Research Software focus area Maturity Model: Description and Practices

Deekshitha, Rena Bakhshi, Jason Maassen, Carlos Martinez Ortiz
Rob van Nieuwpoort, Slinger Jansen

1 The Maturity Model

Figure 1 shows the focus areas and capabilities of the Research Software focus area Maturity Model (RSMM) and Figure 2 depicts the complete model. RSMM includes 4 focus areas, 17 capabilities, and 79 practices. The following subsection briefly describes 4 focus areas, capabilities, and practices of the RSMM.

1.1 Focus Areas

1.1.1 Software project management

Software project management manages the resources and work activities needed to develop and modify software-intensive systems [29]. The software project is a highly people-intensive effort that extends over a considerable period, significantly impacting the work and performance of various stakeholders, including project managers. The primary success criteria for software managers are delivery of systems that satisfy specified needs and **requirements**, on time and within budget [29]. Software development involves challenges, including evolving technology, immature technology, sloppy development practices, and staff changes. **Software project management** addresses these complexities by promoting stakeholder involvement, managing risks, and fostering transparent communication, helping project managers navigate and overcome these obstacles.

Thus, various factors influence the software project management process, and they are almost the same for research software development. We included 3 capabilities and 19 practices related to this focus area. The capabilities of this focus area are:

- **Requirements**
- **Code quality and security**
- **Communication and Collaboration**

1.1.2 Research software management

Like software management, **Research software management** is a process of managing research software. Increasing usage of research software in the various research domains highlights the importance of adhering to best practices [12]. This model includes 4 capabilities and 22 practices. The capabilities of this focus area are:

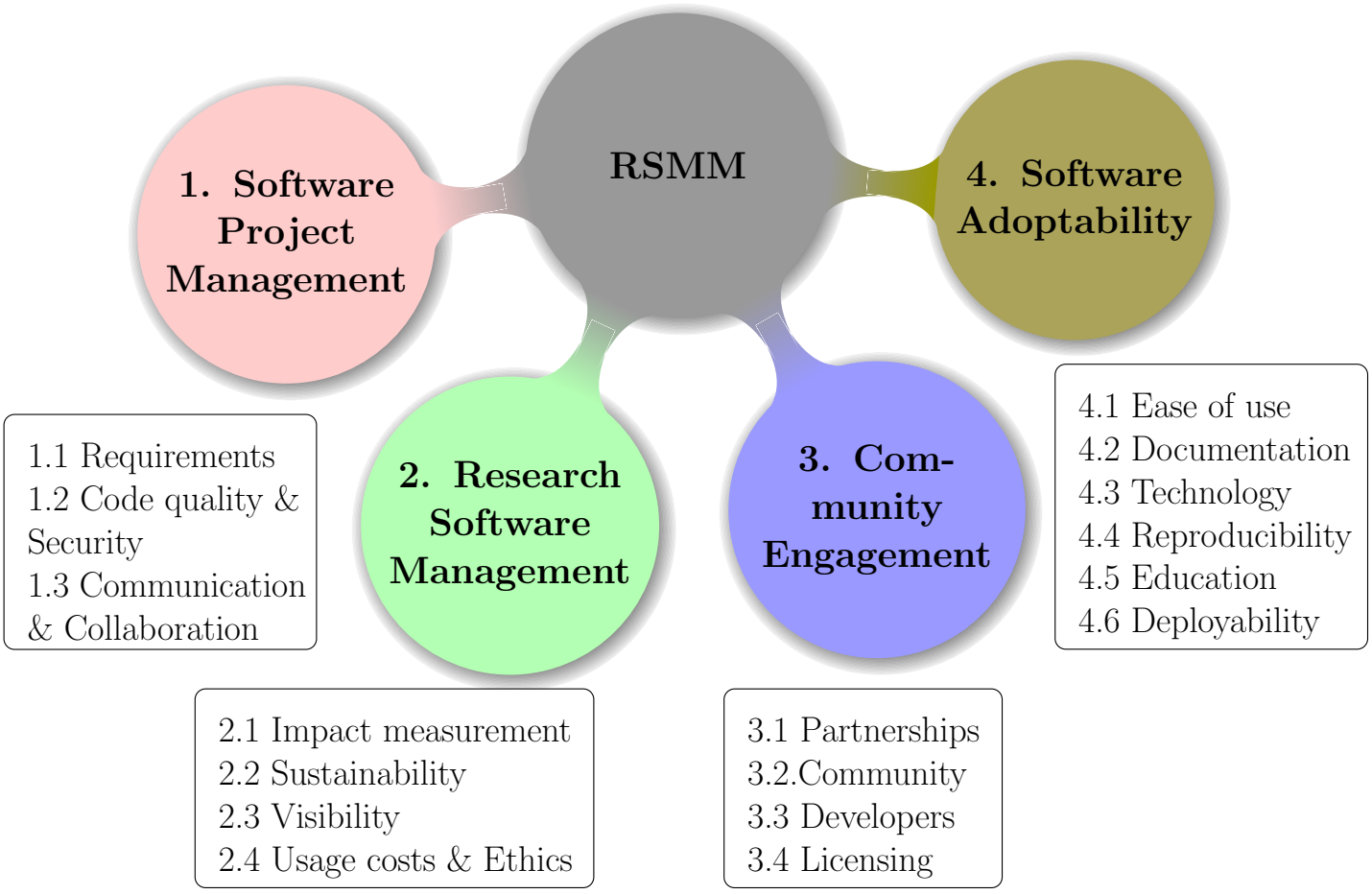

Figure 1: Focus areas and capabilities: The 4 focus areas and 17 capabilities of the RSMM v1.0 are shown in the figure.

Maturity levels	1	2	3	4	5	6	7	8	9	10
1	**Software Project Management**									
Requirements		File issues in an issue tracker	Act on feedback				Manage requirements explicitly	Perform release management		Communicate roadmap
Code quality and security	Provide a coding standard	Conduct code reviews	Implement continuous integration	Provide executable tests	Use crash reporting	Conduct security reviews	Measure project stability continuously	Define code coverage targets	Execute tests in a public workflow	Follow an industry standard for security
Communication and collaboration			Store project in public repository with version control				Use public communication platform	Provide news letter		Provide community website
2	**Research Software Management**									
Impact measurement		Define a clear audience for the project	Perform infrequent impact measurement				Evaluate whether the audience's goals are met		Perform continuous impact measurement	Explore new audiences regularly
Sustainability		Acquire temporary funding		Write software management plan			Obtain support from a national research software center	Acquire viable pathways for project sustainability	Secure continuous funding	Define end-of-life policy
Visibility	Make code citable		Enable indexing of project meta-data	Promote the project continuously	Publish in a research software directory	Acquire research software center acknowledgement	Enable indexing of the project's source code			Garner industrial partner adoption
Usage costs & Ethics				Analyze privacy usage impact	Analyze ethical consequences of project use	Document the cost of running the application	Consider total energy consumption			
3	**Community Engagement**									
Partnerships				Acknowledge partners and funding agencies on website			Develop advanced partnership model			
Community			Impose community norms	Onboard researchers as part of the community	Develop code of conduct	Appoint support team	Organize community events		Provide front page chat support	Focus on diversity and inclusion
Developers		Make developer names and roles publicly available				Document how to join the team		Set maximum response time for pull requests	Provide access to developer training and skill development	
Licensing	Select a license		Get institutional support for license choice					Evaluate license policy regularly		
4	**Software Adoptability**									
Ease of use		Provide a statement of purpose	Provide a simple how to use			Provide online tutorials				
Documentation		Provide a read me file with project explanation	Provide a how-to guide	Provide a common example usage		Provide a documentation as repository/ documentation wiki		Provide API documentation		
Technology		Use common non-exotic or established technology		Facilitate integration into scientific workflow					Evaluate technology relevance regularly	
Reproducibility		Provide instructions on how to put into research workflow		Provide instructions on how to make part of a replication package	Make part of standardized workflows		Make part of a replication package			
Education					Develop generic educational materials				Organize training events in person	Make project part of an educational program
Deployability						Provide with standard deployment tools		Enable deployment on a wide range of technology	Provide coordination mechanisms for workflow distribution over different machines	Generate SBOM

Figure 2: RSMM v1.0: The updated version of RSMM. It includes 4 focus areas, 17 capabilities, and 79 practices. These practices are placed between maturity levels 1-10.

- **Impact measurement**
- **Sustainability**
- **Visibility**,
- **Usage cost and Ethics**

1.1.3 Community Engagement

A community generally evolves and maintains research software, creating an ecosystem of competing and collaborative products. It is influenced by the open-source movement's culture of sharing and collaboration [44]. This includes 4 capabilities and 16 practices associated with this focus area to foster community development around research software. Capabilities of this focus area are:

- **Partnerships**
- **Community**
- **Developers**
- **Licensing**

1.1.4 Software Adoptability

The focus area **Software Adoptability** concerns with how easily and effectively research software can be adopted and utilized by users. This focus area is aimed at understanding and enhancing the user-friendliness, accessibility, and overall adoption strategies of research software by the research community. It includes 6 capabilities and 22 practices. Capabilities are:

- **Ease of use**
- **Documentation**
- **Technology**
- **Reproducibility**
- **Education**
- **Deployability**

1.2 Practices Description

In the following list all 79 practices from the RSMM are described. For each practice, we provide

- its *Practice Code*;
- *Name*;
- *Description*;

- *When Implemented*. For each description, we apply the MoSCoW method [47] to categorize it as Must Have (M), Should Have (S), or Could Have (C). Note that the Won't Have (W) category is not used in these descriptions;
- *Resources required*;
- *Dependencies*; and
- *References*.

Acknowledgement

We want to acknowledge the experts who contributed to redefining RSMM (not in any particular order): Bernadette Fritzsch, Jayesh Badwaik, Michael Schlottke-Lakemper, Pablo Lopez-Tarifa, Raoul Schram, Nicolas Renaud, Arend Rensink, Jan Philipp Dietrich, Axel Loewe, Aljen Uitbeijerse, Tomas Turner-Zwinkels, and Martine de Vos.

Practice Code: 1.1.2
Practice Name: File issues in an issue tracker
Description: This practice involves documenting and reporting bugs, tasks, or feature requests in an issue tracking system ensuring they are properly documented and addressed.
When implemented:

- (M) Users are encouraged to file issues in the project's issue tracker whenever they encounter bugs, have feature requests, or need assistance.
- (S) Regular updates are provided to users on the status and progress of their filed issues, including when they are being addressed or resolved.
- (C) Issues filed by users are promptly reviewed, categorized, and prioritized based on their severity and impact on the project.

Resources required:

- Time: It depends on the volume and complexity of reported issues, requiring regular monitoring and management.
- Issue tracking software or platform for efficiently managing and tracking reported issues.
- Communication channels for facilitating interaction between users and the researchers / research software engineers regarding filed issues.
- Development resources for fixing bugs, implementing requested features, or providing support in response to reported issues.

Dependencies: 1.1.1 < 1.1.7
References: [65, 38, 7]

Practice Code: 1.1.3
Practice Name: Act on feedback
Description: This practice involves actively engaging with feedback received from various stakeholders, including users, clients, or internal team members.
When implemented:

- (M) Research Software Engineers actively solicit feedback from users, collaborators, and stakeholders through various channels such as user surveys, bug reports, feature requests, and direct communication.
- (M) Feedback received is thoroughly reviewed and considered by the researchers / research software engineers, acknowledging its importance in improving the research software.
- (S) researchers / research software engineers communicate transparently with the feedback providers, providing updates on the status of their suggestions and the actions taken in response.
- (C) Research Software Engineers prioritize feedback based on its impact on the research software's objectives and roadmap, focusing on addressing critical issues and implementing valuable suggestions.

Resources required:

- Time: It depends on the frequency and volume of feedback received, but typically involves regular intervals for feedback review and response.
- Collaboration tools for facilitating communication and collaboration between researchers / research software engineers and feedback providers.
- Development resources to implement necessary changes or improvements based on feedback (e.g., coding, testing, documentation updates).
- Training or guidelines for researchers / research software engineers on how to effectively engage with and act on feedback in a constructive manner.

Dependencies:
References: [2, 25]

Practice Code: 1.1.7
Practice Name: Manage requirements explicitly
Description: Clearly define, document, and manage the requirements of the research software project throughout the development lifecycle to ensure alignment with stakeholders' needs and project goals.
When implemented:

- (M) All functional and non-functional requirements of the research software project are explicitly defined and documented, including features, functionalities, performance criteria, and constraints.
- (M) Requirements are validated and approved by relevant stakeholders, including researchers, end-users, project sponsors, and domain experts, to ensure that they accurately reflect their needs and expectations.
- (S) Requirements are traced and prioritized throughout the development lifecycle, with clear links established between requirements, design decisions, implementation tasks, and testing activities to track progress and manage changes effectively.
- (C) The requirements management process is adaptable and responsive to evolving project needs, with mechanisms in place to capture, evaluate, and incorporate new requirements or changes requested by stakeholders.

Resources required:

- Time: Required to implement a system in which requirements are made explicit, in terms of time and effort, as to facilitate planning. Furthermore, a requirements prioritization method must be selected.
- Use of requirement elicitation techniques (e.g., interviews, surveys, workshops), documentation tools (e.g., requirements management software, wikis, spreadsheets), and collaboration platforms (e.g., project management software, version control systems) to capture, organize, and track requirements.
- Engagement and collaboration with stakeholders throughout the requirements management process, including regular meetings, reviews, and feedback sessions to ensure alignment, transparency, and accountability in requirement decisions and changes.

Dependencies:
References: [77, 8]

Practice Code: 1.1.8
Practice Name: Perform release management
Description: Implement a systematic approach to plan, coordinate, and execute the release of software versions, ensuring that new features, enhancements, and bug fixes are delivered to users in a timely and organized manner.
When implemented:

- (M) The project follows a defined release cycle, with clear milestones, timelines, and objectives for each release, allowing for systematic planning and coordination of release activities.
- (M) Software versions are managed using version control systems (e.g., Git) and tagged with unique identifiers, enabling precise tracking and documentation of changes included in each release.
- (S) Release content is determined through collaborative planning and prioritization, considering factors such as user feedback, project roadmap, stakeholder requirements, and development effort.
- (S) Releases undergo rigorous testing and QA processes, including functional testing, regression testing, performance testing, and user acceptance testing, to ensure that they meet quality standards and are fit for deployment.

Resources required:

- Time: It depends on the complexity and frequency of releases, required to release planning and coordination for each release cycle.
- Use of release management tools and automation scripts to streamline release processes, including version control systems, continuous integration/delivery (CI/CD) pipelines, deployment orchestration tools, and issue tracking systems.
- Provisioning of testing environments, tools, and resources to support thorough testing and validation of releases across different configurations and environments.
- Preparation of release notes, documentation, and communication materials to inform users, stakeholders, and the researchers / research software engineers about the contents, changes, and improvements introduced in each release.

Dependencies: 1.2.4 < 1.1.8
References: [81, 28]

Practice Code: 1.1.10
Practice Name: Communicate roadmap
Description: Communicating the project's goals and planned milestones to stakeholders using a roadmap.
When implemented:

- (M) A clear and comprehensive roadmap for the research software project is documented, detailing key milestones, deliverables, and timelines.
- (S) The roadmap is regularly updated to reflect changes in project priorities, resource availability, and stakeholder feedback.
- (C) The roadmap is accessible to all relevant stakeholders, including team members, funders, collaborators, and users, through centralized platforms or communication channels.
- (C) Stakeholders are informed proactively about updates to the roadmap, highlighting any significant changes or adjustments to project plans.

Resources required:

- Time: Regular intervals for updating and communicating the roadmap, depending on the pace of project development and changes in project priorities.
- Communication platforms or tools for sharing the roadmap with stakeholders (e.g., project management software, shared documents, email newsletters).
- Coordination efforts among project team members to ensure alignment between the roadmap and ongoing project activities.
- Training or guidelines for stakeholders on how to interpret and engage with the project roadmap effectively.

Dependencies:
References: [77, 10]

Practice Code: 1.2.1
Practice Name: Provide a coding standard
Description: Establish and communicate a coding standard or style guide for the research software project to promote consistency, readability, and maintainability of the codebase.
When implemented:

- (M) A comprehensive coding standard or style guide is documented and made available to all project contributors, outlining conventions and best practices for writing code.
- (M) The coding standard covers various aspects of coding, including naming conventions, formatting, commenting, error handling, and documentation requirements.
- (M) All code contributed to the project adheres to the established coding standard, with deviations documented and justified as necessary.
- (S) Regular reviews and updates are conducted to ensure that the coding standard remains relevant and reflective of evolving best practices and project requirements.

Resources required:

- Time: It is needed initially to develop and document the coding standard, and ongoing time is required to enforce compliance and update the standard as needed.
- Collaboration tools or platforms for sharing and managing the coding standard document, ensuring accessibility to all project contributors.
- Training or guidelines for team members on how to interpret and apply the coding standard effectively in their coding practices.
- QA processes or automated tools to check code against the coding standard and provide feedback to researchers / research software engineers on adherence and potential violations.

Dependencies: 1.2.1 $<$ * (all other practices)
References: [15, 24]

Practice Code: 1.2.2
Practice Name: Conduct code reviews
Description: Implement a systematic process of reviewing code changes made by team members to ensure code quality, consistency, and adherence to coding standards within the research software project.
When implemented:

- (M) All code changes, including new features, bug fixes, and enhancements, undergo thorough peer review by team members before being merged into the main codebase.
- (S) Reviewers provide constructive feedback on code changes, focusing on identifying potential issues, suggesting improvements, and ensuring alignment with project requirements.
- (C) Code reviews are conducted using established guidelines and criteria, covering aspects such as functionality, performance, readability, and maintainability.
- (C) Reviews are documented, and feedback is addressed promptly, with necessary revisions made to the code before finalizing the changes.

Resources required:

- Time: It depends on the size and complexity of code changes, but typically involves allocating time for researchers / research software engineers to conduct thorough reviews.
- Code review tools or platforms (e.g., GitHub pull requests, GitLab merge requests, code review plugins) to facilitate the review process and track feedback.
- Training for team members on best practices for conducting and participating in code reviews effectively.
- Coordination efforts to ensure timely completion of code reviews and resolution of any identified issues or concerns.

Dependencies:
References: [41, 46, 7]

Practice Code: 1.2.3
Practice Name: Implement continuous integration
Description: Establish a process of integrating and testing code changes automatically and frequently within the research software project's development environment to detect and address issues early in the development lifecycle.
When implemented:

- (M) A continuous integration (CI) system is set up to automatically build, test, and validate code changes whenever new code is committed to the version control repository.
- (M) Automated tests, including unit tests, integration tests, and other relevant tests, are integrated into the CI pipeline to ensure code quality and functionality.
- (M) researchers / research software engineers receive immediate notifications of build or test failures, enabling them to address issues promptly and prevent integration conflicts.
- (S) The CI pipeline is configured to provide rapid feedback on the status of code changes, including build success/failure, test results, and code coverage metrics.

Resources required:

- Time: Initial setup time is required to configure the CI system and integrate automated tests into the pipeline. Ongoing time is needed to maintain and update the CI configuration as the project evolves.
- CI/CD tools or platforms (e.g., Jenkins, Travis CI, GitLab CI/CD) to automate the build, test, and deployment processes.
- Access to suitable testing environments and resources for running automated tests as part of the CI pipeline.
- Training or guidance for team members on using CI tools effectively and interpreting CI results to drive development efforts.

Dependencies:
References: [65, 77]

Practice Code: 1.2.4
Practice Name: Provide executable tests
Description: Ensure that the research software project includes executable tests to validate its functionality, behavior, and performance automatically.
When implemented:

- (M) Regular security reviews are scheduled and conducted at predefined intervals or in response to significant code changes or updates.
- (S) A suite of automated tests is developed alongside the research software project, covering critical functionality, edge cases, and performance benchmarks.
- (C) Tests are written using appropriate testing frameworks and methodologies, such as unit tests, integration tests, and end-to-end tests, depending on the nature of the software.
- (C) Automated tests are regularly executed as part of the CI/CD pipeline, providing rapid feedback on code changes and ensuring software quality and stability.
- (C) Test results are monitored and reviewed systematically, with failed tests investigated promptly and necessary actions taken to address underlying issues.

Resources required:

- Time: Time is required to develop automated test suites, and ongoing time is needed to maintain and update tests as the software evolves.
- Testing frameworks and tools suitable for the project's programming language and technology stack (e.g., JUnit, PyTest, Selenium).
- Integration with CI/CD platforms to automate test execution and result reporting.
- Training or expertise in test automation and best practices for team members involved in test development and execution.

Dependencies: 1.2.4 < 1.2.9
References: [65, 41]

Practice Code: 1.2.5
Practice Name: Use crash reporting
Description: Employ crash reporting tools or mechanisms to monitor and collect information about software crashes and exceptions occurring in the research software project, enabling rapid identification and resolution of issues.
When implemented:

- (M) Crash reporting functionality is integrated into the software application, capturing critical information such as error messages, stack traces, and environmental data when a crash occurs.
- (M) Crash reports are automatically generated and transmitted to a centralized platform or service upon occurrence, allowing researchers / research software engineers to track and analyze software stability in real time.
- (M) researchers / research software engineers receive timely alerts or notifications for critical crashes, enabling them to prioritize and address issues promptly to minimize the impact on users and maintain software reliability.
- (S) Crash reports include metadata such as the version of the software, operating system details, and user actions leading up to the crash, facilitating root cause analysis and troubleshooting.

Resources required:

- Time: Initial setup time is required to integrate crash reporting functionality into the software application. Ongoing time is needed to monitor and analyze crash reports, investigate issues, and implement fixes.
- Crash reporting tools or services (e.g., Sentry, Bugsnag, Crashlytics) capable of capturing and reporting software crashes effectively.
- Access to error tracking and logging libraries or frameworks to instrument the application code for crash reporting.
- Training or guidelines for team members on interpreting crash reports, identifying patterns, and prioritizing issues for resolution based on severity and impact.

Dependencies:
References: [5, 59]

Practice Code: 1.2.6
Practice Name: Conduct security reviews
Description: Implement a systematic process for evaluating the security aspects of the research software project to identify and mitigate potential vulnerabilities and risks.
When implemented:

- (M) Regular security reviews are scheduled and conducted at predefined intervals or in response to significant code changes or updates.
- (S) Security assessments cover various aspects of the software, including authentication, authorization, data encryption, input validation, and protection against common security threats.
- (S) Security reviews involve collaboration with security experts or specialists to ensure thorough analysis and adherence to industry best practices and standards.
- (C) Identified security vulnerabilities and risks are documented, prioritized based on their severity and potential impact, and addressed promptly through appropriate remediation measures.

Resources required:

- Time: It depends on the size and complexity of the software, but typically involves dedicating time to conducting comprehensive security assessments and addressing identified issues.
- Security assessment tools or services (e.g., static code analysis tools, penetration testing tools, security scanners) to identify vulnerabilities and assess the overall security posture of the software.
- Training or expertise in security principles and practices for team members involved in conducting security reviews.
- Collaboration with external security experts or consultants, if necessary, to augment internal expertise and ensure comprehensive security assessments.

Dependencies: $1.2.6 < 1.2.10$
References: [68, 63]

Practice Code: 1.2.7
Practice Name: Measure project stability continuously
Description: Implement a system to monitor and assess the stability of the research software project continuously, enabling proactive identification of potential issues and trends affecting software reliability.
When implemented:

- (M) Key metrics and indicators of project stability, such as error rates, crash frequency, uptime/downtime, and performance metrics, are defined and monitored systematically.
- (M) Monitoring tools or systems are configured to collect and analyze stability-related data in real time or at regular intervals, providing insights into the overall health and reliability of the research software.
- (S) Thresholds or benchmarks are established for stability metrics, allowing deviations from expected norms to trigger alerts or notifications for further investigation and action.
- (S) Trends and patterns in stability metrics are tracked over time, enabling stakeholders to identify areas of improvement, assess the impact of changes, and make informed decisions to enhance project stability.

Resources required:

- Time: It is required to define relevant stability metrics, set up monitoring systems, and establish baseline performance indicators. Ongoing time is needed to maintain monitoring systems and analyze stability data.
- Monitoring and analytics tools or platforms capable of collecting, processing, and visualizing stability-related metrics effectively (e.g., Grafana, Datadog, Prometheus).
- Integration with logging, error tracking, and performance monitoring tools to aggregate relevant data for stability assessment.
- Training or expertise in data analysis and interpretation for team members involved in monitoring project stability and responding to identified issues or trends.

Dependencies:
References: [3, 69]

Practice Code: 1.2.8
Practice Name: Define code coverage targets
Description: Establish specific goals or targets for code coverage, which measures the percentage of code lines or branches covered by automated tests, to ensure adequate test coverage and QA.
When implemented:

- (M) Code coverage targets are clearly defined and documented, specifying the desired minimum level of code coverage for different components, modules, or layers of the software project.
- (M) Code coverage targets are aligned with project quality goals, objectives, and stakeholders' expectations, ensuring that they contribute to overall software reliability, maintainability, and testability.
- (S) Code coverage metrics are regularly monitored and assessed throughout the development lifecycle, with progress tracked against established targets and deviations addressed on time.
- (S) The project implements strategies and initiatives to continuously improve code coverage over time, including adding new tests, optimizing existing tests, and addressing coverage gaps identified through analysis.

Resources required:

- Time: Required to define, monitor, and manage code coverage targets based on project size, complexity, and existing test coverage.
- Use code coverage analysis tools and metrics reporting frameworks to measure, track, and visualize code coverage, identifying areas that need attention.
- Employ automation frameworks for unit, integration, and end-to-end tests to achieve comprehensive coverage efficiently.
- Provide researchers / research software engineers and testers with education on code coverage importance, best testing practices, and strategies for improving coverage.

Dependencies:
References: [18, 84]

Practice Code: 1.2.9
Practice Name: Execute tests in a public workflow
Description: Ensure that the tests for the research software project are executed within a public workflow, enabling transparency, reproducibility, and collaboration among stakeholders.
When implemented:

- (M) Test execution is triggered automatically upon code changes or updates, providing real time feedback on the research software's integrity and functionality to all stakeholders.
- (M) Test results, including successes, failures, and performance metrics, are published openly, allowing stakeholders to review and validate the research software's behavior and quality.
- (S) The project's test suite is integrated into a publicly accessible workflow or pipeline, hosted in a version control system or a dedicated testing platform.
- (C) Collaboration features, such as commenting and issue tracking, are enabled within the public testing workflow, facilitating communication and coordination among contributors and users.

Resources required:

- Time: Initial setup time is required to integrate the test suite into a public workflow, and ongoing time is needed to maintain and update the workflow as the project evolves.
- Integration with version control systems (e.g., GitLab CI/CD, GitHub Actions, Bitbucket Pipelines) or dedicated testing platforms (e.g., Travis CI, CircleCI) to automate test execution within a public environment.
- Access to collaboration and communication tools to enable interaction and feedback exchange among stakeholders within the public testing workflow.
- Training or guidance for team members on setting up and managing public testing workflows effectively while ensuring security and privacy considerations are addressed.

Dependencies:
References: [65, 41]

Practice Code: 1.2.10
Practice Name: Follow an industry standard for security (e.g., OWASP SAMM, BSIMM, OpenSSF)
Description: Adhere to recognized industry standards and best practices for cybersecurity, such as OWASP, SAMM, BSIMM, and OpenSSF, to ensure the security of the research software project.
When implemented:

- (M) The research software project aligns its security practices with a specific industry standard or framework, selecting one that is appropriate for its technology stack, domain, and organizational context.
- (M) Security requirements and guidelines outlined in the chosen standard are integrated into the project's development lifecycle, including design, implementation, testing, and deployment phases.
- (M) Regular assessments and audits are conducted to evaluate the project's adherence to the selected standard and identify areas for improvement or remediation.
- (M) Documentation is maintained to demonstrate compliance with the industry standard, including security policies, procedures, risk assessments, and mitigation plans.

Resources required:

- Time: Significant time may be required initially to research, select, and adopt an appropriate industry standard for security. Ongoing time is needed to integrate security practices into the development process and conduct regular assessments.
- Tools and resources specific to the chosen industry standard, such as reference guides, templates, and assessment tools, to support compliance efforts.
- Collaboration with security experts or consultants, if necessary, to ensure alignment with industry best practices and address complex security challenges effectively.
- Training or expertise in cybersecurity and industry standards for team members involved in implementing and maintaining security measures.

Dependencies:
References: [66, 80]

Practice Code: 1.3.3
Practice Name: Store project in public repository with version control
Description: Host the research software project in a public repository using version control systems (e.g., Git, Subversion) to enable collaborative development, transparency, and version tracking.
When implemented:

- (M) The project codebase is stored in a public repository accessible to all project stakeholders, typically hosted on platforms like GitHub, GitLab, or Bitbucket.
- (M) Version control systems such as Git are utilized to manage changes to the project codebase systematically, providing a history of modifications, branches for feature development, and mechanisms for collaboration.
- (M) Contributions to the project are made via pull requests or merge requests, allowing for peer review, discussion, and validation of code changes before integration into the main codebase.
- (S) The repository is organized logically, with clear directory structures, documentation, and guidelines for contributors to navigate and understand the project's architecture and components.

Resources required:

- Time: Required to set up the public repository, establish version control workflows, and define contribution guidelines. Ongoing time is required for managing repository administration, reviewing contributions, and maintaining project documentation.
- Access to version control platforms and tools for hosting and managing the public repository (e.g., GitHub, GitLab, Bitbucket).
- Communication channels and collaboration tools to facilitate interaction and coordination among project contributors, such as issue trackers, discussion forums, and chat platforms.
- Training or guidance for team members on version control best practices, branching strategies, and collaboration workflows using the chosen version control system.

Dependencies:
References: [12, 40, 27]

Practice Code: 1.3.7
Practice Name: Use public communication platform (e.g., email list, Slack)
Description: Employ a public communication platform such as an email list, Slack workspace, or similar tool to facilitate open and transparent communication among project stakeholders, fostering collaboration, knowledge sharing, and community engagement.
When implemented:

- (M) A public communication platform is established and accessible to all project stakeholders, providing a central hub for discussions, announcements, and information sharing related to the research software project.
- (M) Project updates, announcements, and important discussions are regularly shared on the public communication platform, keeping stakeholders informed about project progress, milestones, and decisions.
- (M) Participation in the communication platform is encouraged and inclusive, welcoming contributions and feedback from all stakeholders, regardless of their role or affiliation.
- (S) The platform is configured with appropriate channels or threads to organize discussions by topic, allowing participants to focus on relevant subjects and minimize noise.

Resources required:

- Time: Required to figure out the public communication platform, establish communication norms and guidelines, and promote adoption among stakeholders. Ongoing time is required for moderation, facilitation, and engagement on the platform.
- Access to public communication platforms or tools suitable for the project's needs and preferences (e.g., email list services, Slack, Discord, Microsoft Teams).
- Coordination efforts to ensure alignment between communication on the platform and other project management and collaboration tools, such as issue trackers and version control systems.
- Training or guidance for team members on how to use the communication platform effectively, and best practices for engaging with other participants.

Dependencies:
References: [34, 7]

Practice Code: 1.3.8
Practice Name: Provide newsletter
Description: Distribute a regular newsletter to project stakeholders containing updates, announcements, highlights, and relevant information about the research software project, enhancing communication and engagement.
When implemented:

- (M) A newsletter schedule is established, specifying the frequency (e.g., weekly, bi-weekly, monthly) and distribution channels for sending out project updates and announcements.
- (M) The newsletter content includes a variety of topics such as project milestones, recent developments, upcoming events, relevant resources, community contributions, and opportunities for involvement.
- (S) The newsletter content is carefully selected and presented in a clear, concise, and engaging manner, tailored to the diverse interests and requirements of stakeholders such as researchers / research Software Engineers, users, funders, and collaborators.
- (C) Feedback mechanisms are provided to allow recipients to provide input, suggest topics, or express preferences for newsletter content, ensuring ongoing relevance and alignment with stakeholder expectations.

Resources required:

- Time: Required to plan, and create newsletter content, as well as to distribute and manage subscriber lists.
- Newsletter distribution platforms or tools (e.g., Mailchimp, Constant Contact, Substack) for creating and sending newsletters, managing subscriber lists, and tracking engagement metrics.
- Collaboration with subject matter experts, project contributors, and community members to gather content and insights for inclusion in the newsletter.
- Graphic design resources or templates for creating visually appealing newsletter layouts and branding elements, if applicable.
- Evaluation and analysis tools to measure newsletter performance, including open rates, click-through rates, and subscriber feedback, to inform future iterations and improvements.

Dependencies:
References: [52, 17]

Practice Code: 1.3.10
Practice Name: Provide community website
Description: Establish a dedicated website for the research software project to serve as a central hub for project information, resources, documentation, community engagement, and collaboration.
When implemented:

- (M) A community website is developed and deployed, featuring comprehensive information about the research software project, including its objectives, features, documentation, FAQs, and contact information.
- (S) The website provides interactive features such as forums, discussion boards, or chat rooms, facilitating communication and collaboration among project contributors, users, and stakeholders.
- (S) Resources and tools relevant to the project, such as tutorials, guides, code repositories, issue trackers, and development documentation, are accessible and well-organized on the website.
- (C) The website design is user-friendly, responsive, and accessible, and reflects the project's branding and identity.

Resources required:

- Time: Significant time is required initially to plan, design, develop, and launch the community website. Ongoing time is needed for website maintenance, content updates, and community engagement efforts.
- Web development expertise or resources for building and maintaining the website, including front-end and back-end development, UI/UX design, and content management system (CMS) implementation.
- Collaboration with stakeholders, including project contributors, users, and community members, to gather input, requirements, and feedback for website features and content.
- Content creation and documentation, tutorials, and community resources.
- Promotion and outreach resources to attract visitors and encourage participation on the community website, such as social media marketing, email newsletters, and search engine optimization (SEO) strategies.

Dependencies:
References: [13, 55]

Practice Code: 2.1.2
Practice Name: Define a clear audience for the project
Description: Identify and specify the primary target audience or user base for the research software project to tailor development efforts, communication strategies, and features to meet their needs and expectations effectively.
When implemented:

- (M) The project team conducts thorough research and analysis to identify potential user groups, stakeholders, or beneficiaries of the research software project.
- (M) The primary target audience for the project is defined based on factors such as domain expertise, use case scenarios, user needs, and project objectives.
- (S) Audience personas or profiles are developed, detailing demographic information, roles, goals, challenges, and preferences of the identified user groups, to inform project decision-making and planning.
- (S) Strategies and initiatives are implemented to engage and involve the target audience actively throughout the project lifecycle, including feedback collection, user testing, and co-creation opportunities.

Resources required:

- Time: Required for conducting audience research, analyzing data, and developing audience personas. Ongoing time is needed for monitoring audience feedback, preferences, and behavior to adapt project strategies accordingly.
- Research tools and methodologies for collecting and analyzing user data, such as surveys, interviews, user testing sessions, and analytics platforms.
- Collaboration with stakeholders, domain experts, and end-users to gather insights and validate audience personas, ensuring alignment with project goals and objectives.
- Training or expertise in user research, persona development, and audience segmentation for team members involved in defining the project's target audience and designing user-centered solutions.

Dependencies:
References: [77, 71]

Practice Code: 2.1.3
Practice Name: Perform infrequent impact measurement
Description: Conduct periodic assessments to measure the impact and effectiveness of the research software project in achieving its intended goals, outcomes, and broader impacts on stakeholders and the research community.
When implemented:

- (M) Impact measurement frameworks and methodologies are established to define key performance indicators (KPIs), metrics, and evaluation criteria aligned with the project's objectives and expected outcomes.
- (M) Impact assessments are conducted at predefined intervals or project milestones using a combination of qualitative and quantitative methods, such as surveys, interviews, case studies, and data analysis.
- (S) Stakeholder engagement is emphasized throughout the impact measurement process, involving end-users, collaborators, funders, and other relevant parties to provide input, feedback, and perspectives on project impacts.
- (S) Findings from impact assessments are documented and communicated transparently to project stakeholders, informing decision-making, strategic planning, and resource allocation for future project activities.

Resources required:

- Time: Required for planning, conducting, and analyzing impact assessments, as well as for reporting findings and recommendations to stakeholders. Infrequent assessments may require dedicated time and resources for comprehensive data collection and analysis.
- Impact measurement tools and software for collecting, managing, and analyzing data, such as survey platforms, data visualization tools, and impact assessment frameworks.
- Expertise or training in impact evaluation methodologies, data analysis techniques, and reporting practices for team members responsible for conducting impact assessments.
- Collaboration with external evaluators, consultants, or research partners with expertise in impact measurement and evaluation, if necessary, to enhance rigor and objectivity in assessments and interpretation of results.

Dependencies:
References: [58, 22]

Practice Code: 2.1.7
Practice Name: Evaluate whether the audience's goals are met
Description: Assess the extent to which the research software project meets the goals, needs, and expectations of its primary audience or user base, informing iterative improvements and adjustments to enhance user satisfaction and project impact.
When implemented:

- (M) Audience goals and requirements are clearly defined and documented through user research, surveys, interviews, or other methods to establish a baseline for evaluation.
- (M) Evaluation criteria and performance indicators are established to measure the alignment between the project's features, functionality, and outcomes with the audience's goals and expectations.
- (S) Regular assessments are conducted to gather feedback from the audience, using methods such as user surveys, usability testing, user interviews, or user analytics, to check satisfaction and identify areas for improvement.
- (S) Feedback and insights from the audience evaluation are analyzed systematically, and actionable recommendations are generated to address gaps, optimize user experiences, and enhance project relevance and impact.

Resources required:

- Time: Required for planning and conducting audience evaluations, analyzing feedback, and implementing improvements based on findings. Regular evaluations may require ongoing time and resources to maintain and adapt evaluation processes.
- Evaluation tools and methods for collecting and analyzing user feedback and satisfaction data, such as surveys, usability testing kits, user analytics platforms, and qualitative research software.
- Collaboration with stakeholders, including end-users, domain experts, project sponsors, and funders, to gather input, validate findings, and prioritize improvement initiatives aligned with audience goals.
- Training or expertise in user research methodologies, survey design, usability testing, and data analysis for team members involved in evaluating audience satisfaction and project impact.

Dependencies:
References: [73, 9]

Practice Code: 2.1.9
Practice Name: Perform continuous impact measurement
Description: Implement a systematic and ongoing process to measure and assess the impact and effectiveness of the research software project throughout its life cycle, allowing for real-time feedback and continuous improvement.
When implemented:

- (M) Continuous impact measurement frameworks and methodologies are established to define key performance indicators (KPIs), metrics, and evaluation criteria aligned with the project's objectives and expected outcomes.
- (M) Automated data collection mechanisms are integrated into project workflows and systems to capture relevant impact data in real-time or at regular intervals, leveraging tools such as analytics platforms, user feedback mechanisms, usage tracking, citations, and mentions.
- (S) Data analysis and reporting processes are automated or streamlined to enable timely and actionable insights from impact data, facilitating decision-making and strategic adjustments to project activities and priorities.
- (S) Stakeholder engagement is embedded in the impact measurement process, with opportunities for ongoing feedback, input, and collaboration to ensure relevance, transparency, and accountability in impact assessment efforts.

Resources required:

- Time: Required for initial setup and configuration of impact measurement frameworks, as well as ongoing monitoring, analysis, and reporting of impact data.
- Impact measurement tools and software for collecting, managing, and analyzing data in real-time, such as analytics platforms, data visualization tools, and dash boarding solutions.
- Training or expertise in data analytics, statistical analysis, and impact measurement methodologies for team members responsible for managing and interpreting impact data.
- Collaboration with stakeholders, including end-users, project sponsors, funders, and domain experts, to define impact metrics, validate findings, and identify opportunities for improvement and innovation based on impact insights.

Dependencies:
References: [12, 65, 22]

Practice Code: 2.1.10
Practice Name: Explore new audiences regularly
Description: Proactively identify and explore potential new audiences or user groups for the research software project, expanding its reach, relevance, and impact within the broader research community and beyond.
When implemented:

- (M) Regular market research, user surveys, and analysis of user demographics and behavior are conducted to identify emerging trends, user needs, and untapped audience segments relevant to the project.
- (M) Collaboration with domain experts, industry partners, and community stakeholders is fostered to gain insights into evolving research priorities, technological advancements, and emerging areas of interest that may represent new audience opportunities.
- (S) Experimentation with targeted outreach to engage with potential new audiences, including participation in relevant conferences, workshops, webinars, and community events.
- (S) Feedback mechanisms are established to gather input and insights from new audience segments, enabling iterative refinement of project strategies, features, and communication approaches to better serve their needs and preferences.

Resources required:

- Time: Required for ongoing audience research, outreach activities, and engagement efforts with new audience segments. Regular monitoring and evaluation of audience feedback and response may necessitate dedicated time and resources for analysis and adaptation.
- Audience research tools and methods, such as surveys, focus groups, market analysis reports, and social listening tools, to gather insights into new audience demographics, behaviors, and preferences.
- Collaboration with marketing and communications teams to develop targeted messaging, content, and outreach strategies tailored to new audience segments, leveraging channels such as social media, email marketing, and content marketing.
- Training or expertise in audience research methodologies, marketing strategies, and community engagement techniques for team members involved in exploring new audiences and expanding project reach.

Dependencies:
References: [73, 9, 85]

Practice Code: 2.2.2
Practice Name: Acquire temporary funding
Description: Secures short-term financial support or grants to sustain and advance the research software project, enabling the execution of specific activities, initiatives, or development milestones within a defined timeframe.
When implemented:

- (M) Opportunities for temporary funding, such as grants, fellowships, awards, or short-term contracts, are actively sought and pursued through targeted research, networking, and engagement with funding agencies, research organizations, and industry partners.
- (M) Transparent and accountable financial management practices are maintained to ensure responsible stewardship of temporary funding resources, including budget planning, expenditure tracking, and reporting requirements as per funding agreements.
- (S) Grant proposals or funding applications are prepared and submitted in alignment with the project's goals, objectives, and funding criteria, highlighting the potential impact, innovation, and value proposition of the proposed activities or initiatives.
- (C) Collaboration and partnerships with external stakeholders, such as academic institutions, government agencies, philanthropic organizations, and industry sponsors, are leveraged to access funding opportunities, enhance credibility, and strengthen project proposals.

Resources required:

- Time: Required for identifying funding opportunities, preparing grant proposals, and managing funding applications and agreements. Additional time may be needed for reporting and compliance activities associated with temporary funding.
- Funding databases, grant directories, and funding alert services to identify and track relevant funding opportunities aligned with the project's objectives and focus areas.
- Grant writing resources and support, including templates, guidelines, and training workshops, to assist project team members in preparing competitive grant proposals and applications.
- Collaboration with institutional research offices, grant offices, or research support units to access expertise, resources, and administrative support for grant seeking and management processes.

Dependencies:
References: [77, 13]

Practice Code: 2.2.4
Practice Name: Write software management plan
Description: Develop and maintain a comprehensive Software Management Plan (SMP) outlining the policies, procedures, and guidelines for the development, deployment, maintenance, and sustainability of the research software project.
When implemented:

- (M) The SMP defines the project's software development lifecycle (SDLC), including phases such as requirements analysis, design, implementation, testing, deployment, maintenance, and retirement, with clear roles, responsibilities, and workflows for each phase.
- (M) Policies and procedures for software documentation, version control, QA, licensing, security, data management, and intellectual property rights are documented in the SMP, ensuring consistency, transparency, and accountability in software management practices.
- (S) The SMP is developed in collaboration with relevant stakeholders, including project team members, research administrators, funding agencies, and institutional partners, to ensure alignment with project goals and compliance requirements.
- (S) The SMP includes strategies and mechanisms for sustainability planning, resource allocation, and risk management to ensure the long-term viability and impact of the research software project beyond the initial funding period.

Resources required:

- Time: Required for developing, reviewing, and updating the SMP throughout the project lifecycle. Collaboration with stakeholders and coordination of input may necessitate dedicated time and resources for SMP development and maintenance.
- SMP templates, guidelines, and best practices from funding agencies, research organizations, and community initiatives such as the Software Sustainability Institute (SSI) or the researchers / research software engineers community, to inform SMP development and implementation.
- Training or workshops on software management planning, software engineering best practices, and compliance requirements for project team members involved in SMP development and implementation.

Dependencies:
References: [41, 77, 57]

Practice Code: 2.2.7
Practice Name: Obtain support from a national research software center
Description: Establish partnerships or collaborations with national research software centers or organizations focused on research software development, providing access to resources, expertise, and support to advance the research software project.
When implemented:

- (M) Identifies relevant national research software centers or organizations with a mission aligned with its goals, objectives, or domain expertise.
- (S) Outreach and engagement efforts are initiated to establish connections and build relationships with key stakeholders, decision-makers, and technical experts at the national research software center, highlighting the value proposition and potential synergies of collaboration.
- (S) Discussions and negotiations are conducted to explore potential forms of support or collaboration, such as funding opportunities, technical assistance, infrastructure access, training and capacity building, or dissemination and outreach activities.
- (S) Formal agreements or partnerships are established with the national research software center, outlining roles, responsibilities, expectations, and mutual benefits for both parties, with clear mechanisms for communication, collaboration, and governance.

Resources required:

- Time: Required for identifying suitable national research software centers, initiating and nurturing relationships, and negotiating partnership agreements. Ongoing time and effort are needed for collaboration coordination, communication, and relationship management.
- Research and networking resources to identify and research potential national research software centers, including online directories, professional networks, and conference events.
- Collaboration platforms and communication tools for facilitating discussions, sharing information, and coordinating activities between the research software project team and the national research software center.
- Training or guidance on partnership development, negotiation skills, and effective communication strategies for team members involved in engaging with national software centers and establishing collaborative relationships.

Dependencies:
References: [35, 82]

Practice Code: 2.2.8
Practice Name: Acquire viable pathways for project sustainability
Description: Identify and secure sustainable pathways to ensure the long-term viability, funding, and impact of the research software project beyond the initial development phase, enabling continued support, maintenance, and evolution.
When implemented:

- (M) Comprehensive sustainability planning is conducted early in the project lifecycle, considering various factors such as funding sources, community engagement, partnerships, and institutional support.
- (M) Opportunities for diversifying funding streams are explored, including grant funding, industry partnerships, fee-based services, licensing revenue, crowdfunding, or membership models, to mitigate dependency on a single funding source.
- (S) Collaboration with stakeholders, including research institutions, funding agencies, industry partners, and user communities, is fostered to co-create sustainable business models, governance structures, and revenue-sharing arrangements aligned with project goals and values.
- (S) Mechanisms for community engagement, user support, and contributions are established to foster a sense of ownership, participation, and investment among stakeholders, ensuring ongoing user adoption, feedback, and contributions to sustain the project ecosystem.

Resources required:

- Time: Required for conducting sustainability assessments, planning activities, and engaging with stakeholders to develop and implement sustainable pathways.
- Expertise or consultation from financial analysts, legal advisors, and fundraising professionals to assess sustainability options, and navigate legal and regulatory considerations.
- Collaboration with institutional research centers, technology centers, or innovation hubs to access resources, expertise, and support services for commercialization, licensing, and intellectual property management.
- Training or capacity building for project team members on sustainability planning, and community engagement strategies to build internal capacity and resilience for sustaining the project's impact and value proposition.

Dependencies:
References: [41, 50]

Practice Code: 2.2.9
Practice Name: Secure continuous funding
Description: Establish mechanisms to secure ongoing financial support and resources to sustain the RS project over the long term, ensuring its continued development, maintenance, and impact.
When implemented:

- (M) The project team conducts strategic planning and forecasting to assess the funding needs and sustainability requirements of the research software project, considering factors such as personnel costs, infrastructure expenses, and operational overheads.
- (M) Diverse funding sources and opportunities are explored and pursued, including government grants, private donations, corporate sponsorships, institutional support, philanthropic funding, and revenue-generating activities, to diversify and stabilize the project's financial base.
- (S) Grant proposals, funding applications, or sponsorship pitches are prepared and submitted regularly, leveraging project achievements, milestones, and impact metrics to demonstrate the value proposition and attract funding partners.
- (S) Relationships with funding agencies, donors, sponsors, and partners are nurtured and maintained through regular communication, reporting, and engagement activities, fostering trust, transparency, and alignment of interests.

Resources required:

- Time: Required for researching funding opportunities, preparing funding applications, and engaging with potential funders or sponsors. Ongoing time and effort are needed for relationship management, reporting, and compliance activities associated with continuous funding.
- Expertise or consultation from grant writers, fundraising professionals, financial analysts, and legal advisors to develop funding strategies, craft compelling proposals, and negotiate funding agreements effectively.
- Collaboration with institutional research offices, development offices, or grant support units to access resources, training, and support services for grant seeking, proposal development, and compliance management.

Dependencies:
References: [77, 13, 4]

Practice Code: 2.2.10
Practice Name: Define end-of-life policy
Description: Establish clear guidelines and procedures for managing the end-of-life phase of the research software project, including discontinuation, and archival of project assets to ensure responsible stewardship of resources and continuity for users.
When implemented:

- (M) An end-of-life policy is developed in consultation with project stakeholders, including researchers / research software engineers, users, funders, and institutional partners, to define criteria, timelines, and responsibilities for discontinuing the research software project.
- (M) Criteria for determining end-of-life triggers, such as obsolescence, lack of funding, loss of support, or achievement of project goals, are specified in the policy, providing clarity and transparency for stakeholders.
- (C) Responsibilities for managing end-of-life activities, such as communication, documentation, resource reallocation, and stakeholder engagement, are assigned to designated individuals or teams within the project organization, ensuring accountability and continuity in execution.

Funder
Resources required:

- Time: Required for developing and documenting the end-of-life policy. Ongoing time and effort may be required for monitoring and updating the policy periodically to reflect changing project circumstances.
- Collaboration with legal counsel, institutional policy experts, and regulatory compliance officers to ensure alignment with legal and regulatory requirements governing data retention, privacy, intellectual property rights, and contractual obligations.
- Training or guidance for project team members involved in end-of-life planning and execution, including communication strategies, stakeholder engagement techniques, and archival best practices for preserving project assets and knowledge.

Dependencies: * (all other practices) $<$ 2.2.10
References: [30, 36]

Practice Code: 2.3.1
Practice Name: Make code citable
Description: Implement mechanisms to assign Digital Object Identifiers (DOIs) or other persistent identifiers to the research software project's code repositories, enabling proper citation and recognition of the software as a scholarly contribution.
When implemented:

- (M) The research software project establishes a policy or procedure for assigning DOIs or other persistent identifiers to specific releases, versions, or milestones of the project's codebase stored in public repositories.
- (S) Integration with code hosting platforms or repository services, such as GitHub, GitLab, or Zenodo, is configured to enable automatic assignment and registration of DOIs for tagged releases or stable versions of the software.
- (C) Metadata associated with the DOI registration includes essential information about the software, such as title, description, version history, licensing information, relevant publications or documentation, and researchers / research software engineers name, to facilitate accurate citation and attribution.
- (C) Guidelines and recommendations for citing the software code are provided to users and contributors, outlining citation formats, preferred attribution practices, and acknowledgment requirements to promote proper citation in scholarly works and publications.

Resources required:

- Time: Required for configuring DOI registration workflows, updating metadata, and disseminating citation guidelines to users and contributors.
- Integration with DOI registration services or repository platforms that support automatic DOI assignment for code repositories, such as Zenodo's GitHub integration or DataCite's repository services, to streamline the process of making code citable.
- Collaboration with institutional librarians, repository managers, or scholarly communication specialists to ensure compliance with best practices and standards for data citation, metadata quality, and citation metrics for software publications.

Dependencies:
References: [12, 65]

Practice Code: 2.3.3
Practice Name: Enable indexing of project meta-data
Description: Implement mechanisms to ensure that project meta-data, including descriptive information about the RS project, is indexed and discoverable by relevant search engines, repositories, and discovery platforms, enhancing visibility and accessibility.
When implemented:

- (M) Comprehensive meta-data for the research software project is structured according to relevant standards and best practices, including information about the project's purpose, scope, contributors, funding sources, version history, documentation, and related publications.
- (M) Meta-data is encoded using machine-readable formats, such as JSON-LD, RDF/XML, or schema.org markup, to facilitate automated indexing and interpretation by search engine crawlers, repository harvesters, and metadata aggregators.
- (S) Integration with indexing services, discovery platforms, or repository networks, such as Google Scholar, BASE, or Research Gate, is established to enable automatic indexing and updating of project meta-data, ensuring visibility and accessibility to a wider audience.
- (C) Regular monitoring and maintenance of meta-data quality and consistency are conducted to address errors, gaps, or outdated information, ensuring accurate representation and discovery of the research software project across indexing platforms.

Resources required:

- Time: Required for encoding, and updating project meta-data according to relevant standards and guidelines.
- Knowledge or expertise in meta-data standards, such as Dublin Core, schema.org, or DataCite meta-data schema, for structuring and encoding project meta-data effectively to support indexing and discovery.
- Collaboration with repository managers, technical experts, or metadata librarians to ensure compatibility and compliance with indexing protocols, metadata schemas, and indexing service requirements for research software projects.

Dependencies:
References: [12, 40]

Practice Code: 2.3.4
Practice Name: Promote the project continuously
Description: Implement ongoing promotional activities to raise awareness, generate interest, and engage stakeholders with the research software project, enhancing visibility, adoption, and impact.
When implemented:

- (M) A comprehensive promotional strategy is developed, outlining goals, target audiences, key messages, channels, and tactics for promoting the research software project effectively across various platforms and communities.
- (M) Regular communication channels and engagement touch points, such as project websites, blogs, social media profiles, mailing lists, newsletters, and community forums, are established to share updates, announcements, achievements, and resources related to the project.
- (S) Outreach efforts are conducted proactively to engage with relevant communities, user groups, professional networks, and media outlets, leveraging opportunities such as conferences, workshops, webinars, and publications to showcase the project's value proposition and impact.

Resources required:

- Time: Required for planning, executing, and monitoring promotional activities, including content creation, community engagement, and relationship building efforts.
- Content creation resources, including writing, design, multimedia production, and editing skills, to develop compelling and engaging promotional materials such as blog posts, videos, infographics, and social media content.
- Collaboration with graphic designers, researchers / research software engineers, and marketing professionals to enhance the visual appeal, usability, and effectiveness of promotional materials and communication channels for the research software project.
- Training or guidance for project team members on communication strategies, social media best practices, and community engagement techniques to effectively promote the project's mission, achievements, and impact on diverse audiences.

Dependencies:
References: [56, 77]

Practice Code: 2.3.5
Practice Name: Publish in a research software directory
Description: Publishing research software projects in a research software directory facilitates discovery, citation, and collaboration among researchers, research software engineers, and users within the research community.
When implemented:

- (M) Identification and selection of suitable research software directories that align with the project's focus area, target audience, and visibility goals, considering factors such as reputation, coverage, and accessibility.
- (M) Preparation and submission of comprehensive meta-data and documentation for the research software project to the selected directory.
- (S) Compliance with directory-specific submission guidelines, metadata standards, and QA criteria to ensure accurate representation, indexing, and discoverability of the research software project within the directory's database and search interface.
- (C) Monitoring and updating of project listings and meta-data in the research software directory as needed to reflect changes, updates, or new releases of the software

Resources required:

- Time: Required for researching and selecting appropriate research software directories, preparing and submitting project listings, and maintaining directory entries over time. Ongoing time and effort may be needed for monitoring directory performance, responding to inquiries, and updating project information.
- Knowledge or expertise in directory submission processes, metadata standards, and documentation requirements for research software projects
- Collaboration with research software directory team to publish projects on their website.

Dependencies:
References: [60]

Practice Code: 2.3.6
Practice Name: Acquire research software center acknowledgment
Description: Seek acknowledgment or recognition from reputable software centers or repositories specialized in research software, validating the quality, impact, and scholarly value of the research software project within the academic community.
When implemented:

- (M) Identification and selection of recognized software centers or repositories known for their expertise in indexing, and promoting high-quality research software projects, aligning with the project's scope, domain, and target audience.
- (M) Submission of the research software project to the selected software center or repository for evaluation, review, and potential inclusion in their catalog, following their submission guidelines and quality criteria.
- (S) Collaboration with software center representatives to provide additional information, documentation, or demonstration of the RS project's significance, novelty, and utility to support its evaluation and selection process.
- (C) Receipt of acknowledgment, certification, or endorsement from the software center acknowledging the research software project's inclusion, accreditation, or recommendation, signifying its quality, relevance, and scholarly impact within the research community.

Resources required:

- Time: Required for researching and identifying reputable software centers, preparing submission materials, and engaging with software center representatives throughout the evaluation and acknowledgment process. Ongoing time and effort may be needed for communication, follow-up, and compliance with software center requirements.
- Expertise or consultation on software center submission processes, evaluation criteria, and QA standards to ensure alignment and compliance with submission guidelines and expectations.
- Collaboration with institutional partners, domain experts, or collaborators with existing relationships or affiliations with software centers to leverage networking opportunities, advocacy, and support for acquiring acknowledgment or recognition.

Dependencies:
References: [75]

Practice Code: 2.3.7
Practice Name: Enable indexing of the project's source code
Description: Facilitate the indexing and preservation of the research software project's source code in platforms like Software Heritage Graph, enhancing its visibility, traceability, and long-term accessibility for future research and collaboration.
When implemented:

- (M) Identification and selection of suitable platforms or repositories for indexing and archiving the project's source code, with a focus on reputable, trustworthy platforms known for their commitment to software preservation and provenance tracking.
- (M) Integration of the project's source code repository with the selected indexing platform, leveraging features such as repository synchronization, metadata extraction, and version tracking to ensure comprehensive coverage and accuracy of code indexing.
- (S) Compliance with platform-specific requirements, standards, and APIs for submitting and indexing source code repositories, including metadata enrichment, licensing information, and versioning metadata, to facilitate discovery and attribution of the project's source code.
- (S) Regular monitoring and maintenance of indexed code repositories on the platform, including updates, revisions, and new releases of the project's source code, to ensure currency, completeness, and integrity of the indexed data over time.

Resources required:

- Time: Required for researching and selecting suitable indexing platforms, configuring integration settings, and monitoring indexing performance. Ongoing time and effort may be needed for maintenance, updates, and troubleshooting of indexed repositories.
- Technical expertise in version control systems (e.g., Git, SVN), repository management platforms (e.g., GitHub, GitLab), and indexing APIs or protocols (e.g., Software Heritage API) for integrating and synchronizing source code repositories with indexing platforms.
- Collaboration with platform administrators, technical support teams, or community contributors to resolve integration issues, optimize indexing settings, and ensure compliance with platform policies and guidelines for source code submission and preservation.

Dependencies:
References: [77, 39]

Practice Code: 2.3.10
Practice Name: Garner industrial partner adoption
Description: Cultivate relationships and collaborations with industrial partners to promote the adoption and utilization of the research software project within industry settings, fostering technology transfer, innovation, and commercialization opportunities.
When implemented:

- (M) Identification and outreach to potential industrial partners, including companies, startups, and industry consortia, that can benefit from leveraging the research software project's capabilities, expertise, or technologies to address industry challenges or enhance product development.
- (M) Collaboration agreements or partnership arrangements are established with industrial partners, defining roles, responsibilities, expectations, and mutual benefits of collaboration, such as joint development projects, technology licensing, and knowledge exchange.
- (S) Engagement with industrial partners through networking events, industry conferences, workshops, or targeted outreach campaigns to raise awareness about the research software project, showcase its value proposition, and explore collaboration opportunities aligned with industry needs and priorities.
- (C) Ongoing communication, support, and collaboration with industrial partners to facilitate technology transfer, integration, and adoption of the research software project within industrial workflows, providing training, technical assistance, and customization services as needed to address industry-specific requirements and use cases.

Resources required:

- Time: Required for identifying, engaging, and nurturing relationships with industrial partners, as well as for negotiating and managing collaboration agreements and supporting partner onboarding and adoption efforts. Ongoing time and effort may be needed for relationship management, collaboration support, and follow-up activities with industrial partners.
- Networking and outreach resources, such as industry databases, trade associations, and business networks, to identify and connect with potential industrial partners and decision-makers interested in collaborating with research software projects.

Dependencies:
References: [56, 77, 4]

Practice Code: 2.4.4
Practice Name: Analyze privacy usage impact
Description: Conduct a comprehensive analysis to assess the privacy implications and usage impact of the research software project, evaluating data collection, processing, storage, and sharing practices to ensure compliance with privacy regulations and ethical principles.
When implemented:

- (M) Identification and documentation of all data collection points, including user interactions, system logs, telemetry, or third-party integrations, within the research software project to understand the scope and scale of data processing activities.
- (M) Implementation of privacy-enhancing measures and safeguards, such as data anonymization, encryption, access controls, audit trails, and user consent mechanisms, to mitigate identified privacy risks and protect individuals' privacy rights and interests.
- (S) Conducting a privacy impact assessment (PIA) or data protection impact assessment (DPIA) to systematically evaluate the potential risks and consequences associated with data processing activities, considering factors such as data sensitivity, user consent, data retention, and data sharing practices.
- (C) Transparency and accountability in communicating privacy practices and policies to users, stakeholders, and regulatory authorities through privacy notices, terms of service, data protection agreements, or privacy impact statements, ensuring informed consent and trust in the project's data handling practices.

Resources required:

- Time: Required for conducting privacy impact assessments, reviewing privacy policies, and implementing privacy-enhancing measures within the research software project. Ongoing time and effort may be needed for monitoring, auditing, and updating privacy practices in response to evolving regulatory requirements and user expectations.
- Collaboration with stakeholders, including users, data subjects, project sponsors, and regulatory authorities, to gather input, address concerns, and build trust in the project's privacy practices and compliance efforts through transparent and accountable data handling practices.

Dependencies:
References: [19]

Practice Code: 2.4.5
Practice Name: Analyze ethical consequences of project use
Description: Conduct a systematic analysis to evaluate the ethical implications and consequences of using the research software project, considering potential risks, benefits, and societal impacts on individuals, communities, and society as a whole.
When implemented:

- (M) Identification and documentation of potential ethical issues and dilemmas associated with the research software project, including concerns related to privacy, security, fairness, bias, discrimination, accountability, transparency, and societal impact.
- (M) Integration of ethical principles, guidelines, and safeguards into the design, development, deployment, and usage of the research software project, such as ethical design practices, responsible data handling, user empowerment, and stakeholder engagement mechanisms.
- (S) Engagement with relevant stakeholders, including users, domain experts, ethicists, community representatives, and affected parties, to gather diverse perspectives, insights, and concerns about the ethical implications of project use across different contexts and user groups.
- (C) Conducting an ethical impact assessment or ethical review process to systematically evaluate and prioritize ethical considerations, weighing potential risks against benefits, and identifying strategies to mitigate or address ethical challenges effectively.

Resources required:

- Time: Required for conducting ethical impact assessments, engaging stakeholders, and integrating ethical considerations into project planning and execution. Ongoing time and effort may be needed for monitoring, evaluating, and adapting ethical practices in response to evolving ethical standards and community feedback.
- Expertise or consultation from ethicists, social scientists, policy experts, and community representatives to guide ethical analysis, decision-making, and stakeholder engagement processes, ensuring alignment with ethical principles and values relevant to the project's context and domain.
- Collaboration with institutional review boards, ethics committees, or regulatory authorities to ensure compliance with ethical standards, regulations, and guidelines governing research involving human subjects or sensitive data, as applicable to the RS project.

Dependencies:
References: [13, 33]

Practice Code: 2.4.6
Practice Name: Document the cost of running the application
Description: Record and document the various costs associated with running the application, including infrastructure expenses, licensing fees, personnel costs, maintenance, and other operational expenditures, to facilitate budgeting, financial planning, and resource allocation.
When implemented:

- (M) Identification and documentation of all direct and indirect costs related to running the application, including hardware costs, cloud computing services, software licenses, subscription fees, and personnel expenses such as salaries, benefits, and training.
- (M) Implementation of cost tracking mechanisms, such as expense tracking software, financial spreadsheets, or budgeting tools, to record and categorize costs accurately, ensuring transparency, accountability, and traceability of expenditures.
- (S) Integration of cost documentation and reporting into project management and decision-making processes, providing stakeholders with timely and accurate cost information to support strategic planning, investment decisions, and project prioritization.
- (C) Regular review and analysis of cost data to assess cost trends, identify cost-saving opportunities, and optimize resource utilization, such as optimizing cloud resource usage, renegotiating vendor contracts, or streamlining operational workflows.

Resources required:

- Time: Required for documenting, tracking, and analyzing costs associated with running the application, including data collection, analysis, and reporting activities. Ongoing time and effort may be needed for regular updates, reviews, and adjustments to cost documentation and budget forecasts.
- Tools or software for expense tracking, budgeting, and financial reporting, such as accounting software, spreadsheet applications, or cloud-based financial management platforms, to streamline cost documentation and analysis processes.
- Expertise or consultation from financial analysts, accountants, or budget managers to develop cost documentation processes, establish cost tracking metrics and interpret financial data to support informed decision-making and resource optimization.

Dependencies:
References: [21, 16]

Practice Code: 2.4.7
Practice Name: Consider total energy consumption
Description: Evaluate and document the energy consumption associated with running the application, including electricity usage by servers, data centers, networking equipment, server cost, and other infrastructure components, to understand the environmental impact and promote sustainability.
When implemented:

- (M) Measurement and monitoring of energy consumption metrics, such as power usage effectiveness (PUE), energy usage effectiveness (EUE), or carbon emissions, associated with the operation of the application infrastructure, including both direct and indirect energy usage.
- (M) Integration of energy consumption considerations into decision-making processes, such as infrastructure design, capacity planning, and technology selection, to prioritize energy-efficient solutions and minimize the carbon footprint of running the application.
- (S) Analysis of energy consumption data to identify energy-intensive components, processes, or configurations within the application infrastructure, such as inefficient hardware, cooling systems, or workload distribution patterns, that contribute to excessive energy usage.
- (C) Implementation of energy efficiency measures and optimization strategies to reduce energy consumption and minimize environmental impact, such as server virtualization, workload consolidation, hardware upgrades, or utilization of renewable energy sources.

Resources required:

- Time: Required for collecting, analyzing, and interpreting energy consumption data, as well as for implementing and monitoring energy efficiency measures within the application infrastructure. Ongoing time and effort may be needed for continuous monitoring, optimization, and reporting of energy consumption.
- Tools or software for energy monitoring and management, such as energy management systems, power monitoring tools, or environmental monitoring sensors, to track and analyze energy usage metrics across the application infrastructure.
- Expertise or consultation from environmental engineers, energy consultants, or sustainability specialists to develop energy consumption monitoring strategies, identify energy-saving opportunities, and implement sustainable practices within the application environment.

Dependencies:
References: [26]

Practice Code: 3.1.4
Practice Name: Acknowledge partners and funding agencies on website
Description: Recognize and publicly acknowledge the contributions and support provided by partners and funding agencies to the research software project by prominently featuring their logos, names, or testimonials on the project website, demonstrating gratitude and fostering transparency and accountability in project operations.
When implemented:

- (M) Compilation of a list of all partners, collaborators, and funding agencies that have contributed resources, expertise, or financial support to the research software project, including details such as organization names, logos, funding amounts, project duration, and collaboration activities.
- (S) Creation of a dedicated "Partners" or "Acknowledgments" section on the project website, showcasing logos, names, and brief descriptions of partners and funding agencies, along with links to their websites or contact information for further inquiries or collaborations.
- (C) Regular updates and maintenance of the partners and funding agencies section on the website to reflect new partnerships, collaborations, or funding awards, ensuring accuracy, completeness, and currency of acknowledgment information over time.
- (C) Engagement with partners and funding agencies to obtain consent for acknowledgment and approval of logo usage on the project website, respecting branding guidelines, copyright policies, and privacy preferences, and providing opportunities for partners to provide testimonials or endorsements if desired.

Resources required:

- Time: Required for compiling acknowledgment information, updating the website content, and obtaining approvals from partners and funding agencies for acknowledgment and logo usage. Ongoing time and effort may be needed to maintain and update the acknowledgment section as new partnerships and funding are secured.
- Collaboration and communication tools for coordinating with partners, funding agencies, and internal stakeholders to gather acknowledgment information, obtain approvals, and ensure compliance with branding guidelines and legal requirements for logo usage and acknowledgment practices.

Dependencies:
References: [77, 4]

Practice Code: 3.1.7
Practice Name: Develop advanced partnership model
Description: Develop and implement an advanced partnership model that goes beyond standard collaborations, fostering strategic, long-term relationships with partners that involve shared goals, joint decision-making, resource pooling, and mutual benefits, to drive innovation, sustainability, and impact in the research software project.
When implemented:

- (M) Alignment of partner goals and objectives with the mission, vision, and strategic priorities of the research software project, identifying synergies, complementarities, and shared interests for collaboration.
- (M) Collaboration with partners in co-creating, co-developing, and co-innovating research software solutions, products, or services, leveraging partner expertise, resources, and networks to address common challenges and opportunities.
- (M) Establishment of governance structures, such as joint steering committees, advisory boards, or collaborative forums, to facilitate communication, coordination, and decision-making among partners.
- (S) Sharing of resources, including funding, infrastructure, expertise, and intellectual property, among partners to maximize efficiency, leverage economies of scale, and accelerate progress towards shared goals and outcomes.
- (S) Focus on creating mutual value and benefits for all partners involved.

Resources required:

- Time: Required for developing, negotiating, and managing advanced partnerships, including conducting partner assessments, strategic planning, partnership agreement negotiations, and ongoing partnership management activities.
- Adequate resources and infrastructure are needed to support partnership activities, such as dedicated staff, collaboration tools, communication platforms, and meeting spaces, as well as financial resources for funding joint initiatives, projects, or activities agreed upon by partners.

Dependencies:
References: [56, 13, 4]

Practice Code: 3.2.3
Practice Name: Impose community norms
Description: Establish and enforce community norms, guidelines, and codes of conduct to define acceptable behavior, foster inclusivity, respect, and professionalism, and maintain a positive and supportive environment within the research software project community.
When implemented:

- (M) Collaborative development of community norms, guidelines, or codes of conduct through consultation with community members, stakeholders, and relevant experts, incorporating principles of inclusivity, diversity, respect, and professionalism.
- (M) Clear communication and dissemination of community norms to all community members through multiple channels, such as project websites, documentation, forums, mailing lists, and social media platforms, ensuring accessibility, visibility, and awareness of expected behaviors and standards.
- (S) Establishment of clear procedures and mechanisms for enforcing community norms, including reporting mechanisms, and disciplinary actions for addressing violations or breaches of conduct, ensuring accountability and fairness in addressing community concerns.
- (S) Regular engagement with community members to promote awareness, understanding, and adherence to community norms through training, workshops, discussions, and educational materials, fostering a culture of mutual respect, empathy, and responsibility.

Resources required:

- Time: Required for developing, communicating, and enforcing community norms, as well as for engaging with community members, addressing concerns, and resolving conflicts. Ongoing time and effort may be needed to monitor community dynamics, update norms, and provide support to community moderators.
- Expertise and skills: Expertise in community management, conflict resolution, and diversity and inclusion practices, as well as strong communication, interpersonal, and leadership skills, are essential for effectively imposing community norms and fostering a healthy and supportive community environment.

Dependencies:
References: [41, 77, 64]

Practice Code: 3.2.4
Practice Name: Onboard researchers as part of the community
Description: Facilitate the seamless integration and active participation of researchers into the research software project community by providing structured onboarding processes, resources, and support to foster engagement, collaboration, and contribution.
When implemented:

- (M) Developing comprehensive onboarding materials, including welcome guides, orientation videos, documentation, and tutorials, to introduce researchers to the research software project, its objectives, governance structure, community norms, and available resources.
- (M) Conducting onboarding sessions, workshops, webinars, or meet-and-greet events to welcome new researchers, provide an overview of the project's mission, vision, and activities, and facilitate introductions to key community members, contributors, and stakeholders.
- (M) Assigning mentors or onboarding buddies to new researchers to provide guidance, and support, answering questions, and facilitating connections with relevant working groups or initiatives.
- (S) Offering various opportunities for researchers to engage with the community and contribute to the project's goals.

Resources required:

- Time: Required for planning, organizing, and delivering onboarding activities, as well as for providing ongoing support and mentorship to new researchers.
- Expertise in community management, communication, and mentorship, as well as strong interpersonal, organizational, and facilitation skills, are essential for effectively onboarding researchers and fostering their integration into the community.
- Development of onboarding materials, resources, and tools, such as welcome packs, orientation guides, online tutorials, and mentorship frameworks, to support new researchers in understanding the project's goals, processes, and expectations.

Dependencies:
References: [11]

Practice Code: 3.2.5
Practice Name: Develop code of conduct
Description: Establish and enforce a code of conduct that outlines expected behavior, standards, and guidelines for all community members participating in the research software project, promoting inclusivity, respect, professionalism, and a safe and welcoming environment for collaboration and interaction.
When implemented:

- (M) Collaborative development of a code of conduct through consultation with community members, stakeholders, and relevant experts, incorporating principles of inclusivity, diversity, respect, integrity, and professionalism.
- (M) Clear communication and dissemination of the code of conduct to all community members through multiple channels, such as project websites, documentation, forums, mailing lists, and social media platforms, ensuring visibility, accessibility, and awareness of expected behaviors and standards.
- (S) Establishment of clear procedures and mechanisms for enforcing the code of conduct
- (S) Regular education and awareness efforts to promote understanding and adherence to the code of conduct among community members.

Resources required:

- Time: Required for developing, communicating, and enforcing the code of conduct, as well as for educating community members and addressing concerns or violations. Ongoing time and effort may be needed to monitor community dynamics, update the code of conduct, and provide support to community moderators.
- Expertise in community management, conflict resolution, and diversity and inclusion practices, as well as strong communication, interpersonal, and leadership skills, are essential for effectively implementing and enforcing the code of conduct.

Dependencies:
References: [77, 13]

Practice Code: 3.2.6
Practice Name: Appoint support team
Description: Appoint a dedicated support team responsible for providing assistance, guidance, and technical support to community members, users, and contributors of the research software project, ensuring timely resolution of issues, effective communication, and a positive user experience.
When implemented:

- (M) Identification and selection of individuals with appropriate expertise, knowledge, and communication skills to serve as support team members, including technical support specialists, community moderators, and subject matter experts.
- (M) Clearly define the roles, responsibilities, and expectations of support team members, including response times, communication channels, and issue resolution workflows, ensuring alignment with project goals and community needs.
- (S) Provide training, resources, and onboarding for support team members to familiarize them with the research software project, its features, functionality, and common issues, as well as best practices for providing effective support and engaging with community members.
- (C) Establish communication channels, collaboration platforms, and workflows for coordinating support activities, sharing knowledge and insights, and escalating complex or unresolved issues to appropriate stakeholders or researchers / research software engineers, for resolution.

Resources required:

- Time: Required for recruiting, training, and coordinating support team members, as well as for responding to support requests, addressing issues, and monitoring support activities.
- Support team members should possess expertise in relevant technologies, software tools, and domain knowledge.
- Provision of training materials, knowledge bases, FAQs, tutorials, and documentation resources to support team members and community members, providing guidance on common issues, troubleshooting steps, and best practices for using the research software project effectively.

Dependencies: 3.2.6 < 3.2.9
References: [56, 70]

Practice Code: 3.2.7
Practice Name: Organize community events
Description: Plan, coordinate, and host community events, such as workshops, hackathons, webinars, conferences, or meetups, to foster engagement, collaboration, knowledge sharing, and networking among community members involved in the research software project.
When implemented:

- (M) Develop event plans, agendas, and timelines outlining event objectives, themes, formats, speakers, and logistics, considering community preferences, interests, and feedback gathered through surveys, polls, or previous events.
- (M) Promote community events through various channels, such as project websites, social media platforms, mailing lists, and partner networks, using targeted marketing strategies, event invitations, and promotional materials to attract participants and generate interest.
- (S) Facilitate interactive and participatory sessions during community events, such as presentations, panel discussions, workshops, hands-on tutorials, or networking sessions, to encourage active engagement, collaboration, and knowledge exchange among participants.
- (C) Feedback evaluations from event participants to assess event satisfaction, relevance, and impact, gathering insights, suggestions, and lessons learned for improving future event planning and execution.

Resources required:

- Time: Required for planning, organizing, and executing community events, as well as for promoting events, coordinating logistics, and facilitating event activities.
- Budget for venue rentals, catering, equipment rentals, marketing materials, and other event-related expenses.
- Dedicated staff or volunteers to assist with event planning, and coordination, as well as to serve as speakers, facilitators, or moderators during event sessions.
- Collaboration with event sponsors, partners, or collaborators to leverage their resources, expertise, and networks in organizing and hosting community events.

Dependencies:
References: [77, 72]

Practice Code: 3.2.9
Practice Name: Provide front page chat support
Description: Offer real-time chat support directly on the front page or landing page of the research software project website to assist visitors, answer inquiries, address issues, and provide guidance or assistance in navigating the website, accessing resources, or learning more about the project.
When implemented:

- (M) Embedding a chat support widget or plugin directly on the front page of the project website, allows visitors to initiate chat conversations with support agents or community moderators without navigating to separate support pages or channels.
- (M) Assigning dedicated support agents or community moderators to monitor and respond to chat inquiries during specified hours of operation, ensuring prompt and helpful assistance to visitors in real time.
- (S) Integrating chat support with a knowledge base or FAQ section on the website to provide quick access to commonly asked questions, troubleshooting tips, and self-help resources, enabling support agents to efficiently address inquiries and provide relevant information.
- (C) Establishing escalation procedures and protocols for handling complex or unresolved inquiries via chat support, including routing inquiries to appropriate specialists, scheduling follow-up communications, or transitioning to alternative support channels as needed.

Resources required:

- Time: Required for setting up and configuring the chat support system.
- Subscription or licensing for chat support software or platforms that offer features such as real-time messaging, chat routing, canned responses, chat history tracking, and integration with website analytics and CRM systems.
- Provision of training materials, guidelines, and documentation for chat support agents on best practices, communication techniques, project knowledge, and issue resolution strategies to deliver effective and consistent support experiences.

Dependencies:
References: [23]

Practice Code: 3.2.10
Practice Name: Focus on diversity and inclusion
Description: Prioritize diversity and inclusion in all aspects of research software development to ensure that the software meets the needs of diverse users, respects diverse perspectives, and promotes equity and inclusion in research and innovation.
When implemented:

- (M) Integrate inclusive design principles into the software development process, considering diverse user needs, preferences, abilities, and contexts throughout the design, development, and testing phases to create software that is accessible, usable, and equitable for all users.
- (S) Incorporate cultural sensitivity, inclusivity, and diversity considerations into the software interface, content, language, imagery, and user experience design, avoiding stereotypes, biases, or exclusionary language, and promoting respect, representation, and inclusion of diverse cultures, identities, and perspectives.
- (C) Ensure diverse representation and participation of stakeholders, end users, and subject matter experts from underrepresented groups in the researchers / research software engineers, user research activities, usability testing, and feedback sessions to incorporate diverse perspectives and insights into the software design and development process.

Resources required:

- Time: Required for implementing accessibility features and inclusive training sessions.
- Provide training and resources on diversity, inclusion, cultural competence, and accessibility for researchers / research software engineers, designers, and other team members involved in the development process, fostering awareness, empathy, and skills to address diversity-related challenges and opportunities.
- Conduct user research, usability testing, and user feedback sessions with diverse groups of users, including individuals from underrepresented communities, minority groups, or marginalized populations, to identify barriers, preferences, and needs and to validate design decisions and software improvements.

Dependencies:
References: [77, 67]

Practice Code: 3.3.2
Practice Name: Make developer names and roles publicly available
Description: Provide transparency and visibility into the researchers / research software engineers by making the names and roles of researchers / research software engineers publicly available on the research software project's website or documentation, fostering accountability, recognition, and trust among community members and stakeholders.
When implemented:

- (M) Acknowledge and attribute contributions made by researchers / research software engineers to the research software project, including code contributions, documentation updates, bug fixes, feature enhancements, or community engagement activities, providing recognition and visibility for their efforts and contributions.
- (S) Publish researchers / research software engineers profiles on the project website, team page, or contributors section, making them easily accessible to community members, users, collaborators, and potential contributors who are interested in learning more about the individuals behind the project and their respective roles and responsibilities.
- (C) Create individual profiles or bios for each researchers / research software engineers involved in the RS project, including their name, role, expertise, background, and contributions to the project, showcasing the diverse skills and experiences of the researchers / research software engineers.
- (C) Maintain researchers / research software engineers profiles and team information up to date with changes in team composition, roles, or contributions, ensuring accuracy, relevance, and completeness of researchers / research software engineers information over time, and reflecting the dynamic nature of the researchers / research software engineers.

Resources required:

- Time: Required for creating and maintaining the researchers / research software engineers list on the project website or documentation.
- Utilize website content management systems (CMS), documentation platforms, or version control repositories to create and publish researchers / research software engineers profiles, team pages, or contributors sections on the project website, enabling easy access and navigation for users.
- Design visually appealing and user-friendly researchers / research software engineers profile templates or layouts, incorporating features such as photos, bios, social media links, and contribution metrics to enhance engagement and recognition of researchers / research software engineers within the community.

Dependencies:
References: [41, 76]

Practice Code: 3.3.6
Practice Name: Document how to join the team
Description: Provide clear and accessible guidelines, instructions, and procedures for individuals interested in joining the research software project team, outlining the steps, requirements, expectations, and opportunities for becoming a contributor, collaborator, or team member.
When implemented:

- (M) Create a "Join Us" section on the project website with details on contributing.
- (S) Highlight contribution opportunities like coding, documentation, or outreach.
- (C) Specify prerequisites like skills or experience needed to join.
- (C) Outline the onboarding process, including accessing resources and finding mentorship.

Resources required:

- Time: Required for drafting and maintaining the documentation on joining the team.
- Utilize a website content management system (CMS), version control repository, or documentation platform to create and maintain the "Join the Team" or "Contribute" section, enabling collaborative editing, version control, and accessibility of joining instructions.
- Develop clear, concise, and engaging joining instructions, using plain language, bullet points, and visual aids to improve readability and comprehension, and incorporating feedback from team members and community stakeholders to enhance clarity and completeness of information.
- Provide links or instructions for joining project communication channels, such as mailing lists, forums, chat groups, or social media platforms, where individuals can connect with existing team members, ask questions, and express interest in joining the team.

Dependencies:
References: [77, 72]

Practice Code: 3.3.8
Practice Name: Set maximum response time for pull requests
Description: Establish a clear and reasonable maximum response time for reviewing and addressing pull requests submitted by contributors to the research software project, ensuring timely feedback, transparency, and efficiency in the contribution review process.
When implemented:

- (M) Set a maximum response time for pull request reviews, communicating the timeframe for feedback.
- (M) Document the response time policy in project guidelines, explaining expectations and consequences.
- (S) Assign reviewers responsible for timely reviews, rotating roles to distribute the workload.
- (C) Establish procedures for handling overdue reviews, like escalating to project leads or seeking community assistance to avoid delays.

Resources required:

- Time: Required for establishing and communicating the response time policy to project contributors and team members.
- Update project guidelines to include a maximum response time for pull requests, with clear instructions and consequences.
- Ensure reviewers have time and expertise to meet response time targets, adjusting schedules if needed.
- Use project tools to track pull request status, identify delays, and address bottlenecks.
- Encourage open communication between contributors and reviewers, providing updates and feedback for transparency and motivation.

Dependencies:
References: [54]

Practice Code: 3.3.9
Practice Name: Provide access to developer training and skill development
Description: Offer opportunities, resources, and support for researchers / research software engineers to enhance their skills, expand their knowledge, and further their professional development through training programs, courses, workshops, and learning resources relevant to research software development.
When implemented:

- (M) Provide access to a variety of training programs, courses, and workshops relevant to RS development, covering topics such as programming languages, version control systems, software engineering best practices, data management, cybersecurity, and project management.
- (M) Maintain a repository or library of learning resources, tutorials, guides, documentation, and online materials on research software development topics, accessible to researchers / research software engineers within the project community for self-paced learning and skill enhancement.
- (S) Organize skill development initiatives, such as coding challenges, hackathons, peer learning groups, or mentorship programs, to encourage collaboration, knowledge sharing, and hands-on practice among researchers / research software engineers, fostering a culture of continuous learning and improvement.
- (C) Provide financial support or stipends for researchers / research software engineers to attend external training courses, conferences, workshops, or certification programs related to RS development, enabling them to access specialized training opportunities and stay updated on emerging technologies and trends.

Resources required:

- Time: Required for identifying and planning relevant training resources and opportunities.
- Set aside money for researchers / research software engineers training, covering course fees, workshops, materials, and travel expenses.
- Invest in online learning platforms for researchers / research software engineers to access various educational materials.
- Hire external trainers or experts for specialized sessions, complementing internal training.
- Use an LMS to organize and track training activities, making administration more efficient.

Dependencies:
References: [41, 67]

Practice Code: 3.4.1
Practice Name: Select a license
Description: Choose an appropriate open-source license for the research software project that defines the terms and conditions under which the software is made available to users, researchers / research software engineers, and other stakeholders, ensuring legal clarity, compatibility, and openness of the project codebase.
When implemented:

- (M) Document and communicate the selected license in the project repository, README file, or license file, providing clear and explicit information about the licensing terms, obligations, permissions, and requirements for using, modifying, and distributing the project code.
- (M) Conduct a legal review of candidate licenses with legal counsel or experts to ensure compliance with applicable laws, regulations, and best practices, addressing any potential legal risks, liabilities, or ambiguities associated with the selected license.
- (S) Evaluate various open-source licenses based on their terms, permissions, restrictions, and obligations, considering factors such as license compatibility, project goals, community preferences, commercial use, distribution requirements, and intellectual property considerations.
- (C) Seek input and feedback from project stakeholders, contributors, and community members on the choice of license, soliciting their opinions, concerns, and preferences regarding licensing options and their potential impact on project governance, collaboration, and sustainability.

Resources required:

- Time: Required for research on different license options and their implications.
- Use online tools like OSI, SPDX, or Choose a License to find suitable licenses based on project needs.
- Seek advice from legal experts in open-source licensing to ensure chosen licenses comply with laws and project goals.
- Gather feedback from the project community through channels like mailing lists or forums to involve stakeholders in the decision-making process and gain support for the selected license.

Dependencies: 3.4.1 < 3.4.3
References: [12, 13]

Practice Code: 3.4.3
Practice Name: Get institutional support for license choice
Description: Engaging with institutional stakeholders to explain and justify the selected license choosing a license for the research software project, considering various factors such as project goals, community dynamics, collaboration models, and legal implications, to ensure that the selected license aligns with the project's values, objectives, and long-term sustainability.
When implemented:

- (M) Conduct a thorough assessment of available open-source licenses, examining their terms, conditions, restrictions, and implications about the project's needs, priorities, and constraints, and considering factors such as license compatibility, commercial use, attribution requirements, and patent grants.
- (M) Seek guidance and advice from legal counsel or experts specializing in open-source licensing and intellectual property law to analyze and interpret the legal implications and consequences of different license options, ensuring compliance with relevant regulations, mitigating legal risks, and protecting the project's interests.
- (S) Document the considerations, and decision-making process behind the choice of license in the project repository, README file, or license file, providing clear and accessible information about the selected license, its terms, permissions, and requirements, and communicating it effectively to users, contributors, and downstream researchers / research software engineers.
- (C) Engage with project stakeholders, contributors, and community members to gather input, feedback, and insights on license considerations, fostering open dialogue, transparency, and collaboration in the decision-making process, and addressing any concerns, preferences, or conflicts related to licensing choices.

Resources required:

- Time: Required for researching and evaluating different license options.
- Use OSI and FSF resources or legal guides to understand open-source licenses.
- Get advice from legal experts on open-source licensing for selecting licenses that align with project goals.
- Engage stakeholders through project channels for discussions and feedback on license choices, ensuring diverse input in decision-making.

Dependencies:
References: [78]

Practice Code: 3.4.8
Practice Name: Evaluate license policy regularly
Description: Establish a recurring review process to assess the effectiveness, relevance, and compliance of the project's chosen license(s), ensuring that they align with evolving project goals, community dynamics, legal requirements, and industry standards, and addressing any emerging issues, concerns, or opportunities related to licensing.
When implemented:

- (M) Set a schedule (like yearly or biennially) to review the project's license choices and document outcomes.
- (M) Periodically audit the project's licenses to ensure compliance with laws and address any issues promptly.
- (S) Establish benchmarks for evaluating licenses based on factors like compatibility, community adoption, legal compliance, and project dynamics.
- (S) Involve stakeholders, including contributors and legal advisors, for feedback on license evaluations, aiming for consensus.

Resources required:

- Time: Time for conducting regular reviews and assessments of the license policy.
- Create a checklist to assess the project's license policy, covering legal, technical, social, and strategic aspects for a thorough review.
- Keep records of license policy evaluations, documenting findings, recommendations, and decisions transparently for stakeholders.
- Seek advice from legal experts on open-source licensing to address licensing issues effectively.
- Engage stakeholders through project communication channels for inclusive discussions and informed decision-making on license policy evaluations.

Dependencies: * (all other practices) $<$ 3.4.8
References: [58, 13, 78]

Practice Code: 4.1.2
Practice Name: Provide a statement of purpose
Description: Crafting and presenting a clear, concise statement that outlines the fundamental goals and objectives of a research software project.
When implemented:

- (M) The statement of purpose communicates the project's objectives, goals, target audience, and intended impact, ensuring alignment and understanding among stakeholders.
- (M) The statement of purpose serves as a guiding document for project planning, execution, and decision-making, ensuring that project activities and outcomes are consistent with the stated objectives and goals.
- (S) The statement of purpose reflects input and feedback from key stakeholders, including researchers, researchers / research software engineers, funders, and end-users, ensuring that diverse perspectives and priorities are considered.
- (S) The statement of purpose is made readily available and accessible to all project stakeholders, including through project documentation, websites, and communication channels, facilitating transparency and engagement.

Resources required:

- Time: Required for drafting, reviewing, and finalizing the statement of purpose, including time for stakeholder consultations and revisions.
- Document editing and collaboration tools (e.g., Microsoft Word, Google Docs) for drafting and reviewing the statement of purpose, and communication platforms (e.g., email, video conferencing) for coordinating stakeholder input and feedback.
- Knowledge of project objectives, goals, and target audience, as well as effective communication and stakeholder engagement skills, are essential resources for developing a compelling and impactful statement of purpose.

Dependencies:
References: [72]

Practice Code: 4.1.3
Practice Name: Provide a simple how to use
Description: Offer a straightforward guide outlining the steps required to effectively use the research software project, aimed at users with varying levels of expertise.
When implemented:

- (M) The how-to-use guide is easy to understand and follows a logical sequence of steps, making it accessible to users with diverse backgrounds and skill levels.
- (M) The guide covers all essential functionalities and features of the software, providing sufficient detail and examples to assist users in performing common tasks and workflows.
- (S) The guide is refined based on feedback from actual users and usability testing sessions, ensuring that it addresses common user questions, concerns, and challenges effectively.
- (C) The guide is made available in multiple formats and channels, including online documentation, user manuals, video tutorials, and interactive demos, to accommodate different learning preferences and accessibility needs.

Resources required:

- Time: Required for initial drafting and development of the how-to-use guide, with additional time allocated for revisions, updates, and maintenance as the software evolves.
- Use of documentation authoring tools (e.g., Markdown, LaTeX, Adobe InDesign), screen capture software (e.g., Snagit, Camtasia), and collaboration platforms (e.g., Google Drive, Confluence) to create, edit, and publish the guide.
- Establishment of feedback channels (e.g., surveys, feedback forms, user forums) to gather input and suggestions from users regarding the clarity, usefulness, and effectiveness of the how-to-use guide.
- Knowledge of instructional design principles, technical writing techniques, and user-centered design methodologies are valuable resources for creating an effective and user-friendly how-to-use guide.

Dependencies: 4.1.3 < 4.1.6
References: [62]

Practice Code: 4.1.6
Practice Name: Provide online tutorials
Description: Develop and offer online tutorials or instructional materials that guide users through the usage and functionalities of the research software project, enhancing their understanding and proficiency.
When implemented:

- (M) The online tutorials cover a range of topics relevant to users, including basic functionality, advanced features, troubleshooting tips, and best practices, different user needs and skill levels.
- (M) The tutorials are accessible online through a user-friendly platform or website, featuring clear navigation, search functionality, and multimedia content (e.g., videos, screenshots, interactive demos) to facilitate learning and engagement.
- (S) The tutorials are regularly updated and maintained to reflect changes in the software, address user feedback, and incorporate new features or improvements, ensuring that they remain relevant and effective over time.
- (C) The tutorials incorporate interactive elements, quizzes, exercises, or hands-on demonstrations to actively engage users and reinforce learning objectives, encouraging participation and knowledge retention.

Resources required:

- Time: Plan, create, record/edit, and publish tutorials, with ongoing updates and maintenance.
- Use video editing software (like Adobe Premiere Pro, Camtasia), screen recording tools (e.g., OBS Studio, ScreenFlow), learning management systems (e.g., Moodle, Canvas), and content management systems (e.g., WordPress, Drupal) for hosting tutorials.
- Collaborate with researchers / research software engineers, researchers, and experts to ensure tutorial accuracy, relevance, and completeness, and to address user questions and feedback.
- Promote tutorials through social media, email newsletters, and community outreach to boost user participation.

Dependencies:
References: [65, 77, 27]

Practice Code: 4.2.2
Practice Name: Provide a read me file with project explanation
Description: Create a README file containing essential information about the research software project, including its purpose, features, installation instructions, usage guidelines, contribution guidelines, and licensing details, to help users and contributors understand and engage with the project effectively.
When implemented:

- (M) The README file should cover all necessary information about the project, including its objectives, key features, and how to use it.
- (M) The content should be presented in a clear and organized manner, with headings, bullet points, and code snippets used to improve readability and navigation.
- (S) The README file should be kept up-to-date with any changes or additions to the project, ensuring that users have access to the latest information.
- (C) The README file should invite feedback from users and contributors, providing contact information or links to relevant communication channels for reporting issues or making suggestions.

Resources required:

- Time: Time required to create the README file, plus additional time for updates as the project evolves.
- Text editor or markdown editor for writing the README file (e.g., Visual Studio Code, Atom, Typora).

Dependencies: 4.2.2 < 4.2.8 and 4.2.2 < * (all other practices)
References: [65, 79]

Practice Code: 4.2.3
Practice Name: Provide a how-to guide
Description: Develop a comprehensive step-by-step guide that walks users through specific tasks or processes related to the research software project, providing detailed instructions, tips, and examples to help users achieve their objectives effectively.
When implemented:

- (M) The how-to guide should be clear, concise, and comprehensive, covering all necessary steps and details required to complete the task successfully.
- (M) Organize the guide into logical sections or steps, with each step numbered or labeled, and use headings, subheadings, and bullet points to improve readability and comprehension.
- (S) Ensure that the content of the guide is relevant to the target audience and reflects current best practices and procedures, verifying the accuracy of information and instructions provided.
- (S) Collect feedback from users or stakeholders on the usability and effectiveness of the how-to guide, making revisions or updates based on user suggestions or identified areas for improvement.

Resources required:

- Time: Depending on the complexity of the task or process being documented, creating a how-to guide may require several hours to days of effort to research, write, review, and finalize the content.
- Text editor or documentation tool for drafting and formatting the guide (e.g., Microsoft Word, Google Docs, Markdown editors). Additionally, collaboration and version control tools may be used for team collaboration and document management (e.g., Git, GitHub, GitLab).

Dependencies:
References: [62]

Practice Code: 4.2.4
Practice Name: Provide a common example usage
Description: Present a typical scenario or use case demonstrating how users can effectively utilize the research software project to accomplish a specific task or achieve a common objective relevant to their needs or domain.
When implemented:

- (M) The example usage is easy to understand, with each step or action explained straightforwardly to facilitate user comprehension and replication.
- (S) The example usage provided should be relevant and representative of common tasks or workflows encountered by users within the target domain or context.
- (S) The example usage should reflect realistic scenarios and practical applications of the research software project, demonstrating its utility and effectiveness in addressing real-world challenges or requirements.
- (C) Provide multiple example usages showcasing different aspects, features, or functionalities of the research software project.

Resources required:

- Time: Developing common example usage scenarios may require several hours to days of effort to research, define, and document representative scenarios accurately.
- Knowledge of the target domain or context is essential for identifying relevant and practical example usage scenarios that resonate with users and reflect real-world application scenarios.
- Input and feedback from users or stakeholders can help validate the relevance and effectiveness of example usage scenarios, ensuring that they address common user needs and requirements effectively.

Dependencies:
References: [65, 62]

Practice Code: 4.2.6
Practice Name: Provide a documentation as repository/ documentation wiki
Description: Establish a central repository or wiki system to host comprehensive documentation for the research software project, serving as a centralized source of information, guidance, and resources for users, researchers / research software engineers, and contributors.
When implemented:

- (M) The documentation repository or wiki should cover all relevant aspects of the research software project, including its purpose, features, architecture, installation instructions, usage guidelines, API references, troubleshooting tips, and contribution guidelines.
- (M) The documentation should be organized into logical sections or categories, with clear navigation paths, headings, and subheadings to facilitate easy access and retrieval of information by users.
- (S) Ensure that the documentation content is consistent in style, tone, and formatting and that it accurately reflects the current state of the project, with regular updates and revisions made as needed to keep the information up-to-date and relevant.
- (S) The documentation should be easily accessible to all stakeholders, with user-friendly interfaces, search functionality, and navigation aids provided to help users quickly locate and retrieve the information they need.

Resources required:

- Time: Allocate hours to days for researching, selecting, and implementing documentation technologies, depending on project complexity.
- Use popular documentation tools like Markdown, HTML/CSS, Git, GitHub Pages, or static site generators (e.g., Jekyll, Hugo) for creating and managing project documentation, as they are simple, versatile, and compatible with existing workflows.
- Offer training and support to team members and stakeholders on using chosen documentation tools effectively, enabling efficient contribution and maintenance.

Dependencies:
References: [32, 61]

Practice Code: 4.2.8
Practice Name: Provide API documentation
Description: Create comprehensive documentation for the application programming interface (API) of the research software project, detailing its endpoints, methods, parameters, responses, and usage guidelines to facilitate integration, development, and interaction with the API by researchers / research software engineers and third-party applications.
When implemented:

- (M) The documentation should cover all available API endpoints, methods, and functionalities supported by the research software project, providing detailed descriptions, usage examples, and response schemas for each.
- (M) Each API endpoint and method should be accompanied by clear and concise descriptions explaining its purpose, expected inputs, possible outputs, error conditions, and any additional context or considerations relevant to API usage.
- (S) Include usage examples and code snippets demonstrating how to interact with each API endpoint or method using popular programming languages or frameworks, helping researchers / research software engineers understand and implement API integrations effectively.
- (S) Specify the format and structure of API responses, including data schemas, status codes, and error messages, to guide researchers / research software engineers in handling and processing API responses correctly and handling errors gracefully.

Resources required:

- Time: Developing API documentation may require significant time and effort, ranging from several hours to days or weeks, depending on the complexity and scope of the API, to research, document, review, and finalize the API documentation content.
- Use documentation tools or platforms such as Swagger/OpenAPI, Postman, Slate, or ReDoc to create and manage API documentation, providing features for generating interactive API documentation, code samples, and documentation versioning.
- researchers / research software engineers and documentation team members may require expertise in API design principles, RESTful architecture, HTTP protocols, and JSON/XML data formats to effectively document API endpoints, methods, and responses.

Dependencies:
References: [14]

Practice Code: 4.3.2
Practice Name: Use common non-exotic or established technology
Description: Employ widely adopted, non-exotic, or established technologies and tools for developing, managing, and deploying the research software project, ensuring compatibility, stability, and accessibility for users across diverse environments and systems.
When implemented:

- (M) Ensure that the documentation is compatible with common web browsers, operating systems, and devices, and accessible to users with diverse needs and preferences, including those using assistive technologies or older hardware/software configurations.
- (M) Use documentation tools or platforms that offer robust features for content creation, editing, versioning, and collaboration, enabling efficient and effective maintenance and updates to the documentation over time.
- (S) The chosen documentation technologies should be widely adopted and familiar to users, researchers / research software engineers, and stakeholders within the project's target audience, reducing barriers to entry and facilitating the adoption and usage of the documentation.
- (S) Choose documentation technologies that support scalability and extensibility, allowing for the addition of new content, features, or functionalities as the project evolves and grows in complexity.

Resources required:

- Time: Allocate hours to days for researching, selecting, and implementing documentation technologies, depending on project complexity.
- Use popular documentation tools like Markdown, HTML/CSS, Git, GitHub Pages, or static site generators (e.g., Jekyll, Hugo) for creating and managing project documentation, as they are simple, versatile, and compatible with existing workflows.
- Offer training and support to team members and stakeholders on using chosen documentation tools effectively, enabling efficient contribution and maintenance.

Dependencies: 4.3.2 < * (all other practices)
References: [49, 45]

Practice Code: 4.3.4
Practice Name: Facilitate integration into scientific workflow
Description: Streamline the process of integrating the research software project into common scientific workflows, data processing pipelines, and analysis frameworks used within the target domain, enhancing interoperability, usability, and adoption by scientific researchers and data analysts.
When implemented:

- (M) Evaluate the compatibility of the research software project with existing scientific workflow tools, platforms, and environments commonly used by researchers and data analysts.
- (M) Provide comprehensive documentation and support resources to assist users in integrating the research software project into their scientific workflows, including tutorials, guides, and troubleshooting tips.
- (S) Customize integration solutions to align with the specific requirements and workflows of scientific researchers, ensuring seamless interoperability and data exchange between the research software project and other tools or systems.
- (S) Gather feedback from scientific researchers and data analysts on the effectiveness and usability of the integrated solution, making refinements and adjustments as needed to optimize integration and address user needs.

Resources required:

- Time: Allocate time for compatibility assessment, customization of integration solutions, gathering user feedback, and documentation development, with the duration depending on the complexity and scope of integration requirements.
- Utilize integration frameworks, libraries, APIs, and middleware commonly used in scientific workflow environments to facilitate integration efforts and enhance interoperability.
- Provide training and support to integration specialists and researchers / research software engineers on scientific workflow tools, data exchange standards, and integration best practices to ensure successful integration outcomes.

Dependencies:
References: [79, 43, 20]

Practice Code: 4.3.9
Practice Name: Evaluate technology relevance regularly
Description: Continuously assess the relevance and suitability of the technologies and tools used in the research software project, considering factors such as emerging trends, evolving user requirements, technological advancements, and changes in the project's scope or objectives.
When implemented:

- (M) Collect feedback from users, stakeholders, and contributors regarding their technology preferences, challenges, and requirements, and incorporate this input into the evaluation and selection of technologies for the project.
- (M) Ensure that selected technologies align with the current and future goals, objectives, and requirements of the research software project, supporting scalability, maintainability, performance, and user satisfaction.
- (S) Conduct periodic reviews and assessments of the technologies and tools employed in the research software project to identify outdated, deprecated, or obsolete components and evaluate their impact on project goals and objectives.
- (S) Stay informed about emerging technologies, best practices, and industry trends relevant to the project's domain or target audience, leveraging resources such as industry reports, conferences, forums, and professional networks.

Resources required:

- Time: Allocate dedicated time for technology assessment activities, including research, analysis, and documentation of findings, as well as coordination with stakeholders and team members.
- Utilize the expertise of team members or external consultants with knowledge of emerging technologies, industry trends, and best practices relevant to the project's domain or target audience.
- Leverage tools, resources, and frameworks for technology assessment and evaluation, such as technology radar charts, maturity models, decision matrices, and risk assessment frameworks, to support informed decision-making and planning.

Dependencies: * (all other practices) $<$ 4.3.9
References: [37, 51]

Practice Code: 4.4.2
Practice Name: Provide instructions on how to put into research workflow
Description: Create clear and detailed instructions for integrating reproducibility practices into the research workflow, ensuring that team members can effectively implement reproducible research practices.
When implemented:

- (M) Detailed instructions are provided that outline step-by-step procedures for integrating reproducibility practices into the research workflow.
- (M) Team members successfully integrate reproducibility practices into the research workflow following the provided instructions, leading to consistent and reproducible research outcomes.
- (S) Instructions are easily accessible to all team members involved in the research project, ensuring that they can reference and follow the instructions as needed.
- (C) Feedback is collected from team members regarding the clarity and effectiveness of the instructions, and adjustments are made as necessary to improve usability and understanding.

Resources required:

- Time: Time is required for researching, drafting, and refining the instructions, depending on the complexity of the research workflow and the level of detail required.
- Word processing software (e.g., Microsoft Word, Google Docs) for creating documentation, collaboration platforms (e.g., Google Drive, SharePoint) for sharing and accessing instructions, and communication tools (e.g., email, Slack) for collecting feedback from team members.

Dependencies: 4.4.2 < 4.4.5
References: [43, 79]

Practice Code: 4.4.4
Practice Name: Provide instructions on how to make part of a replication package
Description: Offer detailed guidelines and procedures for researchers to include the research software project as part of a replication package, ensuring reproducibility and transparency of research findings.
When implemented:

- (M) The provided instructions cover all necessary steps and procedures for packaging the RS along with relevant data, scripts, dependencies, and documentation, following best practices for reproducible research.
- (S) The instructions are presented in a clear and accessible format, such as a README file, user manual, or online documentation, with step-by-step guidance, examples, and explanations to assist researchers of varying skill levels.
- (C) The instructions accommodate different research contexts, software environments, and replication requirements, allowing researchers to adapt and customize the replication package according to their specific needs and constraints.
- (C) The instructions include guidance on verifying the correctness and completeness of the replication package, as well as validation procedures to ensure that it can be successfully reproduced and used by others to replicate research findings.

Resources required:

- Time: Time required for researching, writing, reviewing, and finalizing the instructions, with additional time allocated for updates and revisions as needed.
- Use of documentation tools (e.g., Markdown, LaTeX, Microsoft Word), version control systems (e.g., Git), and collaboration platforms (e.g., GitHub, Google Docs) for drafting, reviewing, and publishing the instructions.
- Collaboration with domain experts, researchers, and stakeholders to ensure that the instructions accurately reflect the requirements and practices of the research field and adhere to reproducibility standards and guidelines.
- Collecting feedback from researchers and users to validate the effectiveness, usability, and completeness of the instructions, and to identify areas for improvement or clarification.

Dependencies: 4.4.4 < 4.4.7
References: [79, 43]

Practice Code: 4.4.5
Practice Name: Make part of standardized workflows
Description: Ensure that reproducibility practices are incorporated into standardized workflows, ensuring consistency and adherence to best practices across the research project.
When implemented:

- (M) Reproducibility practices are seamlessly integrated into standardized workflows used across the research project, ensuring that they are followed consistently by all team members.
- (M) Reproducibility practices align with existing standardized protocols and procedures used within the research project, minimizing disruptions and ensuring smooth integration.
- (S) Team members receive training and support to understand and implement reproducibility practices as part of standardized workflows, promoting adoption and compliance.
- (S) Regular monitoring is in place to ensure that reproducibility practices are being followed as part of standardized workflows, with mechanisms in place to address any deviations or issues.

Resources required:

- Time: Time is required for assessing existing workflows, identifying opportunities for integration, and developing training materials for team members.
- Collaboration platforms (e.g., project management software, version control systems) for documenting standardized workflows and tracking compliance with reproducibility practices, communication tools (e.g., email, Slack) for disseminating training materials and providing support.

Dependencies:
References: [43, 79]

Practice Code: 4.4.7
Practice Name: Make part of a replication package
Description: Ensure that reproducibility practices are incorporated into a comprehensive replication package accompanying the research project, facilitating the replication and verification of research findings by others.
When implemented:

- The replication package includes all necessary materials, such as data, code, scripts, documentation, and instructions, to reproduce the research findings.
- (M) The replication package adheres to established standards and guidelines for reproducible research, ensuring transparency, completeness, and reliability.
- (S) The replication package is well-organized and easily accessible, allowing other researchers to locate and utilize the materials effectively.
- (S) The reproducibility of research findings is successfully validated by others using the materials provided in the replication package.

Resources required:

- Time: Time is required for preparing and organizing materials, documenting procedures, and validating reproducibility, depending on the complexity of the research project.
- Version control systems (e.g., Git), data repositories (e.g., Zenodo, Figshare), and collaboration platforms (e.g., Google Drive, GitHub) for storing and sharing replication package materials, documentation tools (e.g., Markdown, LaTeX) for creating documentation, communication tools (e.g., email, Slack) for coordinating efforts and disseminating the replication package.

Dependencies:
References: [74, 1]

Practice Code: 4.5.5
Practice Name: Develop generic educational materials
Description: Create educational materials designed to support users and contributors of the research software project.
When implemented:

- (M) During the early stages, to provide foundational resources for early adopters and internal team members.
- (M) Continuously, as part of routine updates, to keep educational content current with software developments and user feedback.
- (S) Before major releases, ensure comprehensive support materials are available for new features and changes.
- (S) When aiming to grow the user base and contributor community, offering clear and helpful educational materials lowers the barrier to entry.

Resources required:

- Time: Time is required for researching, writing, designing, and reviewing educational materials, depending on the depth and breadth of coverage.
- Word processing software (e.g., Microsoft Word, Google Docs) for creating written content, presentation software (e.g., PowerPoint, Google Slides) for designing slides, graphics software (e.g., Adobe Illustrator, Canva) for creating visual elements, collaboration platforms (e.g., Google Drive, SharePoint) for sharing and reviewing materials.

Dependencies: $4.5.5 < 4.5.10$
References: [56, 83]

Practice Code: 4.5.9
Practice Name: Organize training events in person
Description: Conduct in-person training events, workshops, or seminars providing hands-on learning about the research software project and fostering collaboration among researchers.
When implemented:

- (M) Training events attract active participation and attendance from researchers, demonstrating interest and commitment to learning about the research software project.
- (S) Training sessions incorporate interactive activities, case studies, and hands-on exercises to facilitate learning and application of the research software.
- (C) Training events facilitate networking and collaboration among participants, fostering knowledge sharing, peer support, and community building around research software.

Resources required:

- Time: Varied depending on the scale and duration of training events, ranging from several hours for short workshops to multiple days for intensive seminars or conferences.
- Securing suitable venues with appropriate facilities such as meeting rooms, audiovisual equipment, and internet connectivity for hosting training events.
- Recruiting qualified trainers and speakers with expertise in the particular research domain to deliver engaging and informative sessions.
- Developing promotional materials (e.g., flyers, posters, online advertisements) to market training events and attract participants.
- Providing necessary materials, handouts, and supplies for training sessions, including laptops, notebooks, pens, and refreshments.

Dependencies:
References: [41, 31]

Practice Code: 4.5.10
Practice Name: Make project part of an educational program
Description: Incorporate the research software project into an educational program or curriculum, providing opportunities for students to engage with and contribute to the project while gaining valuable skills and knowledge.
When implemented:

- (M) The research software project is seamlessly integrated into an educational program or course curriculum, aligning with learning objectives and academic standards.
- (S) Students achieve learning outcomes related to research software project skills, critical thinking, problem-solving, collaboration, and communication through their involvement in the project.
- (C) Students actively engage with the research software project through coursework, projects, internships, or research assistantships, contributing to project activities and outcomes.

Resources required:

- Time: Required for coordination and planning with educational institutions or programs.
- Development of educational materials, resources, and assignments related to the research software project to support student learning and engagement.
- Assessment tools and processes to evaluate student performance, track learning outcomes, and measure the impact of the project on student learning and development.

Dependencies:
References: [31, 56]

Practice Code: 4.6.6
Practice Name: Provide with standard deployment tools (docker images, etc.)
Description: Equip the research project with standardized deployment tools, such as Docker images, to streamline the deployment process and ensure consistency across different computing environments.
When implemented:

- (M) The deployment tools are compatible with common computing environments and platforms, ensuring that the research software project can be deployed consistently and reliably across different systems.
- (S) Researchers and users can deploy the research software project using the provided tools without encountering significant issues or dependencies, resulting in a functional and operational system.
- (C) Deployment tools are user-friendly and well-documented, allowing researchers and practitioners to quickly deploy the research software project without requiring extensive technical expertise or troubleshooting.
- (C) Standard deployment tools, such as Docker images or containerization scripts, are provided alongside the research software project's codebase, enabling easy setup and deployment.

Resources required:

- Time: Time is required for developing, testing, and documenting deployment tools, depending on the complexity of the project and the deployment requirements.
- Docker or other containerization platforms for creating standardized deployment images, scripting languages (e.g., Bash, Python) for automation and configuration management, and version control systems (e.g., Git) for managing deployment scripts and configurations.
- Creation of clear and comprehensive documentation for using the deployment tools, including installation instructions, configuration options, and troubleshooting guides.
- Provision of training sessions or workshops to familiarize users with deployment tools and best practices for deploying the project, as well as ongoing support and assistance to address any deployment-related issues or questions.

Dependencies: 4.6.6 < 4.6.8 and 4.6.6 < * (all other practices)
References: [79, 79]

Practice Code: 4.6.8
Practice Name: Enable deployment on a wide range of technology
Description: Ensure that the research software project can be deployed on various technology stacks, platforms, and environments to maximize accessibility and adoption by different user groups.
When implemented:

- (M) Users can deploy the project on various technology environments, including different operating systems, cloud providers, and hardware architectures, with minimal effort and without encountering major compatibility issues.
- (S) The research software project is designed and configured to be compatible with a wide range of technology stacks, including different operating systems, programming languages, and infrastructure setups.
- (C) Deployment configurations and dependencies are modular and configurable, allowing users to customize deployment settings based on their specific requirements and preferences.
- (C) The research software project can be deployed seamlessly on different computing platforms, including desktops, servers, cloud infrastructure, and edge devices, without requiring significant modifications or adaptations.

Resources required:

- Time: Time is required for researching, testing, and optimizing deployment setups for different tech environments, depending on complexity.
- Utilize cross-platform development tools, containerization (like Docker), infrastructure-as-code frameworks (e.g., Terraform), and cloud services (AWS, Azure) for diverse tech stack deployments.
- Set up testing infrastructure and automation to validate configurations for compatibility with various environments, OS, and hardware.
- Provide detailed guides and support resources for users deploying the research software project on different platforms, offering ongoing assistance for deployment challenges.

Dependencies:
References: [56, 6]

Practice Code: 4.6.9
Practice Name: Provide coordination mechanisms for workflow distribution over different machines
Description: Implement mechanisms to distribute and coordinate the execution of workflow tasks across multiple machines or computing nodes, optimizing performance and resource utilization in distributed computing environments.
When implemented:

- (M) The system is resilient to failures and interruptions, with mechanisms for task recovery, fault detection, and automatic rerouting to alternative resources in case of machine failures or network disruptions.
- (S) Mechanisms are in place to coordinate and synchronize the execution of distributed tasks, ensuring correct sequencing, data consistency, and dependency management across the workflow.
- (C) The system effectively distributes workflow tasks across different machines or computing nodes based on workload, availability, and resource constraints, maximizing parallelism and minimizing execution time.
- (C) The coordination mechanisms can scale dynamically to accommodate increasing workload and resource demands, maintaining high performance and efficiency in distributed computing environments of varying sizes and configurations.

Resources required:

- Time: Time is required for designing, implementing, and testing coordination mechanisms for workflow distribution, adjusted based on the complexity and size of the computing setup.
- Utilize distributed computing frameworks like Apache Hadoop, message-passing libraries such as MPI, or workflow management systems like Apache Airflow to manage task distribution across machines.
- Set up and configure computing resources like servers, clusters, or cloud instances to support distributed workflow execution.
- Employ monitoring tools to track workflow progress, resource usage, and system performance, aiding in issue detection and resolution related to task coordination.

Dependencies:
References: [79, 42, 48]

Practice Code: 4.6.10
Practice Name: Generate Software Bill of Materials (SBOM)
Description: Create and maintain a Software Bill of Materials (SBOM) that documents all the components and dependencies used in the research software project, providing transparency and visibility into its composition.
When implemented:

- (M) The SBOM includes a comprehensive list of all software components and dependencies utilized in the research project, including libraries, frameworks, modules, and third-party tools.
- (M) Each research software component listed in the SBOM is accompanied by information about its version number, release date, and any relevant metadata, facilitating version tracking and management.
- (S) The SBOM documents the dependency relationships between different research software components, indicating dependencies, sub-dependencies, and potential security vulnerabilities or licensing issues.
- (C) The SBOM is regularly updated and maintained to reflect changes in the research software ecosystem, including additions, removals, updates, and patches to research software components and dependencies.

Resources required:

- Time: Time is required to compile and document the SBOM, with ongoing updates depending on project size and complexity.
- Use SBOM generation or SCA tools to automate SBOM creation, and version control systems like Git to track changes.
- Provide materials to educate contributors and stakeholders on SBOM importance and usage.
- Integrate SBOM processes into development workflows for seamless incorporation into research software development and release processes.

Dependencies:
References: [86, 53]

2 Disclaimer

The descriptions of the practices in this dataset have been partially generated using ChatGPT and subsequently rechecked for accuracy and completeness.

References

[1] C. Titus Brown. Replication, reproduction, and remixing in research software. http://ivory.idyll.org/blog/research-software-reuse.html. Published: Jan. 18, 2013, Accessed: April 24, 2024.

[2] actiTime. The importance of feedback in project management. https://medium.com/actiresults/the-importance-of-feedback-in-project-management-a1b06978e18b. Published: March 29, 2018, Accessed: January 22, 2024.

[3] M. Alenezi. Software architecture quality measurement stability and understandability. *International Journal of Advanced Computer Science and Applications*, 7(7), 2016.

[4] B. Álvarez-Bornstein and M. Montesi. Funding acknowledgements in scientific publications: A literature review. *Research evaluation*, 29(4):469–488, 2020.

[5] L. An, F. Khomh, S. Mcintosh, and M. Castelluccio. Why did this reviewed code crash? an empirical study of mozilla firefox. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 396–405, 2018.

[6] J.-P. Arcangeli, R. Boujbel, and S. Leriche. Automatic deployment of distributed software systems: Definitions and state of the art. *Journal of Systems and Software*, 103:198–218, 2015.

[7] H. Artaza, N. C. Hong, M. Corpas, A. Corpuz, R. Hooft, R. C. Jiménez, B. Leskošek, B. G. Olivier, J. Stourac, R. S. Vařeková, et al. Top 10 metrics for life science software good practices. *F1000Research*, 5, 2016.

[8] B. Aston. 10 best requirements management tools reviewed for 2024. Published: November 23, 2023, Accessed: January 22, 2024.

[9] C. K. Atkin and V. Freimuth. Guidelines for formative evaluation research in campaign design. *Public communication campaigns*, 4:53–68, 2013.

[10] F. B. Aydemir and F. Dalpiaz. A roadmap for ethics-aware software engineering. In *Proceedings of the International Workshop on Software Fairness*, pages 15–21, 2018.

[11] S. Balali, I. Steinmacher, U. Annamalai, A. Sarma, and M. A. Gerosa. Newcomers' barriers... is that all? an analysis of mentors' and newcomers' barriers in oss projects. *Computer Supported Cooperative Work (CSCW)*, 27:679–714, 2018.

[12] M. Barker, C. H. NP, D. Katz, A. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L. Castro, M. Gruenpeter, P. Martinez, et al. Introducing the fair principles for research software. *Scientific Data*, 9(1):622–622, 2022.

[13] F. Belliard, A. M. Maineri, E. Plomp, A. F. Ramos Padilla, J. Sun, and M. Zare Jeddi. Ten simple rules for starting fair discussions in your community. *PLOS Computational Biology*, 19(12):e1011668, 2023.

[14] C. Bird, T. Menzies, and T. Zimmermann. *The art and science of analyzing software data*. Elsevier, 2015.

[15] C. Boogerd and L. Moonen. Assessing the value of coding standards: An empirical study. In *2008 IEEE International conference on software maintenance*, pages 277–286. IEEE, 2008.

[16] G. BOULEZ. Technical documentation: Cost or investment? https://www.fluidtopics.com/blog/best-practices/technical-documentation-cost-or-investment/. Published: Sep 20, 2021, Accessed:April 24, 2024.

[17] Cai, and Robert. 12 Benefits of Newsletters: Why they are essential to your email marketing strategy! Published: Oct. 2, 2023, Accessed: March 28, 2024.

[18] M.-H. Chen, M. R. Lyu, and W. E. Wong. Effect of code coverage on software reliability measurement. *IEEE Transactions on reliability*, 50(2):165–170, 2001.

[19] R. Clarke. Privacy impact assessment: Its origins and development. *Computer law & security review*, 25(2):123–135, 2009.

[20] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345–1350, 2008.

[21] M. Dayarathna, Y. Wen, and R. Fan. Data center energy consumption modeling: A survey. *IEEE Communications surveys & tutorials*, 18(1):732–794, 2015.

[22] Deekshitha, S. Farshidi, J. Maassen, R. Bakhshi, R. Van Nieuwpoort, and S. Jansen. Fairseco: An extensible framework for impact measurement of research software. In *2023 IEEE 19th International Conference on e-Science (e-Science)*, pages 1–10, 2023.

[23] P. Dhoolia, P. Chugh, P. Costa, N. Gantayat, M. Gupta, N. Kambhatla, R. Kumar, S. Mani, P. Mitra, C. Rogerson, and M. Saxena. A cognitive system for business and technical support: A case study. *IBM Journal of Research and Development*, 61(1):7:74–7:85, 2017.

[24] R. M. dos Santos and M. A. Gerosa. Impacts of coding practices on readability. In *Proceedings of the 26th Conference on Program Comprehension*, pages 277–285, 2018.

[25] A. Dziuba. How to Run a Successful Software Engineer Performance Review [Template Included]. https://relevant.software/blog/run-successful-software-engineer-performance-review/. Published: August 30, 2023, Accessed: March 28, 2024.

[26] D. D'Agostino, I. Merelli, M. Aldinucci, and D. Cesini. Hardware and software solutions for energy-efficient computing in scientific programming. *Scientific Programming*, 2021:1–9, 2021.

[27] S. Epskamp. Reproducibility and replicability in a fast-paced methodological world. *Advances in Methods and Practices in Psychological Science*, 2(2):145–155, 2019.

[28] J. R. Erenkrantz. Release management within open source projects. In *Proc. 3rd. workshop on open source software engineering*, volume 10, 2003.

[29] R. E. Fairley. *Software Project Management*, page 1634–1636. John Wiley and Sons Ltd., GBR, 2003.

[30] D. Garijo, H. Ménager, L. Hwang, A. Trisovic, M. Hucka, T. Morrell, and A. Allen. Nine best practices for research software registries and repositories. *PeerJ Computer Science*, 8:e1023, 2022.

[31] GGIR. Ggir training. https://www.accelting.com/ggir-training/. Accessed: April 24, 2024.

[32] GitHub. Documenting your project with wikis. https://docs.github.com/en/communities/documenting-your-project-with-wikis. Accessed: Feb 19, 2024.

[33] J. Gogoll, N. Zuber, S. Kacianka, et al. Ethics in the software development process: from codes of conduct to ethical deliberation. *Philos. Technol.*, 34:1085–1108, 2021.

[34] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen. Communication in open source software development mailing lists. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 277–286, 2013.

[35] W. Hazeleger, T. Bakker, and R. van Nieuwpoort. escience development and experiences in the netherlands. *Informatik-Spektrum*, 41:405–413, 2018.

[36] IBM. End-of-life Software Guidance. https://www.ibm.com/docs/en/randori?topic=guidance-end-life-eol. Updated: Jan. 18, 2024, Accessed: April 24, 2024.

[37] M. Ivarsson and T. Gorschek. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, 16:365–395, 2011.

[38] S. Jansen. Measuring the health of open source software ecosystems: Beyond the scope of project health. *Information and Software Technology*, 56(11):1508–1519, 2014.

[39] S. Jansen, S. Farshidi, G. Gousios, J. Visser, T. van der Storm, and M. Bruntink. Searchseco: A worldwide index of the open source software ecosystem. In *Proceedings of the 19th Belgium-Netherlands Software Evolution Workshop (BENEVOL 2020)*, volume 2912 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020.

[40] R. C. Jiménez, M. Kuzak, M. Alhamdoosh, M. Barker, B. Batut, M. Borg, S. Capella-Gutierrez, N. C. Hong, M. Cook, M. Corpas, et al. Four simple recommendations to encourage best practices in research software. *F1000Research*, 6, 2017.

[41] E. C. Johnson, T. T. Nguyen, B. K. Dichter, F. Zappulla, M. Kosma, K. Gunalan, Y. O. Halchenko, S. Q. Neufeld, M. Schirner, P. Ritter, et al. A maturity model for operations in neuroscience research. *arXiv preprint arXiv:2401.00077*, 2023.

[42] G. Kappel, S. Rausch-Schott, and W. Retschitzegger. Coordination in workflow management systems—a rule-based approach. In *Coordination Technology for Collaborative Applications: Organizations, Processes, and Agents 2*, pages 99–119. Springer, 1998.

[43] E. Karoune and E. Plomp. Removing barriers to reproducible research in archaeology. *Zenodo, ver. 5, peer reviewed and recommended by Peer Community in Archaeology*, 2022.

[44] D. S. Katz, L. C. McInnes, D. E. Bernholdt, A. C. Mayes, N. P. C. Hong, J. Duckles, S. Gesing, M. A. Heroux, S. Hettrick, R. C. Jimenez, M. Pierce, B. Weaver, and N. Wilkins-Diehr. Community organizations: Changing the culture in which research software is developed and sustained. *Computing in Science & Engineering*, 21(2):8–24, 2019.

[45] B. Kleinke. Challenges and lessons learned introducing an evolving open source technology into an established legacy ada and c++ program. *ACM SIGAda Ada Letters*, 40(2):70–72, 2021.

[46] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey. Investigating code review quality: Do people and participation matter? In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 111–120, 2015.

[47] T. Kravchenko, T. Bogdanova, and T. Shevgunov. Ranking requirements using moscow methodology in practice. In *Computer Science On-line Conference*, pages 188–199. Springer, 2022.

[48] A. Kumar, W. M. Van Der Aalst, and E. M. Verbeek. Dynamic work distribution in workflow management systems: How to balance quality and performance. *Journal of Management Information Systems*, 18(3):157–193, 2002.

[49] Kwixand. The costs and challenges of maintaining your legacy erp software. https://www.kwixand.com/post/the-costs-of-maintaining-legacy-erp-software. Published November 28, 2022, Accessed: Feb 19, 2024.

[50] M. Leach, L. Mehta, and P. Prabhakaran. Sustainable development: A gendered pathways approach. In *Gender equality and sustainable development*, pages 1–33. Routledge, 2015.

[51] A. M. León, Ó. F. C. Domínguez, and F. A. Vargas. Evaluating, selecting and relevance software tools in technology monitoring. *Ingeniería e Investigación*, 26(1):101–110, 2006.

[52] P. Levin. Project controls online: Website, e-newsletter examine growing trend. *Cost Engineering*, 47(10):9, 2005.

[53] J. Linåker, G. Robles, D. Bryant, and S. Muto. Open source software in the public sector: 25 years and still in its infancy. *IEEE Software*, 40(4):39–44, 2023.

[54] J. Liu, J. Li, and L. He. A comparative study of the effects of pull request on github projects. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 313–322. IEEE, 2016.

[55] M. R. Lunn, M. Lubensky, C. Hunt, A. Flentje, M. R. Capriotti, C. Sooksaman, T. Harnett, D. Currie, C. Neal, and J. Obedin-Maliver. A digital health research platform for community engagement, recruitment, and retention of sexual and gender minority adults in a national longitudinal cohort study——the pride study. *Journal of the American Medical Informatics Association*, 26(8-9):737–748, 2019.

[56] C. Martinez-Ortiz, D. S. Katz, A. Lamprecht, M. Barker, A. Loewe, A. Fouilloux, J. Wyngaard, D. Garijo, J. Moldon, L. Castro, et al. Fair4rs: Adoption support. Technical report, Zenodo, 2022.

[57] C. Martinez-Ortiz, P. Martinez Lavanchy, L. Sesink, B. G. Olivier, J. Meakin, M. de Jong, and M. Cruz. Practical guide to software management plans, Jan. 2023.

[58] K. E. Maull and M. Mayernik. Expanding impact metrics contexts with software citation. *NCAR Technical Notes NCAR/TN-567+ PROC*, page 23.

[59] M. Medeiros, U. Kulesza, R. Bonifacio, E. Adachi, and R. Coelho. Improving bug localization by mining crash reports: An industrial study. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 766–775. IEEE, 2020.

[60] C. Meeßen. Increasing the visibility of Research Software: The Helmholtz Research Software Directory, May 2023.

[61] S. Nesbitt. 3 tips for effectively using wikis for documentation. Published:January 2, 2017, Accessed: Feb 19, 2024.

[62] U. of Bristol. How to publish research software. https://www.bristol.ac.uk/acrc/research-software-engineering/software-howtos/how-to-publish-software/. Accessed: January 22, 2024.

[63] S. Osafo-Addo. Security in Code Reviews: Ensuring Secure and Robust Software Development. Published: August 07, 2023, Accessed: March 28, 2024.

[64] S. O'Mahony. The governance of open source initiatives: what does it mean to be community managed? *Journal of Management & Governance*, 11:139–150, 2007.

[65] K. Quach. Mapping research software landscapes through exploratory studies of github data. Master's thesis, 2022.

[66] A. Ramirez, A. Aiello, and S. J. Lincke. A survey and comparison of secure software development standards. In *2020 13th CMI Conference on Cybersecurity and Privacy (CMI)-Digital Transformation-Potentials and Challenges (51275)*, pages 1–6. IEEE, 2020.

[67] M. Régnier and M. Dacos. *Report of the G7 Open Science-Research on Research Sub-Working Group*. PhD thesis, Comité pour la science ouverte, 2023.

[68] K. R. Riisom, M. S. Hubel, H. M. Alradhi, N. B. Nielsen, K. Kuusinen, and R. Jabangwe. Software security in agile software development: A literature review of challenges and solutions. In *Proceedings of the 19th International Conference on Agile Software Development: Companion*, pages 1–5, 2018.

[69] Rob Healy, Brian Fitzgerald,Kieran Conboy, Tapajit Dey, Edwin Lewzey, and Ben Linders . How Agile Teams Can Improve Predictability by Measuring Stability. https://www.infoq.com/articles/improve-predictability-measure-stability/. Accessed: March 28, 2024.

[70] P. N. Robillard and M. Lavallée. Software team processes: A taxonomy. In *2012 International Conference on Software and System Process (ICSSP)*, pages 101–109. IEEE, 2012.

[71] K. Schrum, N. Majury, A. L. Simonelli, and S. Bodgewiecz. Audience matters. *Teaching and Learning Inquiry*, 10, 2022.

[72] S.Druskat, and S. Harriet. What are best practices for research software documentation? https://www.software.ac.uk/blog/what-are-best-practices-research-software-documentation. Published: June 21, 2019, Accessed: January 22, 2024.

[73] B. C. Seagram, L. H. Patten, and C. W. Lockett. Audience research and exhibit development: A framework. *Museum Management and Curatorship*, 12(1):29–41, 1993.

[74] F. Shull, V. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. Mendonça, and S. Fabbri. Replicating software engineering experiments: addressing the tacit knowledge problem. In *Proceedings international symposium on empirical software engineering*, pages 7–16. IEEE, 2002.

[75] J. H. Spaaks, T. Klaver, S. Verhoeven, F. Diblen, J. Maassen, E. Tjong Kim Sang, P. Pawar, C. Meijer, L. Ridder, L. Kulik, T. Bakker, V. van Hees, L. Bogaardt, A. Mendrik, B. van Es, J. Attema, W. van Hage, E. Ranguelova, R. van Nieuwpoort, R. Gey, and H. Zach. **Research Software Directory**. 2020. version: 3.0.1.

[76] Stephan Druskat, Daniel S. Katz, David Klein, Mark Santcroos, Tobias Schlauch, Liz Sexton-Kennedy, and Anthony Truskinger . Credit and recognition for research software: Current state of practice and outlook. Published: November 26, 2018, Accessed:April 24, 2024.

[77] C. Strasser, K. Hertweck, J. Greenberg, D. Taraborelli, and E. Vu. Ten simple rules for funding scientific open source software. *PLOS Computational Biology*, 18(11):e1010627, 2022.

[78] THALES. All about software licensing management. https://cpl.thalesgroup.com/software-monetization/software-licensing-management. Accessed: January 22, 2024.

[79] A. Trisovic, M. K. Lau, T. Pasquier, and M. Crosas. A large-scale study on research code quality and execution. *Scientific Data*, 9(1):60, 2022.

[80] Y. Valdés-Rodríguez, J. Hochstetter-Diez, J. Díaz-Arancibia, and R. Cadena-Martínez. Towards the integration of security practices in agile software development: a systematic mapping review. *Applied Sciences*, 13(7):4578, 2023.

[81] A. Van Der Hoek, R. S. Hall, D. Heimbigner, and A. L. Wolf. Software release management. *ACM SIGSOFT Software Engineering Notes*, 22(6):159–175, 1997.

[82] J. van Eijnatten, M. Barker, V. Azzarà, T. Bakker, M. Cruz, D. S. Katz, C. Martinez-Ortiz, V. Pang, et al. Amsterdam declaration on funding research software sustainability. In *International Funders Workshop*, pages 1–4, 2022.

[83] V. van Hees, Z. Fang, E. Mirkes, J. Heywood, J. H. Zhao, C. P. Joan, S. Sabia, and J. H. Migueles. **GGIR**, Sept. 2022. version 2.7-6.

[84] T. Williams, M. Mercer, J. Mucha, and R. Kapur. Code coverage, what does it mean in terms of quality? In *Annual reliability and maintainability symposium. 2001 Proceedings. International symposium on product quality and integrity (Cat. No. 01CH37179)*, pages 420–424. IEEE, 2001.

[85] Y. Y. J. Woo. Engaging new audiences: Translating research into popular media. *Educational Researcher*, 37(6):321–329, 2008.

[86] B. Xia, T. Bi, Z. Xing, Q. Lu, and L. Zhu. An empirical study on software bill of materials: Where we stand and the road ahead. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2630–2642. IEEE, 2023.