18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 262

Open Source Community-Built Software Standards for Space and Ground Software. An Overview of the Software Lifecycle Improvement & Modernization (SLIM) Project from NASA's Jet Propulsion Laboratory

Rishi Verma[a], Kyongsik Yun[a], Paul M. Ramirez[a]

[a] *Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, 91108 Pasadena, CA USA*

Abstract

The Software Lifecycle Improvement & Modernization (SLIM) initiative at NASA's Jet Propulsion Laboratory (JPL), sponsored by the NASA's Advanced Multi-Mission Operations System (AMMOS) effort, represents an advancement in the rapid infusion of software development best practices for space and ground software. It is a shared, open-source resource for discussing, iterating, and referencing best practices for software development. It draws on strategic insights from NASA JPL's internal and external open-source initiatives, and it has been field- tested widely at JPL across hundreds of software project repositories. SLIM champions the development, iteration, and widespread dissemination of standardized software development best practices and is aimed at collectively maturing sub-topics such as project governance, software lifecycle automation, and software documentation through a participatory and open community model. SLIM's methodology harnesses a "standards-as-code" philosophy, where best practices are not only developed openly through community contributions and feedback but are patchable directly into project codebases or deployable via infrastructure enhancements, including using generative artificial intelligence techniques. This unique approach allows for the scalable dissemination of improvements makes a tangible impact on space-related software project performance and significantly reduces infusion barriers for software user, developer, and stakeholder communities. SLIM's best practices, encompassing artifacts like templates, workflow automation, or software starter kits are open-source licensed and free to use and iterate. We explore the SLIM initiative in detail, focusing on its three-phase process for infusing best practices: soliciting input and feedback from a community of projects, developing actionable standards encodable as patched improvements, and directly pushing these improvements into active project codebases through software pull requests and issue tickets. We also highlight the community-driven aspect of SLIM, discussing the network of contributors who are pivotal in curating and advancing the SLIM repository of best practices, as well as the results of our overall process.

Keywords:

Software development, best practices, open-source, NASA, standards-as-code, artificial intelligence

1. Introduction

Software development for space missions presents challenges that require standards and best practices to ensure maintainability. The Software Lifecycle Improvement & Modernization (SLIM) project at NASA's Jet Propulsion Laboratory (JPL), sponsored by NASA's Advanced Multi-Mission Operations System (AMMOS) effort, addresses these challenges through a uniquely positioned approach.

SLIM represents an advancement in the rapid infusion of software development best practices for space and ground software systems. Operating as both a community of contributors as well as a continuously evolving repository of best practice documentation, it provides open-source resources under a permissive license that any individual anywhere can use and contribute to. The project brings two key innovations to the state-of-the-practice in software quality improvement: first in conducting improvements through community consensus and in an open source setting, and second using a "standards-as-code" automation strategy that accelerates real-world infusion. For the first innovation, a shared platform for discussing and iterating best practices in software development has been developed, drawing on insights from NASA JPL's open-source software projects. Second, SLIM employs its "standards-as-code" methodology where best practices are developed in a way that makes them directly patchable to existing or new community member project codebases or infrastructure. This approach enables the initiative to scale best practice dissemination widely to many projects simultaneously and in an automated way, which significantly reduces barriers to adoption, as will be discussed. For best practices that cannot be directly patched to repositories, contributors develop automation that can be run as scripts or commands, deploying these to through issue tickets that can then be relayed to infrastructure by respective team members. The "standards-as-code" approach, which is reminiscent of the "infrastructure-as-code" principle, is discussed more in depth in the *Section 3.1 "Standards-as-Code".* Additionally, the project uniquely functions through an open source participatory model where practitioners from across JPL (and more widely) can contribute to the development and refinement of best practices. This community-centered approach

includes both contributors who provide input to the project and community member projects that receive and implement best practices. This, in effect, consolidates the practice of improving software processes, helping to improve the bottom-line for many projects together rather than each project focusing on improvement on its own. Additionally, the community-oriented approach of SLIM helps drive engagement, interest, and participation in improving software – helping make it a "team-sport" that people are more inclined to not only participate in, but extract meaning from. The project has been field-tested extensively at JPL across hundreds of software project repositories, spanning mission-centric projects, research projects, and open source projects. This wide-scale deployment has helped to validate strategy. Metric-based tools, as will be discussed in *Section 6.1 Implementation Across JPL*, additionally help validate the real-world impact of the initiative on projects.

This paper explores how SLIM assists software projects in creating and infusing best practices in a consensus-driven way and through automation. By leveraging open-source best practices and the "standards-as-code" philosophy, it enables infusion of best practices that can be directly integrated into existing project codebases, helping to make a tangible impact on space-related software projects.

2. Background

2.1 Challenges in Space Software Development

Space and ground software development faces hurdles that make the adoption of best practices in software challenging. For instance, the tight staffing and schedules sometimes preclude the adoption of software best practices in a thorough way. Additionally, in some cases staff are simply not trained or incentivized to leverage software best practices that are prevalent in industry. This is problematic, especially for space software, where software failures can lead to significant mission impacts or even catastrophic outcomes. Ensuring consistent application of software best practices across the diverse projects and teams in the space industry therefore is a unique problem that must be addressed. Traditional approaches to software best practice infusion often involve static documentation or recommendations that quickly become outdated and fail to adapt to evolving technologies and methodologies. Space software development is characterized by reliability requirements, where systems must operate correctly for extended periods with minimal or no human intervention. These systems typically have long mission lifetimes, sometimes spanning decades, during which the software must remain maintainable and adaptable to changing user needs. Additionally, space software development involves diverse stakeholder needs, including scientists, engineers, operators, and users, each with their own expectations. Balancing these diverse needs while maintaining system functionality also presents a significant challenge for development teams. Together, these challenges collectively underscore the need for standardized, robust, and adaptable software development best practice development and infusion.

2.2 The Open Source Way

NASA JPL has increasingly embraced open-source methodologies both internally and externally, marking a significant change in how software development is conducted. This shift has demonstrated the value of collaborative development and transparency. SLIM builds upon these values, applying open-source principles to the development of the best practices themselves. The decision to make the project itself open-source through a public Apache Version 2 license agreement aligns with NASA's broader commitment to transparency and knowledge sharing. By making best practices publicly available, it enables not only JPL projects, but the wider NASA and space software development to benefit from collective expertise. The open-source approach represents a natural progression of this evolution. Rather than simply sharing recommendations, tutorials, step-by-step guides, and infusible software assets are all shared and in effect, reduce barriers to adoption to ensure a wide spectrum of software developers and managers can follow along. With the project itself being open source, the opportunity for a community focused on helping to define, improve, and disseminate the best practices also comes into existence. It, therefore, creates a platform where standards can be exchanged, evolve and improve based on real-world implementation experiences across diverse projects.

2.3 Need for Standardized Yet Adaptable Practices

While standardization is crucial for consistent quality across projects, space software also requires adaptability to accommodate mission-specific requirements. SLIM addresses this challenge by creating standardized practices that can be customized and extended through community contributions. The fundamental challenge in space software development is balancing the need for organizational standards with the flexibility required to address unique user needs. Overly rigid standards can stifle innovation and prevent teams from addressing specific challenges, while completely customized approaches for each project lead to inconsistency, duplication of effort, and difficulties in collaboration across projects. It navigates this balance by providing a foundational set of best practices that establish

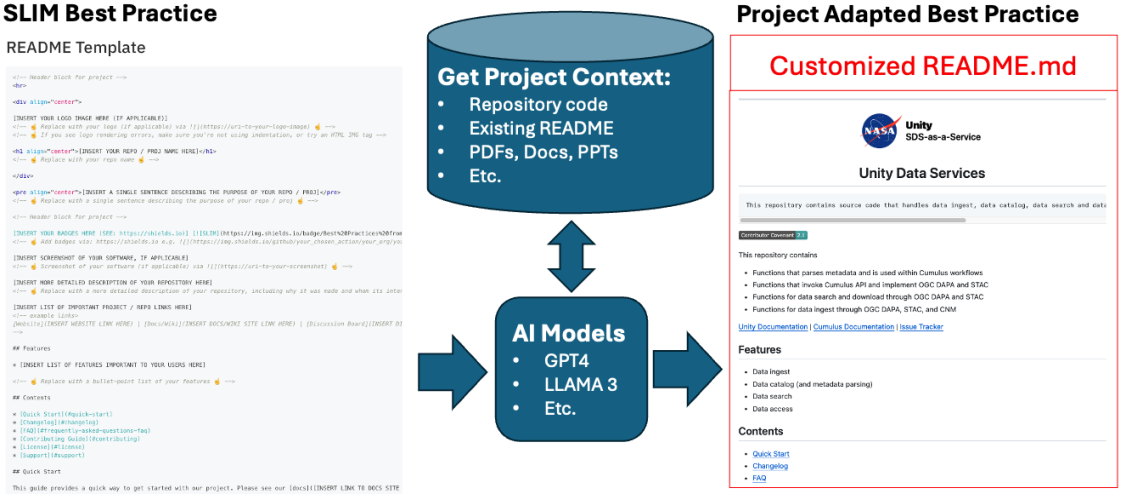
a common baseline for quality and reliability, while allowing for customization and extension based on specific project requirements. Each project has the opportunity to adapt part or all of a best practice standard, as well as to tailor to their specific needs. This dissemination and adaptation approach is discussed more in *Section 4.3 Phase 3: Infusion*. This approach ensures that critical aspects of software development—such as security, testing, and documentation—meet consistent standards across projects, while still providing the flexibility needed to address unique mission needs. The adaptability of SLIM's practices is enhanced by its community-driven development model. As software projects infuse these best practices, they also have the opportunity to contribute back improvements based on real-world experience. This continuous refinement ensures that the practices remain relevant as technologies and real-world conditions evolve. Moreover, its "standards-as-code" approach, discussed in the next section, inherently supports adaptability, as the practices are encoded in a form that can be directly applied to project codebases and modified through the software pull request process.

3. Overview

3.1 "Standards-as-Code"

The key driver of SLIM's best practice infusion process is the "standards-as-code" philosophy, which represents a shift in how software development best practices are conceptualized, iterated, and disseminated. This approach treats best practices not as static documentation but as living, executable assets that can be directly infused to software projects in a version-controlled way – similar to software components that are the target of SLIM itself. The "standards-as-code" philosophy means that best practices are developed in forms that can be directly patched to existing or new community member project codebases or deployed to their infrastructure. Readers may be familiar with the software methodology of "infrastructure-as-code" – where instructions to help deploy, install, or execute software in a target environment are themselves encoded as code. Infrastructure-as-code has become an industry success story, as several popular frameworks like Puppet, Chef, or Ansible have emerged to implement this philosophy. In a similar way, the "standards-as-code" approach helps best practice recommendations transform into concrete, actionable improvements that can be automatically applied across projects. For example, instead of simply documenting that projects should have a structured README file, SLIM provides a template that can be directly integrated into repositories. Similarly, rather than simply recommending security checks, it offers pre-configured scanning tool configurations that projects can adopt with minimal effort. Moreover, it leverages generative artificial intelligence (AI) to even fill out the recommended README based on the custom contents of the software codebase. This AI capability has been a popular infusion approach, and we anticipate this capability becoming more mature as the field advances. Ultimately, the generative AI capabilities may represent a majority of the infusion approaches proposed by SLIM. The power of this approach is illustrated below (Fig. 1) with an example of a README. A consistent, yet project agnostic README template is automatically transformed into a customized README for a project, by way of scanning the project's repository code to understand context.

Figure 1: transformation of a template into a customized artifact automatically



"Standards-as-code" is applicable to every part of the software lifecycle, and to all of the key categories of best practices that it targets, including software lifecycle best practices, documentation / information sharing, and governance. This approach also allows us to scale out best practice dissemination widely to many projects at once,

significantly reducing the barriers to adoption. The traditional challenges of implementing best practices – such as interpreting guidelines, developing custom solutions, and expending a high amount of effort and time – are instead resolved by providing ready-to-use implementations that can be directly infused into existing software projects and processes quickly and easily. For best practices that cannot be directly patched to software repositories, contributors often develop automation that can be run as scripts or step-by-step recipes that can be followed, and deploy these to community member projects through issue tickets. This comprehensive approach ensures that improvements can be efficiently infused at scale across diverse projects, regardless of their specific technical environments or constraints. The "standards-as-code" philosophy also enables continuous improvement of the best practices themselves. As with code, these practices can be versioned, allowing projects to, in the future, track which standards they have adopted and to systematically update to newer versions.

3.2 Key Focus Areas

SLIM focuses upon three key domains of software development, providing a comprehensive framework for improving software quality across the entire development lifecycle. These domains – project governance, software lifecycle automation, and software documentation – each play a critical role in the success of software projects, particularly in the complex and demanding context of space mission software.

3.2.1 Project Governance

SLIM provides standards for project organization, contribution workflows, and decision-making processes that promote transparency and collaboration. Effective governance is essential for project success, particularly as teams grow and projects mature. The governance best practices establish clear structures that facilitate coordination while maintaining focus on project goals. For example, we offer governance models (GOVERNANCE.md files) that are tailored to different team sizes, from small teams (1-3 active members) to large teams (10+ active members). This scalability ensures that projects can adopt appropriate governance structures at each stage of their evolution, adapting as their needs change. In addition to governance models, we provide templates for codes of conduct, contribution guides, issue templates, and pull request templates. These best practices help projects establish clear expectations and processes for community interaction and contribution, facilitating collaboration and reducing potential conflicts.

3.2.2 Software Lifecycle

SLIM promotes best practices for continuous integration, testing, deployment, and other automation processes that enhance software reliability and efficiency. Automation is particularly crucial in the space domain, where software failures can have significant consequences and where rigorous validation is essential. We provide guidance on choosing appropriate tools for various aspects of the software lifecycle, from code analysis and testing to deployment and release management. This helps projects navigate the complex ecosystem of development tools and select solutions that meet their specific needs while aligning with industry best practices. SLIM offers best practice guides as well as infusible assets for tasks such as setting up automated testing to establishing security scans. The infusible assets include configuration templates, workflow definitions, and scripts that projects can adapt to their specific environments. By automating key processes, projects can ensure consistent quality checks, reduce manual errors, and accelerate development cycles. It also provides starter kits for different programming languages, such as the Python Starter Kit, which exemplifies an integrated approach to building, releasing, and publishing software using GitHub's CI/CD automation. These starter kits serve as comprehensive templates that projects can use to quickly establish a fully functional development environment with best practices already incorporated. Additionally, it addresses security concerns by providing guidance on implementing container vulnerability scanning, enabling GitHub security features, and setting up secrets detection to prevent sensitive information from being inadvertently committed to repositories. These security-focused automation practices help projects identify and address vulnerabilities early in the development process, enhancing overall system reliability.

3.2.3 Documentation & Information Sharing

SLIM offers templates and tools for creating comprehensive, maintainable documentation that serves the needs of diverse stakeholders. Effective documentation is essential for software usability, maintainability, and knowledge transfer, particularly in the space domain where projects often have long lifespans and changing personnel. SLIM's documentation resources help projects establish clear and consistent guidance for users, contributors, and maintainers. By following documentation templates, projects can ensure that critical information is communicated effectively and maintained over time, reducing the risk of knowledge loss and facilitating onboarding of new team members. It provides templates for essential project documents like README files, change logs, and documentation hosting recommendations. These templates help ensure that documentation is focused on user needs as well as the

software development community. The README template, for example, guides software project teams in creating an informative introduction to their software that communicates the project's purpose, usage guidelines, and contribution process. SLIM also offers guidance on documentation hosting options, helping projects select platforms that best serve their specific needs. This includes trade studies of different documentation hosting tools and frameworks, comparing features like search capabilities, version control integration, and collaborative editing. Furthermore, it provides recommendations for documenting different aspects of a project, from API references to user guides, ensuring comprehensive coverage of all information that stakeholders might need. This holistic approach to documentation helps projects create a cohesive information ecosystem that supports users, contributors, and maintainers throughout the software lifecycle.

3.3 Community Participation

SLIM operates through a collaborative and participatory model where practitioners from across JPL, NASA, and beyond can contribute to the development and refinement of best practices. This community-driven approach ensures that standards remain relevant and practical, drawing on the collective expertise and experience of diverse people. The community includes both contributors who provide input to the project (guidance, code, issues, documentation, etc.) and community member projects that receive and infuse best practices. This dual structure creates a virtuous cycle where best practices are both developed collaboratively and tested in real-world environments, leading to continuous improvement and refinement. Community member projects have representatives who interface with the initiative, receive pull requests or issues containing best practice recommendations, and sometimes contribute back to the SLIM project itself. This bidirectional relationship ensures that best practices are not only disseminated but also enriched through practical implementation experience. Current community participants include a diverse range of NASA projects, such as projects from the Advanced Multi-Mission Operations System (AMMOS) Multi-mission Ground System and Services (MGSS) [1], Hybrid Science Data System (HySDS) project [2], Observational Products for End-Users from Remote Sensing Analysis (OPERA) project [3], Planetary Data System (PDS) Engineering Node (EN) [4], and Unity Science Data System (SDS) project [5], and more. This broad participation demonstrates the applicability of best practices across different types of space-related software projects and provides a rich ecosystem for testing and refining these practices. The community model also includes clearly defined roles and contribution paths, from users who deploy the best practices to contributors who actively participate in their development and refinement. This structured approach helps organize community participation while ensuring that all contributions align with overall goals and quality standards.

The governance structure, including technical and project steering committees, provides oversight and direction for the community, ensuring that the project maintains focus on its core goals while remaining responsive to community needs and input. The structured governance and open participation help create a framework for ongoing development and improvement of best practices. By fostering a vibrant community around software best practices, the initiative is creating a collaborative environment where knowledge and experience are shared across projects, organizations, and even across the broader space software community.

4. Methods

SLIM employs a structured three-phase process (Fig. 2) for developing and implementing best practices across JPL projects. This approach ensures that best practices are not only sound but also practically applicable and easily adoptable. The process begins with soliciting community input on needs, proceeds through the development of actionable solutions, and culminates in the direct infusion of these standards into active software projects.

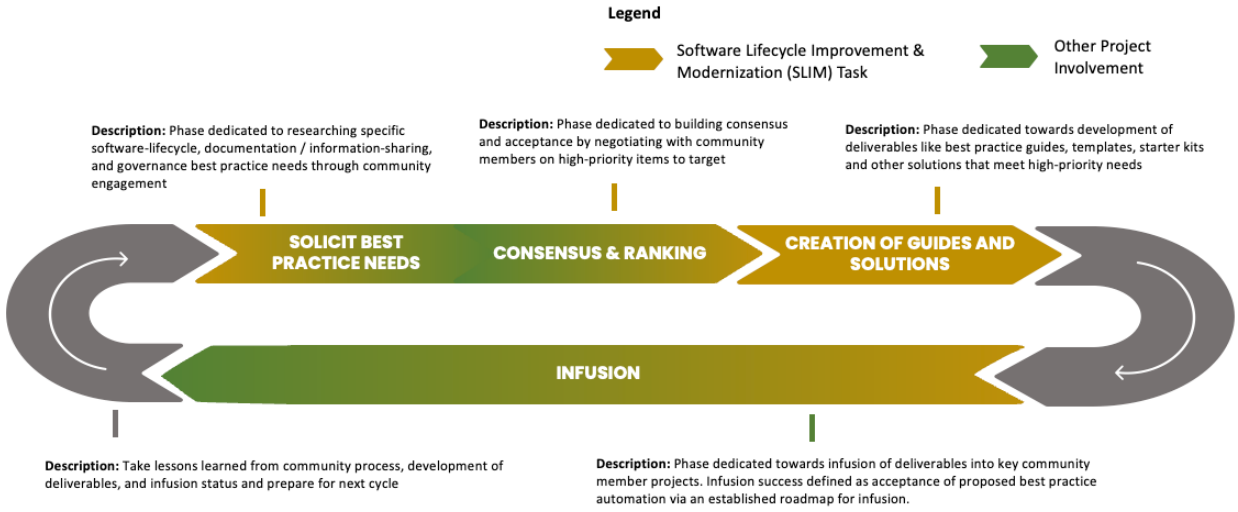


Figure 2: Key phases of the methodology behind SLIM. First, solicitation and ranking of best practice needs. Second, creation and development of solutions. Third, infusion of best practices to projects.

4.1 Phase 1: Community Input and Iteration

The first phase of the process involves gathering input from the community of software projects at JPL. This collaborative approach ensures that efforts address the most pressing needs of the community, maximizing the impact of developed best practices. SLIM conducted a widely disseminated poll in 2022, that helped identify over fifty best practice needs from the JPL community. These have since been encoded as issue tickets within the SLIM project GitHub repository [6].

4.1.1 Planning Board

SLIM reaches out to community member projects and solicits input on outstanding process improvement needs, including their relative ranking in importance and criticality. From these identified needs, a subset is chosen for focused development, with the aim of creating best practice guides that include use case lists, trade studies, reference architectures, and starter kits. This prioritization helps ensure that limited staff are directed toward the most impactful improvements, as determined by the community itself. The community-ranked list of best practice ideas is maintained in a planning board, which is publicly accessible through the SLIM GitHub repository [7]. This transparency provides visibility into priorities and progress, allowing community members to understand how their input is being incorporated and when they can expect specific improvements to be available.

4.1.2 Communication Channels

SLIM employs a variety of methods to gather feedback from the community, creating multiple avenues for participation. Repository issue tickets serve as a primary channel for collecting specific requests, suggestions, and bug reports related to existing best practices. These tickets provide a structured way to capture and track community input, ensuring that no valuable feedback is lost. Discussion forums and Slack communication channels offer spaces for more open-ended conversations about best practices, challenges, and potential solutions. Regular, bi-weekly community meetings provide opportunities for synchronous discussion and collaboration, allowing for deeper exploration of complex topics and immediate feedback on proposed approaches. These meetings help build a sense of community and ensure that diverse perspectives are considered in the development of best practices.

4.1.3 Community Contributions

SLIM establishes clear paths for community members to contribute new best practices or suggest improvements to existing ones. External contributors are free to propose best practice guides to the project at any time, following the contribution guidelines outlined in the documentation [8]. For those interested in submitting best practice guides to the SLIM project, a structured process is in place. This begins with creating or selecting an issue ticket to avoid duplication of work and to communicate ideas with the community. Since not every best practice solution is appropriate, community consultation is encouraged before making pull requests. Contributors are encouraged to create draft pull requests rather than submitting finalized guides, allowing for iterative development with community

feedback and contributions. This approach ensures that best practices are refined through collaboration before being finalized, resulting in more robust and widely adopted best practice solutions.

4.2 Phase 2: Development of Actionable Best Practices

The second phase focuses on transforming community input into concrete, actionable standards that can be directly implemented in software projects. This phase is where the "standards-as-code" philosophy is most prominent, as best practices are developed in forms that can be directly infused into projects.

4.2.1 Developing Best Practices Guides and Infusion Assets

As was mentioned in *Section 3.1 "Standards-as-Code",* best practice standards are developed in forms that can be directly infused into projects, such as configuration files, templates, or scripts. The contributing community is encouraged to ensure that every best practice guide is embedded with actionable “standards-as-code” assets that can be readily deployed or patched to software code repositories. SLIM includes guidance for helping existing and new contributors to develop best practice guides that include both prose as well as infusible asset material [9].

When contributors develop submissions for the NASA-AMMOS/slim repository, they are guided to focus on three primary aspects of the development process. First, contributors are encouraged to emphasize automation in their best practice solutions. By presenting solutions as templates, software automation tools, or starter kits, they make adoption easier for end users. This approach facilitates smoother implementation and increases the likelihood of successful adoption within the community. Second, contributors are required to follow the established folder structure conventions of the repository. A new folder is expected to be created for each guide within the appropriate sub-folder in the docs/guides directory. Any infusible assets, such as templates or code, are expected to be stored in the /static/assets directory within a relevant sub-category. Third, contributors are expected to utilize the standard guide template provided by the project to ensure uniformity and comprehensibility of all guides. This template offers a structured format that begins with a title and concise description, followed by sections for introduction, prerequisites, a step-by-step guide with optional images, an FAQ section, and appropriate credits. Finally, contributors register their guides by adding an entry to the shared SLIM registry, which ensures the guide appears in the website’s search page and can be automated through the slim-cli tool – described more in *Section 4.3.1 Pull Request Methodology*. By following these guidelines, contributors can help maintain consistency as well as reduce infusion barriers.

4.2.2 Quality Assurance

Rigorous review and testing ensure that developed standards meet quality requirements before wider dissemination. Recall that SLIM's development process is itself open source based; therefore, there are multiple stages of review and refinement through pull requests, with community feedback playing a crucial role in ensuring that best practices are appropriate and effective. The quality assurance process begins with the initial proposal of a best practice through an issue ticket, where discussion and iteration of the idea occur. This is followed by development of a best practice and submission for review by collaborators and the broader community via pull requests. Feedback from this review is incorporated into revisions, after which the best practice is tested in selected projects to verify its effectiveness in real-world scenarios before larger publication.

4.3 Phase 3: Infusion

The final phase involves the direct infusion of best practice standards to active software projects, ensuring that best practices are not just theoretical but practical and implementable. This hands-on approach differentiates from other similar initiatives, which often stop at the publication of guidelines without addressing implementation challenges directly.

4.3.1 Pull Request Methodology

Best practice standards are infused into target software project codebases through direct pull requests to project repositories. This hands-on approach ensures that improvements are disseminated at scale while the effort required from project teams to implement best practices is also reduced. The pull request method allows for a systematic review process, where proposed changes can be evaluated and refined by project teams before being merged into the project's codebase. This ensures that implemented best practices align with the project's specific needs and constraints, and that they integrate smoothly with existing code and workflows. By creating pull requests, SLIM not only delivers the best practice implementation but also provides an opportunity for project teams to learn about the best practice through the review process. This educational component helps build capacity within teams to maintain and extend the best practice implementation over time. As was mentioned earlier, not all best practices can be readily

infused via pull requests to codebases. For best practices outside this category, such as best practices that directly affect deployed software infrastructure or involve interpersonal behavior changes, SLIM instead creates issue tickets with relevant information to help projects better understand the motivation and steps necessary for adoption.

The SLIM-CLI Tool for Automated Infusion

To streamline the infusion process, the SLIM-CLI tool has been developed – which is a command-line interface designed to automate the application of best practices to Git repositories. This tool serves as a practical implementation of the "standards-as-code" philosophy, enabling teams to seamlessly integrate best practices into their development workflow. The SLIM-CLI tool provides several key capabilities that facilitate direct infusion:

1. The tool can apply multiple best practices to multiple repositories simultaneously. It supports applying multiple best practices across multiple repositories with a single command, significantly reducing the manual effort required to implement best practices across a large project portfolio.
2. SLIM-CLI leverages artificial intelligence capabilities to help customize and tailor the application of best practices based on repository-specific contexts. This ensures that the infused practices are relevant and appropriately adapted to each project's unique needs, rather than being applied in a one-size-fits-all manner.
3. The tool automatically commits and pushes changes to remote software repositories, streamlining the adoption process. This functionality allows for a one-step process to implement best practices, further reducing the implementation burden on project teams. The tool does this by creating standalone Git branches for best practice implementation, allowing teams to review changes before merging them into their main codebase. This approach minimizes disruption to ongoing development efforts, as teams can integrate best practices at a time that aligns with their development schedule.

The SLIM-CLI tool exemplifies how automation can reduce the typical barriers to adopting best practices. By providing a consistent, reliable method for infusing standards directly into codebases, the tool enables projects to implement best practices with minimal manual effort, ensuring that improvements can be efficiently scaled across multiple projects within an organization. For example, to apply and deploy a specific best practice to a repository, SLIM leadership or community teams can use a simple command:

slim apply-deploy --best-practice-ids SLIM-3.1 --repo-urls https://github.com/example/repository --use-ai azure/gpt-4o

This command applies the specified best practice (in this case, a README template), customizes it using AI to match the repository's context, and then commits and pushes the changes to the remote repository—all in a single operation. The direct infusion approach, especially when automated through tools like SLIM-CLI, ensures that best practices move beyond theoretical guidelines to become actual, implemented improvements in active project codebases. This practical focus is central to effectiveness in driving real improvements in software development practices across JPL projects.

5. SLIM Best Practices

The SLIM open source repository and website [10] offers a comprehensive collection of best practice guides and templates that support software development across documentation, governance, and software lifecycle domains. These categories all attempt to embody the "standards-as-code" philosophy, providing practical, adaptable resources that can be directly applied to software projects. Table 1 below provides a current list of available best practice assets.

Table 1: listing of currently available SLIM best practices, their categorization, and links.

Category	Resource	Description	Link
Documentation	README Template	A template for creating clear, informative repository introductions	https://nasa-ammos.github.io/slim/docs/guides/documentation/readme/
	Change Log	Guide for documenting software changes in a human-centric format	https://nasa-ammos.github.io/slim/docs/guides/documentation/change-log/
	Documentation Hosting	Guidance on selecting and implementing documentation hosting tools	https://nasa-ammos.github.io/slim/docs/guides/documentation/documentation-hosts/

Governance	Governance Model	Templates for establishing project decision-making structures for different team sizes	https://nasa-ammos.github.io/slim/docs/guides/governance/governance-model/
	Code of Conduct	Template based on the Contributor Covenant for establishing community behavior standards	https://nasa-ammos.github.io/slim/docs/guides/governance/contributions/code-of-conduct/
	Contributing Guide	Template for creating comprehensive guides for new contributors	https://nasa-ammos.github.io/slim/docs/guides/governance/contributions/contributing-guide/
	Issue Templates	Templates for standardizing bug reports and feature requests in GitHub	https://nasa-ammos.github.io/slim/docs/guides/governance/contributions/issue-templates/
	Pull Request Templates	Templates for standardizing pull request submissions	https://nasa-ammos.github.io/slim/docs/guides/governance/contributions/pull-requests/
Software Lifecycle	Python Starter Kit	Deployment-ready Python 3 application template with integrated CI/CD	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/application-starter-kits/python-starter-kit/
	Continuous Integration	Guidance on implementing automated testing and building processes	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/continuous-integration/
	Continuous Testing	Comprehensive guide for test planning, implementation, and automation	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/continuous-testing/
	Continuous Delivery	Guide for setting up robust delivery pipelines and automating releases	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/continuous-delivery/readme
	GitHub Security	Recommendations for enabling GitHub's security features	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/security/github-security/
	Secrets Detection	Guide for preventing sensitive information leaks in codebases	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/security/secrets-detection/
	Container Vulnerability Scanning	Guide for scanning containers for security vulnerabilities	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/security/container-vulnerability-scanning/
	Metrics Collection	Guide for configuring software lifecycle metrics tracking	https://nasa-ammos.github.io/slim/docs/guides/software-lifecycle/metrics/
Getting Started	Repository Starter Kit	A one-stop resource for setting up new repositories with SLIM best practices	https://nasa-ammos.github.io/slim/docs/guides/checklist

Description: displays the organizational framework of the SLIM repository, illustrating how best practice guides and resources are organized. The table presents a comprehensive collection of best practice resources organized across four key domains: Documentation, Governance, and Software Lifecycle. Each resource is listed with a brief description and corresponding link, demonstrating the "standards-as-code" philosophy in action. This structured approach enables software development teams to quickly locate and implement relevant best practices tailored to their specific needs, from basic repository setup to advanced security implementations.

6. Infusion Results

6.1 Implementation Across JPL

SLIM has been field-tested across hundreds of software repositories at JPL, spanning research projects, mission projects and ground software system support tools. This extensive deployment has helped to validate the efficacy of the infusion approach and helped refine best practices based on real-world conditions.

The clientele includes diverse projects with varying sizes, complexities, and domains. Mission projects have adopted the best practices to enhance their development processes and ensure consistent quality. Research projects with evolving requirements and experimental approaches have used SLIM to establish solid foundations in open source methodology while maintaining the flexibility needed for innovation. Ground software projects, the biggest clientele, have leveraged recommended best practices to improve maintainability and usability within software teams. This breadth of application has provided a robust testing ground for the best practices, ensuring that they address the needs of different types of software projects within the space domain.

SLIM Leaderboard Tool

To systematically track and visualize best practice adoption across repositories, the SLIM team has developed the SLIM-leaderboard tool. This command-line utility generates comprehensive scan reports that showcase how well repositories follow best practices, highlighting areas of compliance and opportunities for improvement. The SLIM-leaderboard tool analyzes GitHub repositories against established best practice criteria, providing a standardized method for evaluating adoption across many repositories simultaneously.

Key features of the SLIM-leaderboard tool include:

- Ability to query a set of GitHub repositories and create a report showcasing compliance to best practices, sorted by most to least compliant
- Compatibility with GitHub.com or GitHub Enterprise repositories
- GraphQL and parallelized queries to GitHub for optimization
- Multiple output format modes including tree, table, and markdown

The leaderboard tool produces leaderboard reports that rank repositories from most to least compliant, offering a clear visual representation of where projects stand in their best practices journey. These reports provide transparency not only to view the impact but also to help encourage healthy competition among projects to implement best practices. The leaderboards serve as a public accountability mechanism, highlighting both achievements and areas for improvement in best practice adoption. Table 2 provides a sample of one such leaderboard report.

Derived from a leaderboard report, an actual scan of the NASA-AMMOS GitHub organization repositories [11] at the time of writing this paper revealed the following overall adoption metrics:

Table 2: metric calculations from a scan of 90+ NASA-AMMOS software repositories

Metric	Value
Overall Best Practice Score (%)	25.0
Additional Documentation Score (%)	61.1
License Score (%)	60.6
Readme Score (%)	46.7
Code of Conduct Score (%)	39.7
Contributing Guide Score (%)	36.4
Changelog Score (%)	32.8
Issue Templates Score (%)	32.2
PR Templates Score (%)	27.5

Governance Model Score (%)	10.0
Secrets Detection Score (%)	2.2
Continuous Testing Plan Score (%)	0.6
GitHub: Vulnerability Alerts Score (%)	0.0
GitHub: Code Scanning Alerts Score (%)	0.0
GitHub: Secret Scanning Alerts Score (%)	0.0

It's important to note that these scan results are only current as of the writing of this paper. Given the dynamic nature of software development and the ongoing efforts to improve best practices across NASA-AMMOS repositories, these metrics are subject to change over time. The team anticipates continued improvement in these metrics as more projects adopt the best practices and ongoing pull requests are merged. Nevertheless, we can garner a few interesting observations. For example, the scan revealed that across 90 evaluated repositories with 14 best practices checked, there were 230 passing checks ("YES"), 197 open pull requests addressing best practices ("PR"), 71 repositories with partial compliance ("PARTIAL"), and 762 non-compliant checks ("NO"). The leaderboard report additionally reveals several important patterns:

1. Adoption of repository best practices recommended by SLIM, like licenses and formatted READMEs are more widely adopted (>60% and >45% respectively) than more involved practices like contributing guides or changelogs (<40%). This reflects a natural progression in best practice adoption, with fundamental documentation being prioritized before more manually intensive practices adopted later.
2. Security-focused practices have the lowest adoption rates (<3%), highlighting an area that requires particular attention. This finding has informed the strategic focus on security best practices and increased efforts to demonstrate its value and facilitate its implementation.

SLIM's iterative approach to development and deployment has proven valuable, allowing for continuous improvement based on real-world experience. The initiative has found that introducing best practices gradually, starting with documentation and governance before moving to more technical practices, leads to higher adoption rates and less resistance from development teams. The automated nature of the SLIM-leaderboard tool supports this iterative approach by making it easy to repeatedly measure adoption across repositories as new best practices are introduced or existing ones are refined. This enables stakeholders to track progress over time and adapt strategies based on quantifiable results. Another key lesson has been the importance of reducing implementation barriers through automation and clear guidance. Projects are more likely to adopt best practices when they can be implemented with minimal effort, particularly when they can be directly integrated into existing workflows through pull requests or automation tools. The flexibility in the approach to different project contexts while maintaining core principles has been key to its successful wide-scale deployment.

7. Community Structure and Governance

The project is structured around a well-defined community model with clear roles, responsibilities, and decision-making processes.

7.1 Community Network

SLIM's community network includes individuals from various organizations, with diverse areas of expertise and contribution patterns. This diversity enriches the development of best practices with perspectives from different domains and applications, ensuring that the practices address a wide range of needs and scenarios. The community network spans multiple JPL projects as well as some external contributors. This broad representation ensures that the best practices benefit from a wealth of experience across different contexts. Contributors bring varied technical expertise, including software development, systems engineering, project management, and documentation. This multidisciplinary approach ensures the initiative addresses all aspects of software development, from technical implementation to process management and communication.

7.2 Governance Model

SLIM follows a liberal contribution model where people and organizations who do the most work have the most influence on project direction. Technical decision-making primarily follows a "consensus-seeking" approach within governing committees, ensuring that decisions benefit from diverse perspectives while still allowing for efficient

progress.

The governance model, as defined in the GOVERNANCE.md (Fig. 3), articulates roles and responsibilities, from users and contributors to technical steering committee members and project steering committee members. Each role has specific permissions and decision-making authority, creating a structured but flexible framework for project governance. This structure provides a framework for community participation and decision-making, ensuring that contributions are effectively coordinated and integrated into the overall framework. Users represent the broadest category, encompassing anyone who downloads, deploys, or operates software that implements the best practices. Contributors include those who provide input to the project, whether through code, documentation, graphics, or other means. Triagers are contributors who have demonstrated familiarity with the project and help manage issues and pull requests. Collaborators have been granted write access to repositories, allowing them to review and approve contributions. The Technical Steering Committee (TSC) consists of collaborators who have technical decision-making authority and administrative privileges, guiding the technical direction of the project. The Project Steering Committee (PSC) includes representatives from sponsoring organizations and key stakeholders, with authority to guide the project based on requirements, budget, and schedule considerations. The Product Manager has final authority over key decisions when consensus cannot be reached within the governance committees. This structured hierarchy of roles allows for both broad participation and efficient decision-making, creating a balanced approach to community governance. The governance model draws inspiration from other notable open-source projects, including Node.js, OpenSSL, PostgreSQL, and OpenMCT. By adapting proven governance approaches from successful open-source projects, SLIM benefits from established best practices while tailoring them to the specific needs of the space software community.

Figure 3: SLIM's governance approach, illustrated through various roles available.

Role	Restricted To	Description	Read/Clone	Propose Pull Request	Comment in Tickets / Discussions	Triage	Review	Commit	Technical Decisions	Project Decisions
User	None	Anyone downloading, deploying, or operating the software to meet a specific objective.	✅	✅	✅	❌	❌	❌	❌	❌
Contributor	None	Anyone providing input to the project, including: code, issues, documentation, graphics, etc.	✅	✅	✅	❌	❌	❌	❌	❌
Triager	Contributor	Subset of contributors demonstrating a strong familiarity with the project.	✅	✅	✅	✅	❌	❌	❌	❌
Collaborator	Contributor	Subset of contributors granted write access to one or more of the project repositories upon selection by TSC	✅	✅	✅	✅	✅	✅	❌	❌
Technical Steering Committee Member	Collaborator	A subset of collaborators having technical decision making authority and admin privileges	✅	✅	✅	✅	✅	✅	✅	❌
Project Steering Committee Member	Stakeholders	Sponsor organization representatives (i.e. those providing funding to the project) and key stakeholders with authority to guide project based on requirements, budget, schedule, etc.	✅	✅	✅	✅	✅	✅	✅	✅
Product Manager	Stakeholders	Overall manager of project with final authority over all key decisions when consensus cannot be reached	✅	✅	✅	✅	✅	✅	✅	✅

8. Conclusions

SLIM continues to evolve with planned enhancements to address emerging needs in the space software community. Future development will focus on increasing the role of artificial intelligence for customizing assets for infusion to projects as well as focusing more on security. Specifically, the SLIM-CLI tool will receive expanded capabilities, including more sophisticated AI-enhanced customization options to further reduce implementation barriers. In particular, the generative AI capabilities will help further improve the infusion aspect of best practice lifecycles, where we envision even complex best practice recommendations to be completely tailored for target projects through leveraging generative AI to perform the customizations automatically. Comprehensive metrics and reporting capabilities will be integrated into the SLIM-CLI tool to first identify gaps in best practices, and then to help fill those gaps automatically. Further, a greater strategic focus on helping to "modernize" legacy software will

be enacted, to support greater outreach to JPL and NASA projects for maintainability purposes. This may involve suggestions of code modifications or refactoring, based on AI-assisted recommendations.

While developed within JPL, the "standards-as-code" approach has potential applications in other domains. The open-source nature of SLIM makes it accessible to government agencies, academic institutions, and private companies worldwide. Research opportunities include exploring the effectiveness of different practices in various contexts, advancing AI integration for best practice development, developing more sophisticated impact metrics, understanding social factors influencing adoption, balancing standardization with project-specific needs, and ensuring long-term sustainability of community-driven initiatives.

The project represents an advancement in how software development best practices are created, shared, and implemented within the space domain. By combining open-source principles with a "standards-as-code" approach, SLIM has demonstrated a powerful new paradigm for improving software quality and development processes. The initiative's focus on project governance, software lifecycle automation, and software documentation addresses critical aspects of software development that impact project success. The three-phase methodology – community input, development of actionable standards, and direct infusion – bridges the gap between theoretical best practices and actual implementation. The repository components, including templates, automation tools, and starter kits, provide tangible resources that projects can immediately apply. Tools like SLIM-leaderboard and SLIM-CLI demonstrate approaches to tracking and implementing best practices automatically. The community structure and governance model provide a sustainable framework for ongoing development and maintenance. Looking forward, we envision best practices as living artifacts that evolve through continuous refinement, with community collaboration and advanced automation ensuring they remain relevant in a changing technological landscape. We invite broader participation from the space software development community to contribute to this ongoing effo**rt.**

Acknowledgements

The work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). The authors wish to thank the SLIM contributor community and the AMMOS program for their support of this initiative.

References

[1] NASA JPL, Hybrid Cloud Science Data System (HySDS), https://hysds.github.io/, (accessed 02.04.25).
[2] NASA JPL, Observational Products for End-Users from Remote Sensing Analysis (OPERA), https://nasa-opera.github.io/, (accessed 02.04.25).
[3] NASA JPL, Planetary Data System Engineering Node, https://pds-engineering.jpl.nasa.gov/, (accessed 02.04.25).
[4] NASA JPL, Unity Science Data System (Unity SDS) Documentation, https://unity-sds.gitbook.io/docs, (accessed 02.04.25).
[5] NASA, Advanced Multi-Mission Operations System (AMMOS), https://ammos.nasa.gov/, (accessed 18.04.25).
[6] NASA-AMMOS, SLIM (Software Lifecycle Improvement & Modernization) issues, https://github.com/NASA-AMMOS/slim/issues?q=is%3Aissue, (accessed 18.04.25).
[7] NASA-AMMOS, SLIM Community Planning Board, https://github.com/orgs/NASA-AMMOS/projects/3, (accessed 18.04.25).
[8] NASA-AMMOS, How to Contribute, https://nasa-ammos.github.io/slim/docs/contribute/contributing/, (accessed 18.04.25).
[9] NASA-AMMOS, Submit a Best Practice Guide, https://nasa-ammos.github.io/slim/docs/contribute/submit-best-practice, (accessed 18.04.25).
[10] NASA-AMMOS, Software Lifecycle Improvement & Modernization (SLIM), https://nasa-ammos.github.io/slim/, (accessed 18.04.25).
[11] NASA, NASA Advanced Multi-Mission Operations System (AMMOS) GitHub, https://github.com/nasa-ammos, (accessed 18.04.25).