface. Operators are also trained in safety procedures using simulated deployments that consider environmental factors and data logging. The Safety and Mission Success Review Board determines if user knowledge of the system is adequate and grants certification (Deiss, 2023).

## e. Support engineer training

Upon conducting a formal design review, Support Engineers collaborate with Design Engineers to appropriately solve interface problems and enhance compatibility within the system. Support Engineers will be onboarded to a technical team based on their skills to perform periodic detailed inspections, equipment modifications, and calibrations. By gaining understanding of design constraints, these specialists are responsible for determining effectiveness requirements and repair policies. These requirements and policies include transportation reliabilities, number of errors per line of code, and hardware configuration (Blanchard, 2014). Support engineers will analyze data collected by the Predictive Interactive Groundstation Interface to detect anomalies and conduct troubleshoots (NASA, 2018). Monthly training is required based on system errors and upgrades.

# 9. Product Support

## a. Maintenance

Each year for 3 years, Support Engineers in coordination with the Post-Launch Assessment Review Board will oversee autonomous updates allowing the satellite to perform corrective maintenance and “heal” itself. The satellite system will check for new versions, download the update using HTTPS, and store the new information. The system will then contain both the old and new versions of the software. Preventative maintenance allows failsafe mechanisms to activate in case of corruption, and the satellite may revert to its prior state. This framework will minimize downtime and maintain the integrity of the system (Badshah, 2023). These methods are integrated with NASA’s Space Communications and Navigation (SCaN) program, providing features for monitoring the diagnostics of satellite actuators (Walker, 2024).

## b. Logistics support

The logistics support architecture must be implemented throughout the project life cycle to minimize cost. This architecture includes supply chain actions to procure/acquire, manufacture/produce, and transport/distribute all system components to the user’s operational sites. This forward flow enhances consumer ability to sustain maintenance throughout the system life cycle. An effective forward flow meets consumer requirements from point of origin to point of consumption by considering logistics support at each project phase such as system requirements, system verification and validation, and system deployment (Blanchard, 2014).

## c. Disposal

Phase-out begins at the end of the system’s 3-year lifespan. The satellite will return to Earth at a landing site, and components will be classified as obsolete, phased out, or nonrepairable failed components. Obsolete components are no longer technically feasible (thrusters), some supporting components may be phased out due to age and emerging technologies (software), and nonrepairable failed components are artifacts discarded upon failure (actuators). Failed components are disposed after the conversion process for a non-environmental impact (Blanchard, 2014). Reusable components may be modified for new applications. Data that has been archived and analyzed may be disposed of once the information has been fully processed. The database management system is cleansed to support downloads for future missions. System closeout involves safe reentry and recovery in accordance with U.S. policy and international treaties (Deiss, 2024).

# 10. References

Akt, F. (2024, May 17). *What is Git?* GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-git/>

Allegranza, C. (2014). Actuators for Space Applications: State of the Art and New Technologies. In *ACTUATOR.* <https://theengineer.markallengroup.com/production/content/uploads/2014/12/Actuator_2014_ESA_Actuators_Space_Applications.pdf>

Arora, N. (2024, June 14). *MATLAB For Data Visualization: Creating Effective And Engaging Charts And Graphs - GeeksProgramming*. GeeksProgramming. <https://geeksprogramming.com/matlab-for-data-visualization/>

Badshah, A. (2023, August 5). Over-the-vacuum Update – Starlink’s Approach for Reliably Upgrading Software on Thousands of Satellites. *37th Annual Small Satellite Conference*.

Blake, C. (2013). *SSC13-V-8 Navigating Export Controls and Regulations for Small Satellites*. <https://s3vi.ndc.nasa.gov/ssri-kb/static/resources/Navigating%20Export%20Controls%20and%20Regulations%20for%20Small%20Satellites.pdf>

Blanchard, B. S. (2014). *Systems engineering and analysis.* Pearson.

Caldwell, S. (2024, June 19). *Space Mission Design Tools - NASA.* <https://www.nasa.gov/smallsat-institute/space-mission-design-tools/>

Chang, E. (2024, March 26). *Satellite Communication Protocols and Ground Stations – Telecomworld101.com.* Telecomworld101.com. <https://telecomworld101.com/satellite-communication-protocols/>

Dabrowski, M. (2005). *THE DESIGN OF A SOFTWARE SYSTEM FOR A SMALL SPACE SATELLITE*. https://lunarcubes.wordpress.com/wp-content/uploads/2014/09/satellitesoftwareion.pdf

Deiss, H. (2023). *SEH 2.0 Fundamentals of Systems Engineering - NASA.* <https://www.nasa.gov/reference/2-0-fundamentals-of-systems-engineering/>

Deiss, H. (2024, August 12). *SEH 5.0 Product Realization - NASA*. Nasa.gov. https://www.nasa.gov/reference/5-0-product-realization/

Goel, A. (2018, May 11). *Software Engineering | System configuration management*. GeeksforGeeks. <https://www.geeksforgeeks.org/software-engineering-system-configuration-management/>

Khatiyan, S. (2021, March 7). *What is Jenkins?* GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-jenkins/>

Licor. (2024). *Satellite Communication | Initial setup*. Licor.com. https://www.licor.com/env/support/Com-Satellite/topics/initial-installation.html

Maymí, F. (2019). *CISSP exam guide* (8th ed.). Mcgraw-Hill Education.

Mcinnes, A. (2001). *SSC01-VI-5 1 A Systems Engineering Tool for Small Satellite Design*. <https://s3vi.ndc.nasa.gov/ssri-kb/static/resources/A%20Systems%20Engineering%20Tool%20for%20Small%20Satellite%20Design.pdf>

Miro. (2024). *What is a UML diagram, and what is it used for? | Miro*. Https://Miro.com/. <https://miro.com/diagramming/what-is-a-uml-diagram/>

NASA. (2018). *Mission Control Software Manages Commercial Satellite Fleets | NASA Spinoff*. Nasa.gov. https://spinoff.nasa.gov/Spinoff2018/it\_9.html

Natural Resources Canada. (2024, March 5). *Tutorial: Satellite Data Reception*. Canada.ca. <https://natural-resources.canada.ca/earth-sciences/geomatics/satellite-imagery-and-air-photos/satellite-imagery-and-products/educational-resources/tutorial-satellite-data-reception/9611>

Paliwal, P. (2023, December 13). *What is Agile Methodology?* GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-agile-methodology/>

Reddy, S. (2024, April 21). *What Is Gitlab? Complete Guide*. GeeksforGeeks. <https://www.geeksforgeeks.org/gitlab/>

Schmeding, H. (2017, August 21). *How to Integrate Valve Actuators with an Automation System*. Isa.org; ISA. <https://blog.isa.org/integrate-valve-actuators-automation-system>

Stallings, W. (2018). *Operating Systems: internals and design principles* (9th ed.). Pearson Education Limited.

Tucker, R. (2024, March 7). *The Lifecycle of Software Releases Explained*. Split. <https://www.split.io/blog/lifecycle-software-releases-explained/>

Vartika, O. (2020, March 5). *Introduction of JIRA*. GeeksforGeeks. <https://www.geeksforgeeks.org/introduction-of-jira/>

Walden, D. D. (2015). *Systems engineering handbook : a guide for system life cycle processes and activities* (4th ed.). Wiley.

Walker, J. (2024, June 13). *Space Communications and Navigation (SCaN) Program - NASA*. Nasa.gov. https://www.nasa.gov/directorates/space-operations/space-communications-and-navigation-scan-program/