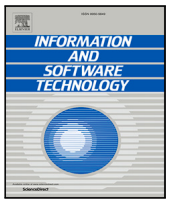
Information and Software Technology 195 (2026) 108138

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



RSMM: A focus area maturity model for research software projects

Deekshitha [a,b,c,*], Rena Bakhshi [b,e], Jason Maassen [b], Carlos Martinez Ortiz [b], Rob van Nieuwpoort [c], Antti Knutas [d], Slinger Jansen [a,d]

[a] Utrecht University, Utrecht, The Netherlands
[b] Netherlands eScience Center, Amsterdam, The Netherlands
[c] Leiden University, Leiden, The Netherlands
[d] LUT, Lappeenranta, Finland
[e] TNO, The Netherlands

ARTICLE INFO

Keywords:
Research software
Research software engineering
Focus area maturity model
Research software project management
Qualitative evaluation

ABSTRACT

Context: Research software is instrumental in producing research results. It plays a special role in advancing scientific discovery through tasks such as data analysis, simulation, and visualization. However, despite its importance, the organizations that produce research software face challenges in managing research software projects. Existing frameworks for software engineering often overlook the needs of research software, including research software engineering practices and open science principles. Without clear guidance, organizations that produce research software must develop and invent new techniques for research software engineering, which is a slow and costly process.

Objective: This work presents RSMM, a maturity model designed to improve organizational practices in research software project management.

Methods: The initial version of RSMM was developed through a systematic literature review. Expert interviews were then conducted to evaluate and refine the model. Finally, multiple case studies were carried out to validate RSMM and demonstrate its applicability in real-world settings.

Results: The final version of RSMM (v1.0) comprises 79 best practices, 17 capabilities, and 10 maturity levels, organized into 4 focus areas: *'Software Project Management'*, *'Research Software Management'*, *'Community Engagement'*, and *'Software Adoptability'*. We provide a comprehensive analysis of RSMM v1.0 and demonstrate its practical applicability through two illustrative case studies.

Conclusion: The RSMM is designed to help organizations in evaluating and improving their research software project management by assessing a project's current maturity level and providing best practices across four focus areas to guide its progression.

1. Introduction

Research software is defined as source code files, algorithms, scripts, computational workflows, and executables created during the research process or for research purposes [1]. It generates, processes, or analyses results intended to appear in a publication, such as a journal, conference paper, monograph, book, or thesis [2]. Despite its importance and widespread use across various research domains, the development and maintenance of research software often remain ad hoc and poorly coordinated, resulting in fragile infrastructures that are challenging to sustain and reproduce [3].

While standard software engineering practices offer many tools for the development and maintenance of software [4,5], they often prove insufficient in addressing the unique requirements of research software. For instance, existing models do not adequately address aspects of research software engineering, such as building user communities, managing licensing, creating training materials, making software easy to adopt in research workflows, and aligning with open-science principles. Furthermore, there are no accredited formal courses or programs in research software engineering [6]. This existing knowledge gap highlights the importance of establishing and promoting best practices for research software projects.

Several initiatives, such as the FAIR (Findable, Accessible, Interoperable, and Reusable) principles for research software [1,7], Research Software MetaData (RSMD) guidelines [8], and Open Source

* Corresponding author at: Utrecht University, Utrecht, The Netherlands.
E-mail addresses: d.kumbla@esciencecenter.nl (Deekshitha), rena@bakhshi.eu (R. Bakhshi), j.maassen@esciencecenter.nl (J. Maassen), c.martinez@esciencecenter.nl (C.M. Ortiz), r.v.van.nieuwpoort@liacs.leidenuniv.nl (R. van Nieuwpoort), antti.knutas@lut.fi (A. Knutas), slinger.jansen@uu.nl (S. Jansen).

https://doi.org/10.1016/j.infsof.2026.108138
Received 20 July 2025; Received in revised form 26 March 2026; Accepted 27 March 2026
Available online 28 March 2026

Security Foundation (OpenSSF) badge program [9] target different aspects of research software development. These guidelines aim to improve the findability, accessibility, interoperability, reusability, and usability of research software and its metadata. However, they are fragmented across various domains and lack a unified framework that offers comprehensive research software project management guidance.

Managing research software is challenging due to limited funding, lack of long-term support, and a widespread lack of digital literacy and software management skills among (non-technical) researchers. As noted by Goth et al. [10], a digital literacy gap persists in academic education, often leaving students and early-career researchers unprepared to communicate effectively with technical colleagues or maintain software beyond the lifecycle of a PhD project. As a result, research software is often no longer maintained after a project ends, and becomes outdated or unusable, which makes it harder to share, reuse, or reproduce scientific work over time. In addition to individual-level challenges, Research Software Engineers (RSEs) and other stakeholders face systemic issues, such as unclear governance structures, insufficient recognition, and pressure to meet funder and policy expectations [11–13]. These diverse demands highlight the need for a framework to support the management of research software projects throughout their lifecycle [14,15].

There exists a line of research on maturity models in software development that offers recommendations to improve software management and governance [5,16]. Two studies have developed focus area maturity models for specific functional domains: software ecosystem governance [17] and API management [18]. In these two models, the authors combined best practices and used the focus area maturity model to measure the maturity of an organization in their functional domain. We were inspired by these two works and developed *Research Software project management focus area Maturity Model (RSMM)*: a maturity model to evaluate and improve research software project management for organizations that produce research software. An organization can vary in the context of research software projects. Some research software projects are run by academic institutes (e.g., CMasher [19]) and some are run by a research software center (e.g., Citation File Format [19]). Some are networks of many organizations that collaboratively contribute, support and fund further research software development (e.g. Invenio [20]). In addition, individual researchers, in some cases in collaboration with (a team of) RSEs, produce essential software for their research [21]. RSMM aims to provide a structured analysis of research software engineering practices that can help organizations that produce research software regardless of their structure.

The contributions of this study are as follows.

- **Maturity model for research software projects:** We present RSMM to assess the maturity of research software projects within research software project management, using best practices from this field.
- **Comprehensive best practices:** The framework includes 79 best practices that cover code quality and security, sustainability,[1] reproducibility, community building, and many other aspects required to evaluate and improve research software project management capabilities. These practices are grouped across four focus areas: *'Software Project Management'*, *'Research Software Management'*, *'Community Engagement'*, and *'Software Adoptability'*.
- **Extension of FAIR principles:** The RSMM framework takes the FAIR principles as a baseline while extending its focus to include aspects such as reproducibility, community building, impact measurement, and other quality factors relevant to open science principles.
- **Community-based approach:** RSMM is developed based on insights gathered from a systematic literature review (SLR) and interviews with researchers, RSEs, and project managers.
- **Practical applicability:** The model is validated through multiple evaluation cycles, including case studies, to ensure its applicability and alignment with stakeholder needs.
- **Published dataset:** RSMM and a complete description of the 79 practices are publicly available as a dataset [23]. This description includes when the practices are implemented and is organized based on the MoSCoW prioritization (Must have, Should have, Could have, Will not have). Additionally, it details the resources required for execution, dependencies among neighboring practices, and literature in which these practices are discussed.

The RSMM should be viewed as a diagnostic and prioritization tool rather than a procedural manual. Its value lies in enabling a shared understanding of project maturity among researchers, research software engineers, and managers. Through the cyclical use of RSMM and software process improvement, research software projects can iteratively improve.

The remainder of the paper is structured as follows. Section 2 introduces maturity models and compares RSMM against existing frameworks. Section 3 provides the study design of RSMM, and Section 4 details the focus areas of RSMM, including a tabular representation of the model. Section 5 illustrates the method for assessing the maturity of research software projects using RSMM on two selected examples. Section 6 provides case study results, and Section 7 presents evaluation results from the expert interview and the case studies. Section 8 provides a discussion and limitations of the study. Finally, Section 9 concludes the paper by summarizing the findings, contributions, and future work.

2. Background and comparison

In this section, we introduce the concept of maturity models and provide some examples that address maturity in certain aspects of project management. Additionally, we summarize best practices in research software engineering and compare them to RSMM.

2.1. Concept of maturity models

Maturity is an evolutionary progress in demonstrating a specific ability or accomplishing a target from an initial stage to a desired end stage [24]. The purpose of maturity models is to guide this evolutionary process by using improvement activities [25]. Pullen (2007) defines a maturity model as *"a structured collection of elements that describe the characteristics of effective processes at different stages of development. It also suggests demarcation points between stages and methods of transitioning from one stage to another"*. These tools are developed for organizations to evaluate and compare, providing a basis for improvement and informed strategies for strengthening capabilities in specific areas [27]. Capabilities encompass the practices and processes within a particular functional domain of the organization. A *capability* refers to the ability to achieve a specific goal tied to a maturity level [28]. Maturity levels outline the expected, desired, or typical evolution path of these capabilities as discrete stages [29]. To reach a particular maturity level, an organization must meet criteria and characteristics related to capabilities or process performance. The initial stage represents a starting point where the organization may have limited capabilities in the assessed domain. At the same time, the highest maturity level indicates organizations that have achieved the most advanced capabilities in that domain [28]. This maturity level assessment of capabilities helps to derive and prioritize improvement measures.

Another type of maturity model is the so-called *focus area maturity model*, as defined by van Steenbergen et al. [28]. It helps organizations to measure their performance in a specific functional domain. A

[1] Throughout this paper, we use *sustainability* to mean technical sustainability as defined in the Karlskrona Manifesto [22].

Table 1
Comparison table: This table compares a few maturity models and guidelines relevant to research software development.

Model	Scope	Structure/Levels	Strength	Limitation
CMMI	Improving organizational processes (beyond software)	5 maturity levels	Broad process maturity model, covers quality assurance, configuration management and others	Ignores openness, community, citation and many other research software-specific practices
OSMM	Open source software (OSS)	3 levels	Assesses OSS trustworthiness	Lacks research software-specific practices
FAIR4RS	Research software principles	Best practice checklist	Covers indexing, metadata, citation, and licensing	Does not consider the research software project's full lifecycle
RSMD	Research software discovery	Best practice checklist	Covers accessibility, metadata, and recognizes developer attribution	Lacks community engagement, sustainability, and other research software management capabilities
OpenSSF badge program	FLOSS Security & quality	145 practices	Mainly testing and security	Lacks sustainability, openness
ADORE.software	Research software policy & funding support	Recommendation based	Research software policy	Lack of Software Adoptability, and many other practices, as the focus is different
Open Source Guides	OSS practice support	Topic based	Community and onboarding support	No research software-specific practices
RSMM	Research software project management	10 maturity levels	Combines research software development, research software-specific, and open source software practices (e.g., value developer attribution)	Does not cover advanced security or test planning

Table 2
Alignment table: This table compares how well other models align with RSMM by showing the proportion of RSMM practices they cover. Alignment is visualized using up to five bubble-filled circles, each representing a range: 0%–19%, 20%–39%, 40%–59%, 60%–79%, and 80%–100%. The final column lists the number of RSMM practices each model includes (out of 79). While models such as CMMI, OSMM, FAIR4RS, RSMD, OpenSSF, ADORE.software, and the Open Source Guide offer valuable practices, most do not fully address research software management.

Model	Software project management	Research software management	Community engagement	Software adoptability	No. of practices
CMMI	●●●●○	○○○○○	●○○○○	○○○○○	11
OSMM	●●●○○	○○○○○	●●○○○	●○○○○	11
FAIR4RS	●○○○○	●●○○○	●○○○○	○○○○○	6
RSMD	●●○○○	●●○○○	●●○○○	●●○○○	13
OpenSSF badge program	●●●○○	●○○○○	●●○○○	●○○○○	17
ADORE.software	●○○○○	●●○○○	●●○○○	○○○○○	7
Open Source Guides	●●●○○	●○○○○	●●○○○	●○○○○	17

functional domain includes activities, responsibilities, and stakeholders involved in fulfilling a well-defined function within an organization and consists of different *focus areas*. A functional domain can be software testing or software management. A *focus area* is an aspect that must be implemented to a certain extent for a functional domain to be effective. Each focus area includes several *capabilities*, and each *capability* provides actions or practices to guide the gradual improvement of the corresponding functional domain. A focus area maturity model is not a fixed-type model; its maturity levels start from 0 and extend to any positive integer. Each focus area is evaluated separately, with its maturity levels.

As shown in Table 1, various maturity models and best practices are compared with RSMM. Table 2 presents their alignment with RSMM practices. The following subsection discusses these comparisons in more detail.

2.2. Comparison

Capability Maturity Model Integration (CMMI) and its predecessor Capability Maturity Model (CMM) are industry standard maturity models [5]. They include five maturity levels. CMM evaluates an organization's software engineering processes through maturity levels. It aims to help developers improve software quality and the overall software engineering process. The scope of CMMI v3.0 extends beyond software development to cover organizational processes such as process quality assurance, configuration management, monitoring and control, planning, estimating, requirements development and management, governance, implementation infrastructure, organizational training, process management, verification and validation. As a result, it involves developers and other departments such as marketing, finance, and purchasing. CMMI covers many quality and security practices, along with some elements of requirements management and limited community engagement. However, CMMI focuses on improving internal organizational processes, which do not directly address aspects of research software development such as openness, sustainability, community involvement, and citation. As a result, it is less suitable for guiding open and collaborative research software practices. CMMI mainly supports internal teams, whereas RSMM emphasizes external community practices such as licensing, community events, and developer onboarding, aspects of open and collaborative research software development.

Open Source Maturity model (OSMM) is an assessment model for the Free/Libre Open Source Software (FLOSS) development process evaluation. It includes 12 trustworthiness elements, grouped into three levels: basic, intermediate, and advanced. However, OSMM remains quite general and does not go into detail about how practices should be implemented. For example, while it includes "Product Documentation" as an element, it does not specify what good documentation should look

like. In contrast, the RSMM includes a capability for documentation and describes how it should support users and enhance usability. Another example is the OSMM element "Results of Assessment of the Product by third Party Companies", which focuses on external evaluations. RSMM includes a similar idea through the practice "Acknowledged by a research software center", offering a community-based form of recognition more suited to the research context. The OSMM aligns with four practices under software project management in RSMM. However, it includes more advanced areas such as test planning, which RSMM does not currently cover. For research software management, OSMM mentions popularity measurement, while RSMM focuses more appropriately on impact measurement, which is more relevant in a research context [7]. In terms of community engagement, OSMM addresses aspects like licensing and user–stakeholder relationships. These correspond to a few of the RSMM practices.

We also examined existing best-practice frameworks for open-source software projects, noting their similarities and differences with RSMM. A few models integrate best practices of research software engineering and open-source software standards: FAIR principles, RSMD guidelines, OpenSSF badge program, Open Source Guides, and ADORE.software.

The FAIR principles (FAIR4RS)[2] are designed for researchers aiming to make their research software Findable, Accessible, Interoperable, and Reusable. These guidelines emphasize creating software that is easy to locate and use across disciplines and research contexts. The FAIR principles primarily focus on enhancing the discoverability and usability of data and software. They do not extensively cover broader software project management or sustainability aspects, making them narrower in scope. However, it covers one practice of that focus area, *'Store project in public repository with version control'*. In *'Research Software Management'*, it covers four out of 22 practices: *'Make code citable'*, *'Enable indexing of project meta-data'*, *'Publish in a research software directory'*, and *'Enable indexing of the project's source code'*. Similarly, *'Community Engagement'* has only one practice *'select a license'* covered by FAIR4RS, and *'Software Adoptability'* has no overlap with FAIR4RS.

The RSMD Guidelines[3] target end-users of research software, including researchers across various disciplines. These guidelines are highly adaptable and flexible, offering user-friendly recommendations that can be applied in different contexts. They cover extensive areas, including accessibility and preservation, licensing, reuse, and legal aspects, emphasizing standardizing metadata to improve software discoverability and usability. However, the guidelines focus less on community engagement or software adoptability. The RSMD framework aligns with a subset of practices from the RSMM across multiple focus areas. It addresses three practices of *'Software Project Management'*, It also supports four practices from *'Research Software Management'*, three from *'Community Engagement'*, and three from *'Software Adoptability'*. However, as outlined in their guidelines, these practices primarily focus on making project data and metadata accessible, and do not cover a significant portion of the practices related to *'Research Software Management'* and other RSMM focus areas, as their objectives and scope differ.

The OpenSSF badge program[4] is designed for FLOSS projects, with a focus on security-conscious developers and users. Its ease of use lies in its checklist-style approach, which allows projects to self-certify their adherence to over 40 best practices. The program's scope is relatively narrow, concentrating on code quality, security, testing, and documentation. OpenSSF includes 17 practices from RSMM and focuses on detailed security and testing practices. In contrast, RSMM includes only two practices related to security and two of testing. However, OpenSSF lacks broader considerations such as community building, sustainability, and research-specific practices.

Open Source Guides[5] is a collection of resources for individuals, communities, and companies to launch and grow open-source projects. They cover aspects such as contributing to open source, starting new projects, building welcoming communities, maintaining projects, getting new users, getting paid for open-source work, legal considerations, and code of conduct. The guides aim to support both newcomers and experienced contributors. While these guidelines apply to open-source research software projects, they lack specific considerations relevant to the research software context, such as sustainability, impact measurement, visibility, ethical considerations, and reproducibility.

The ADORE.software[6] toolkit aims to support the implementation of the Amsterdam declaration on funding research software sustainability. It provides examples of funder programs, policies, and resources for each declaration's recommendations in research software practice, research software ecosystem, research software personnel, and research software ethics. Each focus area outlines recommendations and provides examples of funder programs, relevant funder policies, and resources. However, it does not include specific practices to follow, as it is more resource-oriented.

Existing maturity models and project management frameworks are not designed to address the specific challenges of research software development. While best practices for research software development exist, they are fragmented across domains and lack a comprehensive and actionable framework to guide development. Current frameworks also lack mechanisms to evaluate the maturity of research software project management, making it difficult to assess progress and identify areas for improvement.

To the best of our knowledge, RSMM is the first research software maturity model that combines the best practices of software development and open-source software development while emphasizing research software engineering practices. The focus of RSMM extends beyond the developmental aspects of research software, placing equal emphasis on non-code considerations such as building a community around the research software and enhancing its impact, sustainability, and adoption within the research community.

RSMM is designed from the perspectives of users, researchers, RSEs, funders, and policymakers, addressing the needs of all these stakeholders equally. Our dataset is freely available as an open dataset [23], and includes a complete description of all components of RSMM. The dataset can help individuals with limited software engineering training understand the practices and resources required to implement them. In the following section, we explain the design phases of RSMM.

3. Study design and implementation

Organizations that produce research software often lack unified tools or frameworks designed for research software project management. To address this problem, the study formulates the following research question: *How can organizations that produce research software evaluate their project management practices?*

To develop RSMM, we adopted the maturity model development method proposed by de Bruin et al. [27], which structures model construction into five phases: *Scope*, *Design*, *Populate*, *Test*, and *Deploy*. These phases guided the complete development of the model, as shown in Fig. 1, and are described in detail below.

3.1. Scope

The initial phase in developing a maturity model involves defining its scope, which includes deciding the focus and the target audience. The focus of RSMM is to evaluate and improve research software

[2] https://www.go-fair.org/fair-principles/
[3] https://github.com/FAIR-IMPACT/RSMD-guidelines
[4] https://bestpractices.coreinfrastructure.org/
[5] https://opensource.guide/
[6] https://adore.software/toolkit/

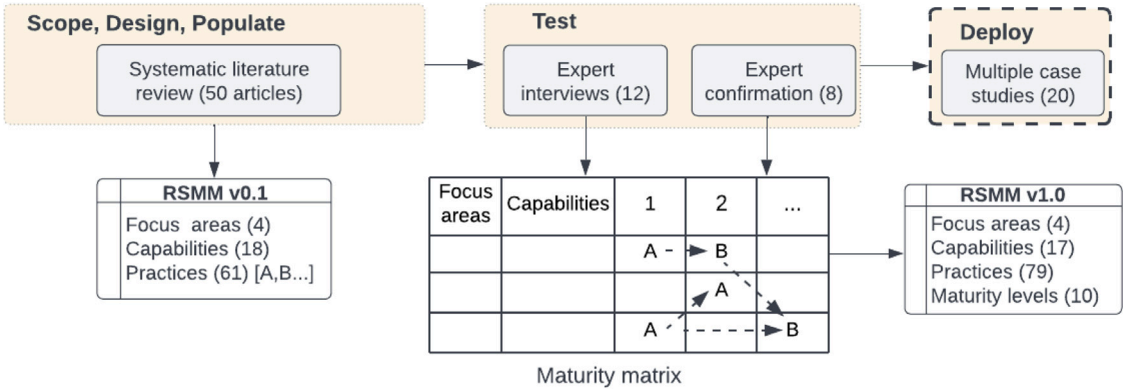


Fig. 1. The development phases of RSMM follow de Bruin et al. [27]: *Scope*, *Design*, and *Populate*, where a literature review led to v0.1. The *Test* phase, includes expert interviews and validation, produced version 1.0 (v1.0). Between v0.1 and v1.0, two intermediate versions were developed. In v0.2, the practices in the maturity matrix were revised, such as added, removed, or merged based on insights gathered from the expert interviews. In v0.3, the practices were positioned within the maturity matrix according to the mode of the responses. In v1.0, we considered dependency among practices. Additional statistical analyses were also conducted based on the collected responses, and these intermediate versions are available upon request. The *Deploy* phase used case studies to assess RSMM's applicability. The practices of RSMM are illustrated with A and B, and arrows represent their dependencies.

project management by identifying and describing practices that contribute to the maturity of research software projects. The target audience of RSMM includes research software-producing organizations, along with a broader group such as RSEs, funders, policymakers, and project managers, who are directly involved in or concerned with research software project management.

3.2. Design

The second phase of the proposed framework defines the overall design and architecture of the model. It considers the needs of the target audience by addressing four questions: **why** they seek to apply the model, **how** it can be applied, **who** should be involved, and **what** outcomes it is intended to support. RSMM is designed to assist organizations that develop research software in evaluating and improving their project management capabilities by assessing and improving project maturity (**why**). It provides a structured, self-assessment-based guideline that allows organizations to implement best practices aligned with their desired maturity level (**how**). The model is applicable to a broad range of organizations, regardless of their structure (**who**). The **what** is demonstrated in Section 6, where we present 20 case studies involving researchers, RSEs, and principal investigators applying RSMM to assess research software projects. These case studies showcase the model's real world applicability and its value in identifying improvement areas.

3.3. Populate

The *Populate* phase involves defining the model's content by identifying what aspects should be measured in the maturity assessment and how this can be measured. To support this process, we conducted an SLR by adapting the methodology developed by [30]. The objective of the SLR was to identify the practices and capabilities of research software project management to design RSMM. We initiated the SLR by searching for keywords related to research software project management. The first and last authors jointly conducted the search across four academic databases: ACM Digital Library, Google Scholar, Scopus, and IEEE Xplore. After applying inclusion and exclusion criteria, we selected 50 relevant articles. We then reviewed the abstracts and key sections of these papers to identify practices and grouped these practices into capabilities. When a paper did not clearly specify a capability, we assigned the practice based on our interpretation and domain knowledge. The full SLR procedure, including the databases used, search strategy, selection process with numbers at each stage, and detailed inclusion and exclusion criteria, is described in the supplementary material [31]. From the final set of 50 selected articles, we identified 61 best practices, which were grouped into 18 capabilities and 4 focus areas based on thematic connections identified in the reviewed literature and discussions between the first and last authors, following both top-down and bottom-up approaches [27]. The validity of these groupings was subsequently verified during expert interviews, where participants reviewed practice-to-capability assignments and suggested reassignments or identified missing practices as needed.

As shown in Fig. 2, we placed practices into RSMM v0.1 without considering their maturity levels. The practices were distributed across consecutive maturity levels, ensuring that no single level contained multiple practices. At this stage, dependencies among practices were avoided. However, in RSMM v1.0, such dependencies are considered, and there is a "natural progression", where higher-level practices typically build upon the implementation of lower-level ones. We incorporated expert feedback to refine the placement of practices within capabilities, focus areas, and maturity levels. This process is described in detail in the *Test* phase.

3.4. Test

Once a model was populated, we conducted 12 expert interviews to determine the positioning of the collected practices and verify the completeness of the model. Table 3 presents details about the interviewees, and the expert selection criteria are as follows: the participants must

- ✓ be knowledgeable in at least two of four focus areas of RSMM.
- ✓ has a minimum of three years of experience in either developing or managing research software projects.
- ✓ work at an organization as an RSE, researcher, or project manager as a part of a team working on research software projects or any comparable role related to research software.

We shared the interview protocol, which details the procedural steps, guidelines, and interview questions, with the selected experts. In addition, the protocol includes a description of the focus areas, capabilities, and practices used in RSMM. As part of the interview study, the Utrecht University Ethics Department reviewed and approved the protocol. The first and last authors of this paper conducted the expert interviews, with the last author participating in 3 out of 12. The interviews were conducted via team communication channel (Microsoft Teams), with one held offline. All interviews were recorded and transcribed for further data processing.

The expert interview included an online card-sorting task [32], where experts categorized practices based on maturity and provided

Focus Area	Capability	1	2	3	4	5	6	7
Software Project Management	Requirements	Requirements are managed explicitly	Authors act on feedback	Roadmap is communicated				
	Code quality	Code reviews are executed	Security reviews are conducted	There are executable tests	Tests are executed in a public workflow	Uses common methods for security evaluation	Project stability is measured continuously	Crash reporting is used
	Communication and Collaboration	A public repository system is used	Has a watering hole for community members					
	Issue management	An infrastructure is available for posting issues	Issues are actively worked on	There is a support team in place				
Research Software Management	Usage costs	Provides documentation on the cost of running the application	Considers energy consumption			Draft		
	Impact measurement	Performs infrequent measurement of impact	Performs continuous impact measurement	Concerned with the privacy aspects of its use	Concerned with the ethical consequences of its use			
	Sustainability	There is a software management plan in place	Obvious source of temporary funding	Viable pathways for project sustainability	Maintained by a research software center	Obvious source of continuous funding		
	Visibility	Code is citable	Project meta-data is indexed (e.g., on Zenodo)	Published in a research software directory	Continuously promoted (e.g., at conferences, Twitter)	Source code is indexed (e.g., SearchSECO, SHG)	Acknowledged by a research software center	Publicly adopted by industrial partners
Research Software Community Management	Partnerships	Informally acknowledges partners and funding agencies	Has an advanced partnership model					
	Community	Onboards researchers as part of the community	Imposes some community norms	Has a code of conduct				
	Developers	Documents how to join the team	Developer names are publicly available	Developers can get access to training and skill development				
	Licensing	A license is carefully selected	License policy is regularly evaluated					
Research Software Adoptability	Ease of use	A simple how to use is provided	A statement of purpose is provided	Online tutorials are provided				
	Documentation	A read me file is provided	Common example usage is provided	A documentation infrastructure is provided				
	Technology	Uses common non-exotic or legacy technology	Technology is easy to use in scientific workflow					
	Reproducibility	Instructions exist on how to put into research workflow	Instructions exist on how to make part of a replication package	Is part of standardized workflows	Is part of a replication package			
	Education	The project is part of an educational program	Generic educational materials have been developed					
	Deployability	The project has an SBOM	Can be deployed on a wide range of technology	Provided with standard deployment tools (docker images, etc.)	Provided as a running service			

Fig. 2. RSMM v0.1 (not the final model): The initial model with 4 focus areas, 18 capabilities, and 61 practices, collected through SLR of academic and gray literature. These practices have maturity levels from 1 to 7 but are not yet organized by maturity.

Table 3
The table summarizes interviewees' current roles, years of experience with research software projects, and expertise in the RSMM's four focus areas. Experience levels are shown with shaded bubbles: black (significant), gray (some), and empty (none). Each interviewee has a unique identifier. Only one interview was conducted in person.

Interviewee	Experience (Years)	Software Project Management	Research Software Management	Community Engagement	Software Adoptability	Interviewee Code
Professor	10+	●	◉	○	◉	IV1
Prof. and Research Scientist	10+	●	●	○	◉	IV2
RSE	10+	●	●	○	◉	IV3
Co-leader and Lead RSE	10+	●	●	◉	◉	IV4
Research Scientist	4	●	◉	○	◉	IV5
Dep. Head of Div. Computing	10+	●	●	◉	◉	IV6
Research Group Leader	10+	◉	◉	◉	○	IV7
Assistant Professor	10+	●	●	◉	◉	IV8
RSE	7	○	◉	◉	◉	IV9
RSE	10+	◉	◉	○	○	IV10
Programme Manager	10+	◉	●	◉	●	IV11
Section Head	10+	●	●	◉	◉	IV12

feedback. We used a Miro dashboard template (an online collaborative workspace) to conduct this task. This template includes initial components of the RSMM, such as focus areas, capabilities, and practices. Practices were written on individual cards, while each capability was represented as a large rectangular box within its corresponding focus area. The practices were not yet organized according to maturity levels and were placed sequentially. The initial template was shared with the first interviewee, who, during the card-sorting task, was asked to move the practice cards into the appropriate capability boxes and arrange them according to maturity levels. After each interview, we updated the Miro template with any new practices suggested by the participant. The revised template was then used in the subsequent interview, following the same procedure. Each interviewee received the updated Miro template, which included all components, with the practices left unarranged according to maturity levels. Through this iterative process, the template was progressively refined as each participant reviewed, categorized, and adjusted the practices. Experts also identified missing capabilities or practices and reclassified them as needed. This iterative process, applied across all focus areas to ensure the model's completeness and accurate classification. Additionally, experts provided feedback on the RSMM's *completeness*, *effectiveness*, *ease of use*,

Table 4
Statistical comparison between two RSMM versions.

Model	Focus Areas	Capabilities	Practices	Modified	Replaced	Merged	Maturity Levels
RSMM v0.1	4	18	61	–	–	–	7
RSMM v1.0	4	17	79	5	3	1	10

Table 5
Inclusion and exclusion criteria for selecting projects for our case studies. Only one explicit exclusion criterion was defined; other fields have no specific exclusions.

Inclusion Criteria (must)	Exclusion Criteria (should not)
be in a public repository	be infrastructure software (DB, OS, Docker, etc.)
be a research software	
be cited in a paper	
have a group of contributors	
deliver working research software	
have documentation written in English	
have a license	
give access to source code	
be in a research software directory	
have a minimum of 25 stars	
be in active development in the last 12 months	

usability, and *operational feasibility* (see Section 7). We followed the same approach used in API-m-FAMM to determine maturity levels. The maturity levels were determined by taking the mode of the rankings assigned by the experts. We also considered dependencies between different practices within a capability and other capabilities in the same focus area, and the various focus areas for the final placement of practices. For example, out of 12 experts, four recommended positioning the *Provide executable tests practice* at level 4. Then, we assigned this practice to maturity level 4, considering its dependencies with another practice, *Execute tests in a public workflow*. It is because the practice, *Provide executable tests* must be implemented or executed before the *Execute tests in a public workflow*. However, additional data was needed for practices introduced in the final interviews to determine their maturity levels. These practices were placed in RSMM v1.0 based on maturity levels suggested in the most recent interviews as a pragmatic solution. Thus, in RSMM v1.0, we positioned practices within the maturity matrix based on expert judgment and dependencies among practices.

Table 4 summarizes changes made to RSMM v0.1 from the *Populate* phase to the *Test phase*. In addition to these changes, we also updated the naming conventions for practices to follow the 'Verb Object (Qualifier)' structure. Additionally, we made minor adjustments to the terminology used for the focus areas and capabilities. Consequently, RSMM v1.0 was sent to the interviewees for confirmation, allowing experts to review the new additions and provide final feedback or suggestions for further refinement.

3.5. Deploy

Following the population and testing phases, the model must be available to assess its generalizability [27]. Accordingly, the *Deploy* phase focused on applying RSMM to research software projects to evaluate its applicability in practice. In this phase, we conducted 50 case studies of research software projects, and we selected these projects based on criteria in Table 5. To collect data for the case studies, we used platforms such as the Research Software Directory (RSD), including those from the Netherlands eScience Center [33], Helmholtz software, Imperial College London, and the Research Software Encyclopedia. We selected these platforms because GitHub lacks a mechanism to differentiate research software from software used in research [34]. Additionally, these platforms offer curated collections of research software projects, and we selected only those that met the inclusion criteria outlined in Table 5. We excluded infrastructure software (e.g., databases and containers), as our study focuses on research-specific tools that advance scientific goals.

We conducted case studies from May 2023 to August 2024. For each case study, we employed multiple data collection methods to ensure comprehensive assessment and triangulation of findings. First, we collected data from GitHub repositories, including practices such as code of conduct, citation guidelines, and licensing. Second, additional practices were identified through searches in research software directories and public information available on project websites, including partner details and crash reports. Third, the preliminary findings were then shared with the case-study participants to confirm the information and fill in any missing details, allowing us to validate documentary evidence against developer knowledge and identify practices not publicly visible. Finally, a short survey was provided to the participants to evaluate the model based on their experience with their project evaluation using RSMM.

The first author initiated contact with the primary representatives of each case study project using the information available on their respective RSD project pages. For projects that did not provide a designated contact person, we identified recent contributors by examining GitHub activity and other publicly accessible sources. These individuals were contacted via email and provided with the preliminary case study report. They were invited either to supply the requested information themselves or to forward the request to the appropriate project maintainer. In several instances, project owners subsequently coordinated internally and encouraged their contributors to complete the required details. During this process, project owners received only descriptions of the practices. For practices that were confusing, participants added clarification questions to the shared Excel spreadsheet. Subsequently, we added assessment questions for each practice in the supplementary material to improve understanding of the practices. The material also includes an example contact spreadsheet.

The final case study results were updated based on the information received from these contributors. For each project, binary values were assigned in the results based on compliance with the corresponding practices.

To illustrate the maturity assessment process, we used two projects, GGIR [35] and ESMvaltool [36], from the Netherlands eScience Center RSD. The results are presented in Section 5. For instance, GGIR met all the inclusion criteria listed in Table 5, had 66 stars (as of April

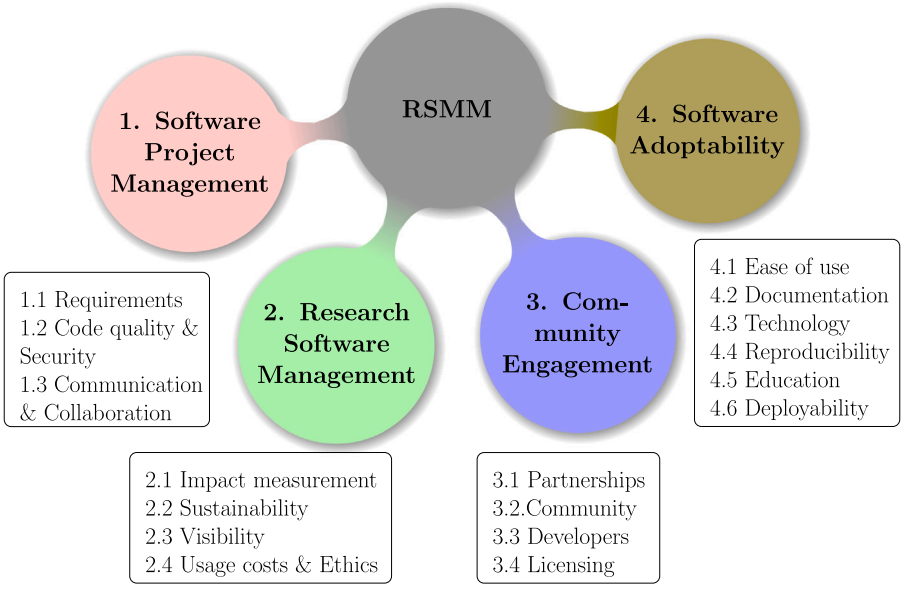


Fig. 3. RSMM v1.0: Four focus areas, 16 capabilities and their codes.

24, 2023), and included the required license and documentation. Additional data for each project was collected from its website, RSD, and GitHub. Binary values were then assigned to indicate adherence to each RSMM practice.

Lastly, the evaluation of RSMM was conducted in three distinct phases: the first phase occurred during the expert interviews, the second phase occurred during the confirmation step, and the final phase was carried out with the case study participants (results of these phases are provided in Section 7). To evaluate RSMM, we used five artifact-quality aspects: *completeness*, *effectiveness*, *ease of use*, *usability*, and *operational feasibility*. *Completeness* assesses whether the model sufficiently covers the relevant practices and capabilities for research software project management. *Effectiveness* evaluates whether the model yields meaningful maturity profiles and supports distinguishing maturity levels across projects. *Ease of use* and *usability* capture whether practitioners can understand the model and apply it without excessive guidance, which is essential for self-assessment settings. *Operational feasibility* assesses whether applying the model is realistic in terms of required effort, information availability, and organizational constraints.

We selected these quality aspects intentionally. Prat et al. [37] emphasize that artifact evaluation criteria are not universal, but must be chosen in relation to the artifact type and the evaluation setting. Their empirical analysis shows that evaluation practice commonly focuses on a limited subset of criteria that can be assessed reliably using naturalistic methods such as interviews and case studies. In our study, expert interviews and real-world case studies provide credible evidence for the five selected aspects, whereas other criteria discussed by Prat et al. (e.g., long-term organizational impact, economic feasibility, or ethical considerations) would require different research designs, longer observation periods, or additional instrumentation beyond the scope of this work.

During evaluation, we collected only qualitative data. Our objective was to obtain a detailed and context-rich evaluation of the RSMM rather than a numerical score. Qualitative responses allowed us to explore whether the current model aligns with the needs of the research software engineering community and to identify potential updates or extensions to the existing practices. Our aim was to gain an in-depth understanding of developers' experiences, perceptions, and challenges when applying the RSMM. The data analysis process is described in Section 7 and [31].

4. Focus area maturity model for research software project management

RSMM covers four focus areas related to research software project management, namely, *'Software Project Management'*, *'Research Software Management'*, *'Community Engagement'*, and *'Software Adoptability'*. Fig. 3 shows these focus areas, capabilities, and corresponding codes.

Fig. 4 provides the full version of RSMM. Each capability within the focus area consists of several practices organized in a maturity matrix based on their maturity level and the dependencies with neighboring practices. When the research software project follows these practices, the corresponding capability improves, achieving maturity in that focus area and improving the research software project capabilities. This section briefly lists capabilities across all focus areas. The taxonomy is structured as follows: "Focus area" (paragraph description), "capability" (listed with brief descriptions), "practice" (selected examples). As *Research Software Management* is a key focus area specific to the academic and research environment (in contrast to general open source or commercial software development), Section 4.2 provides a detailed illustration of the model's structure and documentation approach. This includes an illustrative practice description with its code, MoSCoW categorization, dependencies, and references. The design of RSMM is described in Section 3, and how RSMM can help improve research software project management capabilities is discussed in the following section.

4.1. Software project management

Software project management involves organizing the resources and work activities needed to develop and modify software-intensive systems [38]. Various factors influence the management of software projects, and are equally relevant in the context of research software development. The *'Software Project Management'* focus area of RSMM addresses this by incorporating practices that are fundamental to the development and maintenance of quality software, also in the context of research projects. In total, this focus area includes 3 capabilities and 19 practices aimed at improving aspects such as code quality, security, issue tracking, and fostering collaboration within developer teams.

Key capabilities include managing software project requirements during initial stages (*'Requirements'*), improving code quality and addressing software security (*'Code Quality & Security'*), and fostering

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

Fig. 4. 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.

effective communication within research teams and building collaborative relationships with the broader research community (*'Communication & Collaboration'*). The *'Code Quality & Security'* capability, for instance, includes practices with self-descriptive names such as *'Use automated testing'* and *'Implement continuous integration'*. Together, these practices help ensure that research software is reliable, secure, and maintainable, thereby contributing to the overall quality and reproducibility of scientific research [39].

4.2. Research software management

Although software project management provides the framework for organizing development activities, managing research software presents unique challenges that extend beyond traditional software practices. The focus area *'Research Software Management'* concerns the process of ensuring that software developed in a research project remains usable and maintainable in the long term [40], addressing challenges specific to the academic and research environment. Given the increasing use of research software across various research domains, adherence to best practices has become essential [1]. We identified 4 capabilities and 22 best practices within this focus area that address common challenges such as making research software visible, impactful, and sustainable, as well as promoting ethical development and use.

Key capabilities include evaluating software effectiveness and reach through audience definition and periodic impact assessments (*'Impact Measurement'*), ensuring long-term sustainability through funding strategies, software management planning, and end-of-life policies (*'Sustainability'*), improving visibility through citation, indexing, and software directories (*'Visibility'*), and addressing often-overlooked aspects of privacy, ethical considerations, and cost management (*'Usage Cost and Ethics'*). To illustrate the structure and documentation of practices across all focus areas, we use the *'Visibility'* capability as an example. It includes practices such as *'Make code citable'*, *'Enable indexing of project meta-data'* [1], and *'Publish in a research software directory'* [33]. Fig. 5 describes the latter practice in detail. The practice code "2.3.5" is generated by combining the focus area (2), the capability (3), and the maturity level (5). The description of when this practice is implemented is categorized into MoSCoW categories (with W omitted), including the resources required to complete it, its dependencies on other practices (if present), and references provided within the practice description set. All practices are documented in a similar way within the published dataset.

To provide additional context, the *'Sustainability'* capability focuses on ensuring the long-term sustainability of research software projects through organizational and economic mechanisms such as securing funding, creating comprehensive software management plans, and establishing end-of-life policies. However, software sustainability is a multidimensional concept covering technical, economic, social, and other dimensions (see e.g., [41,42]). Consistent with the Karlskrona Manifesto's framework [22], RSMM addresses these dimensions across

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

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

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 to monitor directory performance, respond to inquiries, and update project information.
- Knowledge or expertise in directory submission processes, metadata standards, and documentation requirements for research software projects.
- Collaboration: With the RSD team to publish projects on their website.

Dependencies:-
References: [46]

Fig. 5. Description set for the practice *'Publish in a research software directory'*. "When implemented" is categorized using the MoSCoW method: M:- Must have, S:- Should have, C:- Could have, W:- Will not have (excluded).

different focus areas: this capability emphasizes organizational and economic sustainability, while *'Software Project Management'* (Section 4.1) and *'Community Engagement'* (Section 4.3) address technical sustainability [43] and social sustainability, respectively. This distribution emerged from our data collection with the research software community.

4.3. Community engagement

A community that evolves and maintains research software fosters an ecosystem of competing and collaborative products. It is shaped by the culture of sharing, collaboration, and collective development of the open-source movement [44]. The focus area *'Community Engagement'* emphasizes strategies for engaging with the research community and fostering a strong, inclusive community around the project. We have identified 4 capabilities and 16 practices associated with this focus area.

Key capabilities include fostering partnerships with research groups and institutions (*'Partnerships'*), encouraging research collaboration and promoting inclusion (*'Community'*), supporting developers through practices for public recognition, collaboration, timely code review, and training (*'Developers'*), and addressing software licensing (*'Licensing'*). As an example, the *'Developers'* capability includes practices with self-descriptive names such as *'Make developer names and roles publicly available'*, *'Document how to join the team'*, *'Set maximum response time for pull requests'*, and *'Provide access to developer training and skill development'*. Additionally, the *'Licensing'* capability addresses licensing concepts that may be unfamiliar in academic environments, including practices such as *'Select a license'*, *'Get institutional support for license choice'*, and *'Evaluate license policy regularly'*. These capabilities and practices collectively support a collaborative and inclusive environment for research software projects.

4.4. Software adoptability

While community engagement builds the collective effort needed to sustain software, successful adoption depends on making the software accessible to potential users. The focus area *'Software Adoptability'* concerns how easily and effectively research software can be adopted and utilized by users. We identified 6 capabilities and 22 practices that address user experience, accessibility, and adoption strategies within the research community.

Key capabilities include providing clear purpose statements and usage instructions (*'Ease of use'*), offering comprehensive documentation (*'Documentation'*), using established technologies that integrate into scientific workflows (*'Technology'*), enabling research workflow integration and replication (*'Reproducibility'*), developing educational materials and training events (*'Education'*), and facilitating deployment through standard tools and broad technology support (*'Deployability'*). As an example, the *'Documentation'* capability focuses on making research software clear and accessible to users. Hermann et al. [45] argue that good documentation is needed to reuse research software. Practices include *'Provide a common example usage'*, which gives users a starting point for experimentation, and *'Provide a documentation as repository/documentation wiki'*, which uses a wiki platform to share comprehensive and detailed information about a project [46], helping individuals create, edit, and organize information in a structured way.

4.5. How to use the RSMM

Algorithm 1 describes a step-by-step process for conducting a maturity assessment of research software projects. The assessment starts by selecting a research software project whose maturity in research software project management is to be evaluated and initiating the

Table 6
Mapping RSMM levels to learning types (mental model perspective).

RSMM level range	Type of learning	Description
1–3	Single loop	Basic practices: start with existing practices and improve them without changing underlying assumptions.
4–6	Transitional- Single loop to Double loop transitioning	From basic to advanced: Intermediate practices emerge, but advancing further typically requires proper guidance and support
7–10	Double loop	Advanced maturity requires new values, new assumptions, and resources.

assessment using RSMM v1.0. To evaluate practices, the assessor transforms each practice defined in the model into a self-assessment question (assessment questions are documented in [31], Section Assessment) to determine whether the practice is implemented in the selected project.

The assessor then examines whether the project follows the practices outlined in the first focus area of the maturity model and records "Yes" if a practice is implemented and "No" otherwise. Only cells containing practices are considered during the evaluation, while empty cells are ignored. After completing the evaluation for a focus area, the maturity level is determined by inspecting the maturity columns. Levels that contain no practices are skipped, and all practices at lower levels must be marked "Yes" to attain a higher level. The maturity level is defined as the highest level in the longest continuous sequence for which all practices are marked "Yes." For example, if the maturity levels of the first, second, and third levels are 3, 2, and 2, respectively, the maturity level of the focus area is 2. The maturity levels are cumulative, meaning that all practices at lower levels must be implemented before a higher maturity level can be attained. This process is repeated for each focus area, while any focus areas not relevant to the assessment may be excluded.

Algorithm 1 Maturity Measurement for Any Research Software Project

Input: Software project
Output: Maturity level of the project

1. *Select any Project:* Choose the software project for which you want to measure maturity.
2. *Start Maturity Model Assessment:* Initiate the assessment of the project using RSMM v1.0.
3. *Evaluate Practices:* Examine whether the selected project follows the practices outlined in the first focus area of a Maturity Model. If a practice is followed, mark 'yes'; otherwise, mark 'no'. Consider only the cell with practices.
4. *Check Maturity Levels:* Inspect the maturity columns for the focus area. If a level contains no practices, skip it; all practices at lower levels must be marked "Yes" to attain a higher level. Determine the maturity level as the highest level in the longest continuous sequence for which all practices are marked "Yes."
5. *Return:* Maturity level of the project for the particular focus area.
6. *Repeat:* Repeat Steps 3, 4 and 5 for all other focus areas.
7. *Skip:* Skip any focus area that you do not want to include in the evaluation.
8. **Return:** Maturity level of the project.

4.6. Theoretical foundation

While in the previous section we described *how* to use RSMM, understanding *why* it works requires theoretical grounding. Research software teams face challenges not only in determining what to improve, but in achieving consensus on priorities and action strategies. Action theory provides a framework for understanding how maturity models address these challenges by enabling shared reasoning about practice-consequence relationships. This practice-consequence reasoning is captured in Argyris and Schön's definition of theories of action [47].

According to their framework, a theory of action can be expressed as: "In situation S, if you want consequence C, under assumptions $a_1 \ldots a_n$, do action A." In RSMM, the situation (S) corresponds to the current maturity level, the consequences (C) are the capabilities, and the action strategies (A) are the specific practices implementing each capability. The assumptions ($a_1 \ldots a_n$) reflect the project's mental model, including priorities, values, and perceived feasibility.

Based on this mapping, we can interpret the patterns in practice adoption using Argyris and Schön's concepts of single-loop and double-loop learning (see Table 6). Single-loop learning occurs when projects refine or optimize practices without changing underlying assumptions or values. Double-loop learning occurs when projects question and modify the assumptions themselves, resulting in deeper changes in priorities, governance, or mental models. In our case study, adoption of foundational practices at levels 1–3 aligns with single-loop learning, while low adoption of advanced practices at levels 7–10 suggests that double-loop learning is constrained by resource or organizational limitations. Levels 4–6 represent a transitional phase, where projects begin shifting from single-loop to double-loop learning.

5. GGIR and ESMValTool Project Assessments using RSMM

We selected two research software projects, GGIR and ESMValTool from a set of 50 case studies conducted to illustrate the assessment of maturity levels using RSMM v1.0. The results from several other case studies are discussed in more detail in Section 6.

We collected data related to GGIR and ESMValTool from their websites and GitHub repositories. Based on these data, we evaluated both research software projects using RSMM v1.0, resulting in maturity scores of 4-3-6-7 for GGIR and 5-4-8-8 for ESMValTool across the corresponding focus areas. The details of our initial assessment are presented in Table 7. The tick mark in the practices box indicates that the tool follows those practices, whereas the cross mark indicates that it does not. Assessing whether a project follows the practices outlined in RSMM v1.0 helps determine its maturity. Maturity scores are determined by evaluating the longest continuous sequence of implemented practices within each focus area.

The report was sent via email to the repository owners or contributors to validate the evaluation. After they reviewed the case study results, the evaluation report was slightly revised. Several practices were updated, such as in the case of GGIR, where practices such as *'Define code coverage targets'*, *'Execute tests in a public workflow'* and *'Evaluate whether the audience's goals are met'*, which were previously marked as 'not followed'. However, the repository owners indicated compliance with these practices and provided links to the corresponding documentation on GitHub and their project websites. We validated their responses by examining these sources and revised our data accordingly.

Additionally, the practice, *'Provide with standard deployment tools'* was initially marked as 'following', but after the review, it was updated to 'not following.' We revised the maturity level of the *'Software Adoptability'* focus area from 6 to 5 based on feedback from repository owners. These changes highlight the importance of validating our case

Table 7
Assessment of the research software GGIR (left) and ESMValTool (right) using RSMM v1.0: A tick mark means the tool follows the practice, whereas a cross mark means it does not. Gray shading of the cells highlights the longest continuous sequence of implemented practices in each focus area, which determines the achieved maturity level.

Focus Area	Capability	1	2	3	4	5	6	7	8	9	10
1	1.1		✔	✔				✔	✔		✗
	1.2	✔	✔	✔	✔	✗	✗	✔	✗	✗	✗
	1.3			✔				✔	✗		✗
2	2.1		✔	✔				✗		✔	✗
	2.2		✔		✗			✔	✗	✗	✗
	2.3	✔		✔	✔	✔	✔	✔			✔
	2.4				✔	✗	✗	✗			
3	3.1				✔			✗			
	3.2			✔	✔	✔	✔	✗		✗	✗
	3.3		✔				✔		✗	✗	
	3.4	✔		✔					✗		
4	4.1		✔	✔			✔				
	4.2		✔	✔	✔		✔		✗		
	4.3		✔		✔					✗	
	4.4		✔		✔	✔		✔			
	4.5					✔				✔	✔
	4.6						✔		✔	✗	✗

(a) The maturity level of GGIR is 4-3-6-7

Focus Area	Capability	1	2	3	4	5	6	7	8	9	10
1	1.1		✔	✔				✔	✔		✔
	1.2	✔	✔	✔	✔	✔	✗	✔	✔	✔	✗
	1.3			✔				✔	✔		✔
2	2.1			✔	✔			✔		✔	✗
	2.2		✔		✔			✔	✔	✔	✗
	2.3	✔		✔	✔	✔	✔	✔			✗
	2.4				✔	✗	✗	✗			
3	3.1				✔			✔			
	3.2			✔	✔	✔	✔	✔		✗	✗
	3.3		✔				✔		✔	✔	
	3.4	✔		✔					✔		
4	4.1		✔	✔			✔				
	4.2		✔	✔	✔		✔		✔		
	4.3		✔		✔					✗	
	4.4		✔		✔	✔		✔			
	4.5					✔				✔	✗
	4.6						✔		✔	✗	✗

(b) The maturity level of ESMValTool is 5-4-8-8.

Table 8
Updated assessment of the research software GGIR (left) and ESMValTool (right) using RSMM v1.0 (after review by the repository owners). Changes in the practices are differentiated using yellow color.

Focus Area	Capability	1	2	3	4	5	6	7	8	9	10
1	1.1		✔	✔				✔	✔		✗
	1.2	✔	✔	✔	✔	✗	✗	✔	✔	✔	✗
	1.3			✔				✔	✗		✗
2	2.1		✔	✔				✔		✔	✗
	2.2		✔		✗			✔	✗	✗	✗
	2.3	✔		✔	✔	✔	✔	✔			✔
	2.4				✗	✗	✗	✗			
3	3.1				✔			✗			
	3.2			✔	✔	✔	✔	✗		✗	✗
	3.3		✔				✔		✗	✗	
	3.4	✔		✔					✗		
4	4.1		✔	✔			✔				
	4.2		✔	✔	✔		✔		✔		
	4.3		✔		✔					✔	
	4.4		✔		✔	✔		✔			
	4.5					✔				✔	✔
	4.6						✗		✔	✗	✗

(a) The maturity level of GGIR is 4-3-6-5

Focus Area	Capability	1	2	3	4	5	6	7	8	9	10
1	1.1		✔	✔				✔	✔		✔
	1.2	✔	✔	✔	✔	✔	✗	✔	✔	✔	✗
	1.3			✔				✔	✔		✔
2	2.1			✔	✔			✔		✔	✗
	2.2		✔		✔			✔	✔	✔	✗
	2.3	✔		✔	✔	✔	✔	✔			✗
	2.4				✔	✔	✗	✔			
3	3.1				✔			✔			
	3.2			✔	✔	✔	✔	✔		✗	✗
	3.3		✔				✔		✗	✔	
	3.4	✔		✔					✔		
4	4.1		✔	✔			✔				
	4.2		✔	✔	✔		✔		✔		
	4.3		✔		✔					✗	
	4.4		✔		✔	✔		✔			
	4.5					✔				✔	✔
	4.6						✔		✔	✔	✗

(b) The maturity level of ESMValTool is 5-5-7-8.

study results. Due to these changes, we included only the 20 validated case studies (out of 50 evaluated) in Section 6. Table 8 presents the final evaluation results for GGIR and ESMValTool. The new maturity levels of GGIR and ESMValTool are 4-3-6-5 and 5-5-7-8, respectively. GGIR adheres to 66% of the best practices outlined in RSMM, while ESMValTool follows 88.5% of them.

GGIR achieves a moderate level of maturity in *'Software Project Management'* but requires further improvement in capability *'Code quality and Security'* to achieve higher maturity. However, it shows a high level of maturity in *'Community Engagement'*, indicating they followed well-defined strategies to build or develop a community around the tool. ESMValTool also achieves a moderate level of maturity in *'Software Project Management'* due to its lack of adherence to security practices. However, as an open-source community-developed tool for climate science, it excels in *'Community Engagement'* and *'Software Adoptability'* and has implemented numerous practices in these focus areas to improve its capabilities.

Case Study Highlight: GGIR

GGIR is an open-source R package designed to process and analyze multi-day raw accelerometer data collected from wearable devices, with a focus on applications in physical activity, sleep, and circadian rhythm research. GGIR was initiated in 2013 by researchers in the field of epidemiology and movement science. Since then, GGIR has become a widely adopted tool in academic studies, clinical research, and large-scale cohort studies, such as the UK Biobank.[a] Its adoption is supported by an active research community, with more than 100 scientific publications referencing it. Currently, it has over 40 contributors, though development is largely driven by a core group of 3–5 active developers.

[a] https://journals.humankinetics.com/view/journals/jmpb/2/3/article-p188.xml

Case Study Highlight: ESMValTool

ESMValTool (Earth System Model Evaluation Tool) is an open-source, Python-based software package for evaluating and comparing Earth System Models (ESMs). It supports reproducible benchmarking by validating model outputs against observational and reanalysis datasets. Widely used in the European climate modeling community, ESMValTool has more than 80 contributors, regular development activity, and more than 23 official releases. Its modular design and extensive diagnostics library make it a key tool in climate data analysis, supporting research, policy, and environmental monitoring.

Thus, using RSMM, users can evaluate the research software project management by identifying areas that need improvement and practices that need to be followed to achieve higher maturity. For example, in the case of GGIR, evaluating the *'Code quality and Security'* capability of the code within the focus area of *'Software Project Management'* reveals the absence of a few practices, such as *'Use crash reporting'*, *'Conduct security reviews'*, and *'Follow industry standard for security'*. The lack of these practices currently prevents GGIR from achieving a high maturity level in this focus area.

These two case studies illustrate how RSMM facilitates project maturity assessment and helps identifying appropriate next steps or areas for improvement. This observation is corroborated from interview and case study participants. It may assist users in choosing existing software for adoption and allow them to track progress and improvements over time. RSMM can be used by research organizations and funding agencies to assess the maturity and viability of research software projects, allowing them to decide which projects to support and fund. Additionally, policymakers can utilize RSMM to review and improve policies as needed based on the evaluation of the previous projects.

The maturity assessment provides a foundation for structured improvement planning. The model does not prescribe how practices should be implemented, as this falls within the domain of software process improvement (SPI) [48]. Instead, stakeholders can use the results to identify capabilities where maturity is low, define concrete goals for change, such as improving test automation or community documentation. Based on these goals, they can develop an SPI plan, identifying actions, responsibilities, and timelines to gradually increase their maturity level in the chosen focus area [49].

In practice, this means the RSMM should be viewed as a diagnostic and prioritization tool rather than a procedural manual. Its value lies in enabling a shared understanding of project maturity among researchers, research software engineers, and managers. Through the cyclical use of RSMM and software process improvement, research software projects can iteratively improve.

6. Case studies: 20 research software project results

The case studies were conducted during the *Deploy* phase to evaluate the applicability of RSMM in research software projects. We selected 50 projects from RSDs and a team of 8 Ph.D. students evaluated 40 projects, while the first author evaluated 10 projects. The assessment was conducted from May 2023 to August 2024, with sessions lasting between 30 and 180 min. Data were collected from each project's GitHub pages, websites, and other online resources, and the results were compiled into an Excel spreadsheet. We were unable to identify some practices from available online sources, so we sent all 50 initial reports to the respective project owners or contributors for validation and completion. This took place between June and August 2024. Each communication included the draft report and the published dataset to familiarize reviewers with RSMM. We also scheduled one-on-one meetings with four participants to clarify doubts and discuss practices they found unclear. After validating the reports, participants completed a brief survey to assess RSMM's operational feasibility, ease of use, usefulness, and effectiveness. As a result, we obtained 20 complete validation reports, summarized in Table 9. The table lists the number of practices implemented for each capability and the percentage of practices implemented in each focus area.

A brief description of each project is provided in [31]. The selected projects vary in research domain, purpose, number of contributors, and usage, reflecting the broad spectrum of objectives within the research landscape. Some projects are supported by active contributor communities (such as 3D Slicer), others by national research software centers (such as ESMValTool), while a few are largely maintained by individual contributors (such as GGIR and Ocean Parcels). Several projects lack consistent funding, human resources, or time dedicated to project management, which is reflected in the results. Projects supported by active contributor communities or national centers tend to follow most best practices and achieve higher maturity levels according to our evaluation using RSMM. For example, the 3D Slicer project started more than 10 years ago and is supported by a team of developers and sustained funding; it has implemented approximately 90% of the practices in RSMM.

Several projects do not fully implement certain capabilities within specific focus areas. Discussions with project owners indicate that these projects are nevertheless working to improve sustainability and foster collaboration through initiatives such as training activities and community outreach aimed at attracting funding. Our evaluation also shows that some practices, such as front-page chat support, are often considered non-essential for research software projects. Case study participants reported using alternative communication channels, such as GitHub Discussions, instead of front-page chat support.

Many practices are difficult to implement for smaller projects with limited contributors, funding, or organizational support. Participants indicated that some practices do not align with the project's current goals or priorities. In particular, many projects only partially address practices related to the focus areas *'Research Software Management'* and *'Community Engagement'*, which received average scores of 60% and 62%, respectively. These scores are lower compared with the other two focus areas. Insights from the expert interviews and feedback from case study participants further indicate that researchers and RSEs generally adhere to core software project management practices. Fig. 6 provides an overview of the projects adhering to the practices outlined in RSMM.

The heatmap shows that a larger proportion of projects adhere to practices associated with maturity levels 1 to 3. These levels represent foundational practices that help establish a structured research software project before progressing to more advanced stages. At maturity levels 4 to 6, projects implement practices to a moderate extent. While these projects demonstrate a higher level of maturity, not all practices are consistently adopted. Adoption decreases further at maturity levels 7 to 10, where practices require greater organizational maturity, resources, or institutional support. Examples include *'Follow an industry standard for security'*, *'Provide newsletter'*, *'Define end-of-life policy'*, *'Garner industrial partner adoption'*, and *'Generate Software Bill of Materials (SBOM)'*. These findings suggest that advanced practices remain difficult to implement for many research software projects.

None of the case study participants indicated that these advanced practices were irrelevant. Instead, participants acknowledged their importance but noted that limited funding, time, and human resources constrain their ability to implement them. Practices related to the capabilities *'Usage costs and Ethics'* (2.4) and *'Reproducibility'* (4.4) are therefore less widely adopted. In contrast, capabilities such as *'Ease of use'* (4.1), *'Documentation'* (4.2), *'Visibility'* (2.3), and *'Requirements'* (1.1) are implemented by a larger number of projects.

There is a strong correlation between the focus areas *'Software Project Management'* and *'Software Adoptability'*. When one of these areas is effectively managed, the other tends to perform well, so that many practices from that focus area are followed. The capability *'Code quality and Security'* achieved the highest average value of 6.5 compared to other capabilities. This suggests that the practices of this

Table 9
Results of 20 case studies: The table shows the number of practices implemented per capability. The number of practices per capability and focus area is included in parentheses. Cells with the highest and lowest averages are also highlighted in green and red. Please note that the low percentage indicates missing RSMM practices, not poor performance; the maintainers plan to update future releases to better reflect some of these practices.

	Focus area	GGIR	ESMvalTool	BridgeDb Java	Kernel tuner	DIANNA	PopPUNK	Scholia	DALES	Litstudy	3D slicer	MAGMA.Celltyping	Ocean Parcels	Haddock3	muscle3	astra-toolbox	ewatercycle	oemof-solph	pyglotaran	MSS	SUMO	Avg	Stdev
1	Software Project Management (19)	68%	89%	63%	89%	79%	63%	42%	47%	42%	100%	53%	79%	89%	68%	47%	79%	84%	74%	95%	79%	72%	
1.1	Requirements (5)	4	5	4	5	5	5	4	5	4	5	4	5	4	4	4	5	5	4	5	3	4.5	0.59
1.2	Code quality & security (10)	7	8	6	9	7	6	3	3	3	10	5	8	9	7	3	7	7	7	9	7	6.5	2.06
1.3	Communication & collaboration (4)	2	4	2	3	3	1	1	1	1	4	1	2	4	2	2	3	4	3	4	4	2.6	1.07
2	Research Software Management (22)	55%	82%	59%	59%	45%	55%	45%	45%	45%	95%	36%	68%	68%	55%	68%	63%	77%	50%	73%	59%	60%	
2.1	Impact measurement (5)	4	4	4	3	3	4	3	3	3	5	3	5	4	3	5	5	5	4	4	2	3.8	0.89
2.2	Sustainability (6)	1	5	2	2	2	1	1	2	2	6	1	2	4	2	3	3	5	1	5	4	2.7	1.55
2.3	Visibility (7)	7	6	6	7	5	6	6	4	5	7	4	7	7	6	5	6	7	6	7	7	6.1	0.97
2.4	Usage costs & Ethics (4)	0	3	1	1	0	1	0	1	0	3	0	1	0	1	2	0	0	0	0	0	0.7	0.95
3	Community Engagement (16)	56%	88%	31%	50%	56%	56%	56%	44%	25%	94%	31%	75%	75%	75%	69%	63%	83%	63%	75%	75%	62%	
3.1	Partnerships (2)	1	2	1	0	1	1	1	0	0	2	1	1	1	1	1	1	1	0	2	1	0.95	0.63
3.2	Community (7)	4	5	1	4	3	5	4	3	1	7	2	6	6	5	5	4	7	6	5	6	4.5	1.84
3.3	Developers (4)	2	3	1	2	3	1	2	2	1	3	1	3	3	3	2	3	2	2	3	3	2.3	0.84
3.4	Licensing (3)	2	3	2	2	2	2	2	2	2	3	1	2	2	3	3	2	3	2	2	2	2.2	0.53
4	Software Adoptability (22)	86%	95%	41%	73%	64%	64%	55%	64%	45%	91%	41%	95%	77%	77%	73%	77%	68%	82%	68%	82%	71%	
4.1	Ease of use (3)	3	3	3	3	3	3	3	3	3	3	1	3	3	3	3	3	3	3	3	3	2.9	0.44
4.2	Documentation (5)	5	5	2	5	5	5	4	5	5	5	4	5	5	5	5	5	5	5	4	5	4.7	0.72
4.3	Technology (3)	3	3	2	2	1	2	2	2	1	3	1	3	3	3	3	2	3	2	2	3	2.3	0.70
4.4	Reproducibility (4)	4	4	0	1	0	1	0	1	0	3	1	4	1	2	1	1	0	3	0	2	1.5	1.35
4.5	Education (3)	3	3	0	3	3	1	1	2	0	3	0	3	3	2	2	3	2	3	3	3	2.2	1.10
4.6	Deployability (4)	2	3	2	2	2	2	2	1	1	3	2	3	2	2	2	3	2	2	3	2	2.1	0.59

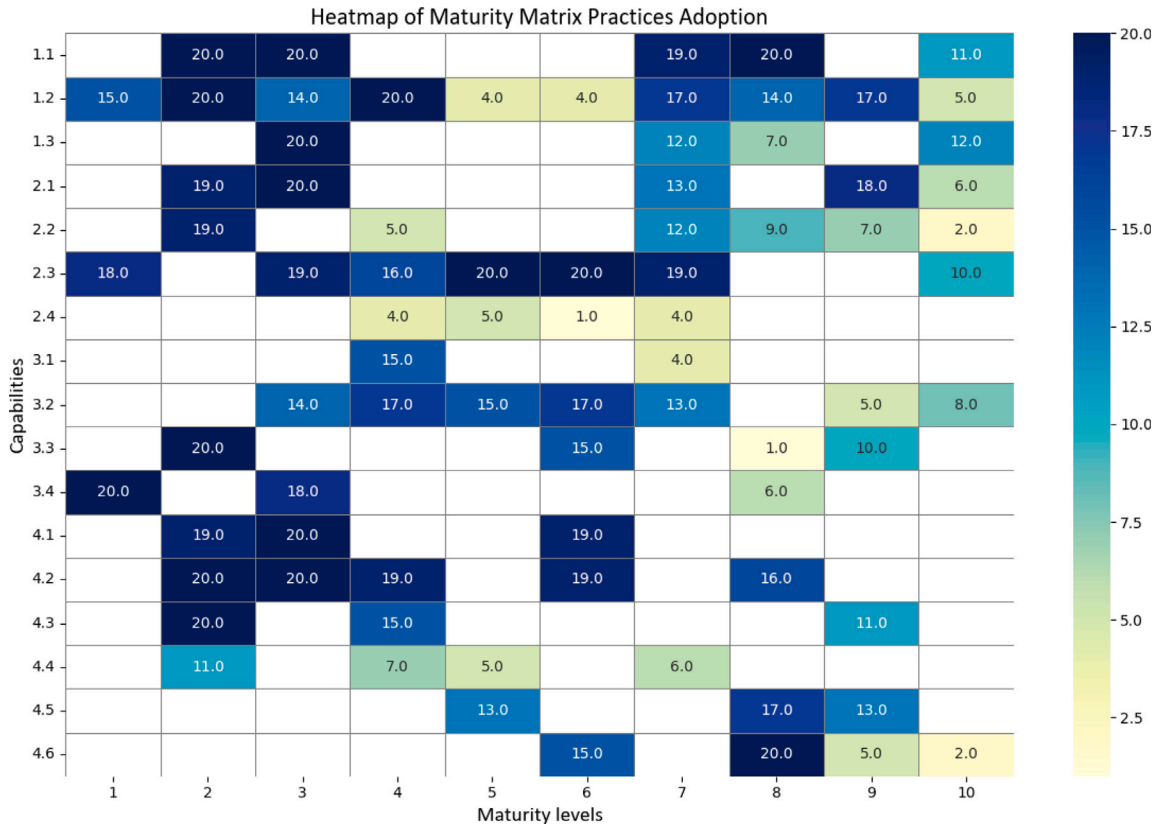


Fig. 6. The statistics show how many projects follow each practice, and the legend explains the color variations (right). In the first capability (1.1 Requirements), for instance, we observe that all 20 projects fulfill the requirements for 3 of the practices, e.g., "File issues in an issue tracker".

capability are well-established, more straightforward to implement, and considered important. In contrast, *'Usage costs and Ethics'* capability has an average value of 0.7, the lowest value among other capabilities, indicating that many projects fail to follow those practices or consider them less important in their work. The assessment prompted teams to plan or raise GitHub issues for future implementation of a few practices, such as a code of conduct.

The case study results show that RSMM is applicable across research domains and supports the evaluation and improvement of research software project management through widely adopted and useful practices. However, case study participants identified a few challenges related to the adoption of RSMM, which are discussed in Section 8.

7. Evaluation of RSMM

The evaluation of RSMM is carried out in three phases: (1) expert interviews with 12 participants, (2) a survey with 8 interview participants, and (3) another survey with 20 case study participants. We applied the following characteristics to evaluate RSMM: *completeness*, *effectiveness*, *ease of use*, *usability*, and *operational feasibility*.

Table 10 presents the evaluation questions used across the three evaluation phases and their corresponding evaluation dimensions. All survey questions were designed to operationalize the evaluation dimensions and to capture descriptive feedback. In Evaluation Phase 2, responses were primarily binary (Yes/No), as this phase focused on validating RSMM v1.0 and confirming participant agreement.

The qualitative data was analyzed using thematic analysis [50] with the support of ATLAS.ti, a qualitative analysis software. We began by defining an initial set of themes derived from the survey questions and aligned with our devised evaluation criteria. Each response was first coded against these predefined themes using deductive coding. Additionally, ATLAS.ti assisted in identifying additional emergent themes, which were subsequently reviewed and refined through manual coding. Both positive and negative sentiments were detected using the software's automated features and then manually validated. While responses containing straightforward statements were easily categorized, several responses were more complex and addressed multiple questions simultaneously. These were analyzed in greater depth, with semantic coding applied to capture the full range of meanings and ensure that all relevant themes were appropriately represented. The questions, the responses of the participants, and the extracted themes are detailed in [31].

Table 11 summarizes the RSMM evaluation results. Regarding the *completeness* of RSMM, 6 out of 12 experts agreed that it covers most of the aspects of maturity, with some describing it as comprehensive and inclusive of various dimensions. Another noted its broad coverage of practices and multiple aspects, including use and quality. However, one interviewee (IV6) pointed out that RSMM shares similarities with other maturity models, remarking, *"In some sense, yes. It's one of many other maturity models that checks this"*. 10 case study participants agreed that RSMM covers many of the best practices of the research software project management process without noticeable omissions. One respondent (CP8) stated, *"It looks very thorough and seems to cover many aspects"*. However, two experts and three case study participants shared negative views, expressing concerns that there is a risk of requiring research software projects to follow all the practices of RSMM, which could be burdensome and impractical. We received seven positive responses from the experts related to the *usefulness* of RSMM, and participants agreed that RSMM is valuable. They indicated that the model can effectively guide users and inspire improvements in the capabilities of the research software management.

Furthermore, an interviewee (IV4) viewed RSMM as a *"good starting point"* for identifying considerations at various stages of the maturity of a research software project: *"I think it is a good starting point to collect ideas what a software project should consider in its various*

Table 10
Evaluation questions and their corresponding evaluation dimensions.

No.	Questions	Evaluation dimensions
	Evaluation phase 1	
1.	Do you think this model will accurately reflect the maturity of research projects that develop research software?	Completeness
2.	Do you think this model helps organizations improve their research software production practices?	Effectiveness
3.	How easily can organizations adapt this model to improve their practices?	Ease of use
	Evaluation phase 2	
4.	Does RSMM accurately represent our discussions?	Effectiveness
5.	Do you find RSMM useful?	Usefulness
6.	Do you anticipate using RSMM (can be practically implemented) in your future work?	Operational feasibility
	Evaluation phase 3	
7.	Is the evaluation useful for your project?	Usefulness
8.	Is RSMM complete?	Completeness
9.	Do you intend to use RSMM to manage your project?	Operational feasibility

Table 11
Evaluation of RSMM. The following symbols are used to reflect the sentiments in the comments according to an evaluation characteristic: ✓ - strong support, ↔ - a mixed response, ✗ - strong opposition. Interview participants were assigned codes starting with 'IV', and case study participants were assigned codes starting with 'CP'.

Quality	#	Example Excerpt(s)
		Evaluation phase 1
Completeness	✓ 6	**IV9** *...you [RSMM] cover a broader range of, yeah, many practices, . . . And also different aspects of the software, not only the use and quality.*
	↔ 4	**IV3** *...I think it touches on a lot of aspects that go into the maturity of projects. Not all the aspects are necessarily applicable to each project.*
	✗ 2	**IV7** *...I think there is a risk of like requiring research software projects to respond to each of those 64 yeah questions or aspects...*
Effectiveness	✓ 4	**IV3** *I do think it can help. Having such a model can make it explicit for an organization to know what they should be doing.*
	↔ 4	**IV7** *...to a certain degree, but I don't think this would be a game changer.*
	✗ 4	**IV9***...you don't need the software to be very mature. And where there's nothing on the goal of the software, but...I think this is maybe for general software.*
Ease of use	✓ 6	**IV3** *easy. For applications, its checklists are much easier...*
	↔ 2	**IV9** *...I think if you can somehow find like proxies for these practices that you can measure easily and automatically that would really help.*
	✗ 4	**IV5** *...I would not say it is easy to adopt...I don't think organizations can do all of them.*
		Evaluation phase 2
Effectiveness	✓ 5	**IV10** *I think it's accurate.*
	↔ 3	**IV4** *There are some deviations from the order I would have in mind and which I mentioned in our discussions but this is certainly to be expected since the outcome is the aggregate of various views on that aspects.*
	✗ 0	...
Usefulness	✓ 7	**IV6** *Yes, the model is very useful. I can imagine that many RSEs will be very happy to have the list of criteria for their work.*
	↔ 1	**IV7** *In principle yes, but for most real-life choices to be made, it'll only be realistic to use if the assessment is provided and doesn't have to be done by the user.*
	✗ 0	...
Operational feasibility	✓ 5	**IV6** *Yes, I will use input from this model in my future work.*
	↔ 1	**IV4** *Potentially yes. Use case would be to use it to cross-check for an existing software project which aspects should probably be considered next.*
	✗ 2	**IV10** *Not directly no.*
		Evaluation phase 3
Usefulness	✓ 12	**CP1** *Yes, it was useful to have such a comprehensive and systematic overview of the different capabilities and focus areas.*
	↔ 6	**CP2** *To some extent, surely. It gives an overview of the options for improvement, despite the fact that some are unattainable for us*
	✗ 2	**CP5** *Not really, not the whole scope of the project was captured*
Completeness	✓ 10	**CP7** *We consider it very comprehensive and detailed. We do not see anything missing right now.*
	↔ 7	**CP10** *Hard question, but it seems to capture the status of a software quite well. The only thing that struck me as odd was the idea with 'chatbot support on the project website'.*
	✗ 3	**CP12** *It's hard to tell, but I think completeness should not be the focus perhaps.*
Operational feasibility	✓ 6	**CP12** *I do like that the scope of the model is a bit broader than only the software, also including a focus on education, enabling (reproducible) research workflows, listing partners and funding resources and such.*
	↔ 12	**CP15** *I would use to see how well it would score, but I would also use my own intuition [...] whether [...] to implement certain practices.*
	✗ 3	**CP6** *Haven't generally got capacity to deal with anything beyond urgent issue implementation, for existing packages.*

stages.". However, a challenge identified by one of the experts was that the users might face difficulty evaluating all the criteria independently due to the extensive number of practices in the model. Additionally, 12 case study participants found that RSMM provides a comprehensive and systematic overview of capabilities and focus areas and offers a good framework for evaluation. Participants appreciated its *usefulness*, describing it as helpful and valuable to have this mapping for practices and maturity levels. From the expert's perspective, RSMM is a tool for organizations, helping them structure and improve their research software project management processes. Many participants mentioned that our study suggests a model with several best practices to help organizations or principal investigators assess their standing within the community, research software management, and other areas. Additionally, it helps to determine the next steps needed to improve

their project capabilities. We received equal positive, neutral, and negative responses for the *effectiveness* of RSMM.

During the expert confirmation step (second phase of evaluation), we wanted to check whether the discussion points raised during the interviews are consistent with the model we developed. We received five positive responses from our experts. Three acknowledged some deviations in the final model, RSMM v1.0, due to the aggregation of different experts' perspectives, and the outcome reflects the discussions. One of the participants (IV6) responded as follows: *"According to my notes, the order of the indicators for the respective levels does not correspond exactly to how I had lined it up on the Miro Board. However, that is not to be expected when the input from different interviews is combined."*.

The 12 case study participants expressed mixed opinions regarding the *operational feasibility*. Six participants viewed the model as a reference or checklist for future use. Additionally, they acknowledged the RSMM as a tool for improving the maturity of research software projects in project management. However, some participants indicated that they were implementing many of the practices recommended by RSMM, which limited its immediate applicability in their current workflows. Two participants highlighted that their primary focus was resolving immediate project issues rather than prioritizing maturity assessment using the model. After seeing an evaluation of their projects using RSMM, some participants wanted to add educational materials to their projects and were inspired to add a code of conduct. Others also shared their aim of improving sustainability. Feedback suggests that RSMM is helpful to researchers and RSEs who want to measure their project maturity. It is also beneficial to benchmark their projects with others; in fact, one respondent noted that they were already benchmarking their project against others as a means of improvement. Regarding *operational feasibility*, five experts expressed positive opinions. Five experts indicated they would consider using RSMM as a reference, guideline, or benchmark tool in future projects, suggesting the model could support project management improvement efforts. Additionally, six experts found the model easy to use and suitable for research software-producing organizations.

We assessed the RSMM process with 20 participants across 15 projects. Completing the evaluation took 20 min to 2 h, and the manual extraction of evidence for 79 practices proved demanding. Some practices were misunderstood, requiring additional clarification—consistent with later case study observations. Despite these challenges, approximately 70% of preliminary assessments aligned with case study participants' responses, suggesting reasonable reliability.

Regarding coverage, existing models lacked some information requirements specific to research software. Applying RSMM across 20 projects from different domains demonstrated its potential generalizability. These findings show that the evaluation is feasible within a reasonable timeframe, though participants sometimes required clarification. The 70% alignment with case study responses indicates reliable assessment, and the successful application across these projects provides initial evidence that the model can be applied to diverse research software projects.

RSMM can serve as a guide for developers and institutions, given its applicability across diverse projects and its utility for tracking progress over time. Both groups suggested automating the evaluation process to make it more accessible.

8. Discussion and limitations

The present section discusses the significance of RSMM, explores the challenges associated with adopting best practices of RSMM, and outlines the study's limitations.

8.1. Adoption of best practices

The case study findings indicate that, among the 20 projects analyzed, 77 practices were implemented by at least one project, and 33 practices were adopted by more than 75% of the projects. Over 50% of the projects implemented 51 practices, demonstrating that 65% of the model's practices have been broadly adopted. This result indicates that RSMM captures widely applicable and relevant best practices for the management of research software projects. Additionally, the findings indicate that RSMM can be applied to research software projects across various domains, as the selected case study projects represent a range of research disciplines and diverse organizational structures. However, despite the comprehensive and diverse nature of the RSMM, effective implementation may require additional resources, as highlighted by a few of the case study participants. Some practices may also require repositioning within the maturity matrix to better align with evolving challenges in research software development. We plan to implement mechanisms for ongoing assessment, including community feedback, surveys, and longitudinal studies, to support the iterative refinement of the model.

Beyond its application at the project level, the RSMM may also support emerging policy initiatives aimed at improving the sustainability and quality of research software. National research funders, research infrastructures, and initiatives such as the European Open Science Cloud (EOSC) increasingly emphasize the importance of sustainable research software practices, including reproducibility, documentation, long-term maintenance, and community engagement [1]. RSMM provides a structured framework that can help institutions and funding agencies operationalize these expectations by offering a transparent way to assess and improve research software management capabilities. For example, research institutes could use RSMM to benchmark internal practices across projects, while funders might incorporate selected maturity indicators when evaluating software sustainability plans in grant proposals. In this sense, RSMM not only supports individual project teams but may also contribute to broader efforts to professionalize research software engineering and strengthen the sustainability of software in the scientific ecosystem [2,51].

8.2. Challenges

Some of the practices of RSMM were categorized as "not applicable" or "not relevant" to many projects, as identified by interview experts and case study participants. For example, security-related practices were noted as more relevant to specific projects and not applicable to all projects. We argue that software-related practices are required for all research software projects. Any software we use could be vulnerable to an exploit or an attack. For instance, a recently discovered critical vulnerability related to the widely used Log4j library [52] illustrates how even indirect dependencies can expose a project to serious risk. Even if a developer considers the issue irrelevant to their project, such vulnerabilities can act as entry points for malicious attacks, potentially affecting other systems. Neglecting these risks may lead to a loss of trust, as researchers might hesitate to reuse insecure software. Therefore, adopting software best practices is needed to promote security and encourage broader adoption in the research community. Additionally, practices such as *'Define end-of-life policy'*, *'Set maximum response time for pull requests'*, *'Document the cost of running the application'*, and *'Generate SBOM'* were implemented by only one or two projects. Participants mentioned resource constraints, such as limited time, funding, and the absence of full-time staff, as primary barriers to adopting a few of these highly matured practices. Similarly, we observed that the practice *'Define end-of-life policy'* was considered important by case study participants. They were planning to adopt this practice. However, they highlighted the need for guidance on implementing this practice. One of the practices, crash reporting, was seen as valuable but challenging to implement due to the lack of free infrastructure and funding. However, crash reporting also contributes to code quality, so adopting manual crash reporting methods for smaller projects, such as user-submitted emails or GitHub issues, may be enough to collect and address errors. The challenges arise as these manual approaches become cumbersome

and inefficient as projects scale. In contrast, automated crash reporting systems offer more scalable solutions, with both free and paid versions.

Within the capability *'Community'*, participants highlighted challenges such as the difficulty of community engagement for independent developers. One participant noted, *"Community engagement is difficult to do as a freelancer."* We acknowledge this challenge, mainly faced by young researchers and RSEs, who find it difficult to build networks and establish credibility early in their careers. Participants also emphasized the need for national-level recognition of the importance of research software and support from institutional bodies, which includes recognizing the contributions of research software developers [53]. Another participant observed: *"Funding and recognition and reward. This is on the national agenda, but little is practically changing."*

8.3. Limitations

RSMM can act as a guide for RSEs, researchers, and users by outlining practices to improve the maturity of research software projects in different focus areas of research software project management. However, it does not provide specific instructions or suggest methodologies for implementing these practices. This lack of guidance may hinder users, particularly junior researchers or early-stage Ph.D. students, who often have limited experience with software development and project management processes.

In addition to guidance limitations, smaller projects, such as those initiated as part of a PhD, often lack the support, time, and funding necessary to implement all recommended practices. Lower-level practices of RSMM are more accessible and feasible for smaller projects, while higher-level practices are often challenging to implement due to resource limitations. Furthermore, the manual evaluation of 79 practices can be time-consuming, especially for users wishing to assess multiple projects simultaneously.

Beyond resource constraints, some practices in RSMM, such as ''Obtain support from a national research software center'' (2.2.3) and ''Acquire research software center acknowledgment'' (2.3.4), rely on infrastructure that is not globally available. This creates challenges for researchers in regions without such centers, potentially limiting their ability to progress through the maturity levels. Additionally, feedback from case study participants indicated that certain practices, such as providing front-page chat support (3.2.8), were often considered non-essential. Their inclusion and maturity level placement may not reflect the priorities or resource constraints of many projects. In future work, we will refine the placement of these practices based on recurring patterns and correlations identified through additional case studies across projects from diverse research domains.

Regarding generalizability, interview experts and case study participants raised concerns about the applicability of RSMM across diverse research software projects and disciplines. While this is a valid consideration, one of the benefits of the focus area maturity model is its customisability. Users can tailor RSMM to their specific needs by removing capabilities or practices that may not be relevant to their projects during the evaluation process. This flexibility improves usability across various project contexts and allows stakeholders to focus on practices most aligned with their goals and resources. An avenue for future research is the creation of cross-disciplinary models, as research software increasingly spans multiple fields. Integrating domain-specific practices into RSMM could better address unique requirements, such as handling sensitive data in medical research or maintaining real-time reliability in environmental simulations.

From a methodological perspective, this study has several limitations. The interviews involved 12 experts primarily from the Netherlands and Germany, which provided valuable insights but represent a geographically limited sample that may not fully represent the broader international RSE community. However, the work is also supported by an SLR of 50 papers, which grounds the model in established research and addresses widely recognized challenges within research software engineering. While this systematic approach provides a foundation based on existing literature, establishing generalizability across all research domains, geographic regions, and project contexts requires further empirical validation in diverse settings. Additionally, the SLR process has several limitations. First, we included only open-access publications, which may have resulted in the exclusion of relevant studies available behind paywalls. Second, the search strings may not have captured all relevant terminology used in the literature, potentially omitting some studies. Third, manual extraction of practices is time-consuming and prone to researcher bias. The latter was confirmed during our interview study, where experts rediscovered practices that had been missed during this process. Finally, although we searched four widely used academic databases, relevant studies indexed in other databases may have been overlooked.

Concerning the evaluation process, the current RSMM evaluation relies on self-reporting, with project maturity assessed primarily through self-assessment and final responses provided by developers. These responses are partially validated using publicly available evidence such as project websites and GitHub repositories, and developers often provide supporting proof of their practices, which helps strengthen the reliability of the assessment. Furthermore, not all 79 practices can be manually or automatically verified, and internal practices may not be publicly documented. The final maturity levels for each practice were determined by aggregating all 12 participant responses. Although two participants expressed concerns that certain results did not fully reflect their own assessments, this type of variation is common in qualitative evaluations, and the majority of participants (eight) agreed with the outcomes and supported the applicability of the model. Nevertheless, the reliance on self-reporting represents a methodological constraint that should be addressed in future work through more robust independent assessment methods.

Finally, the feedback from expert interviews and case study participants is subjective and influenced by their individual experiences, roles, and the context of their projects. While this input offers rich qualitative insights, it introduces the potential for bias or personal preferences. For instance, discussions about security-related practices of RSMM showed varying opinions on their applicability and implementation. Future studies in other domains and with larger, more diverse samples can help strengthen and refine the proposed approach.

9. Conclusion and future work

We present the focus area maturity model RSMM for research software project management. RSMM integrates best practices from three complementary domains: software engineering practices (e.g., requirements analysis, code quality, and security), open-source development practices (including code review, public repository storage with version control, and community building), and research software engineering practices (such as making software citable and discoverable). The model supports organizations producing research software in assessing and improving their research software project management capabilities. The development of RSMM addressed the research question: *How can organizations that produce research software evaluate research software project management practices?*

Several contributions emerge from this work. A structured framework capturing the key processes and organizational capabilities involved in research software project management is introduced. The development of the model drew on insights from the literature together with input from the research software engineering community. A detailed description of the focus area maturity model and its components is provided, supported by an accompanying dataset. Researchers and RSE practitioners can use this description as guidance when understanding and implementing practices associated with each capability. The application of De Bruin's maturity model development methodology, combined with card sorting techniques for maturity level assignment, is demonstrated through multiple evaluation cycles. Case studies

further indicate that RSMM can be deployed in practice with minimal involvement from researchers, highlighting its practical applicability and ease of adoption.

Several limitations of existing models discussed in Section 2 are addressed by RSMM. With 79 practices, 17 capabilities, and 4 focus areas, the model provides a comprehensive perspective on research software project management. Such breadth introduces moderate complexity, requiring users to invest time in understanding and applying the framework, but also enables organizations to address a broader range of concerns than traditional software maturity models. Sustainability, community engagement, and usability are considered alongside more traditional software development practices. Feedback from interview experts and case study participants indicates that RSMM provides a practical mechanism for projects to assess their current maturity level, compare progress with other projects, and identify areas requiring improvement.

Future work. Future research will focus on improving the usability and adoption of RSMM through partial automation of the assessment process. For example, dashboards could automatically analyze compliance with RSMM practices using data from GitHub or other software repositories. Such tools could generate reports on code quality, documentation standards, and reproducibility; analyze project metadata and suggest improvements for FAIR compliance; and evaluate sustainability indicators such as funding continuity and community engagement. An online implementation of RSMM is currently under development and will be evaluated with developers from several research software projects.

Further research will also investigate the relationship between maturity and project sustainability, including potential correlations between maturity levels and project longevity. Examination of how research software projects evolve across maturity levels over time, as well as identification of factors that trigger transitions between levels (such as organizational changes or long-term funding), forms another direction for future work. Additional analysis of dependencies and sequencing between practices within RSMM may provide deeper insight into how improvements in one capability influence others.

Refinement of the placement of practices within RSMM will also continue based on feedback from interview and case study participants, with the goal of improving clarity and applicability across different research software contexts. Additional focus areas may emerge as the model evolves. Ethical considerations surrounding research software, including issues related to AI-driven decision-making, transparency, and privacy, represent one potential extension. Further expert consultation may also lead to the inclusion of practices related to user interface design, accessibility, and internationalization.

CRediT authorship contribution statement

Deekshitha: Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Methodology, Data curation, Conceptualization. **Rena Bakhshi:** Writing – review & editing. **Jason Maassen:** Writing – review & editing. **Carlos Martinez Ortiz:** Writing – review & editing. **Rob van Nieuwpoort:** Writing – review & editing. **Antti Knutas:** Writing – review & editing. **Slinger Jansen:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Informed consent

Before starting the interview, we obtained audio-recorded informed consent from all participants.

Ethical approval

The research proposal entitled 'Focus Area Maturity Assessment Model for Research Software Project Management' was reviewed and approved by the Ethics department of Utrecht University.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used Curie, Grammarly and OpenAI ChatGPT-4 to improve the text and for grammar checking. After using these tools, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Funding

This research work is part of the CIT project *SearchSECO*, funded by the Netherlands eScience Center, the Netherlands, with Grant ID NLESC.CIT.2021.002.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Deekshitha reports financial support was provided by Netherlands eScience Center. Deekshitha reports a relationship with Deekshitha that includes: employment. Has patent pending to. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We want to acknowledge the experts who contributed to redefining RSMM (not in any 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.

We also wish to acknowledge all who contributed to data collection from various platforms for the research software projects. In no order: Mahyar Mohammadi, Roope Luukkainen, Hatef Shamshiri, Muhamad Zakiyamani, Ashok Tripathi, Maryam Gulzar, Haoyue Chen, and Erkang Zhang.

We want to acknowledge the case study participants for their valuable contributions to the validation and evaluation of the RSMM (not in any order): Vincent van Hees, Bouwe Andela, Jean-Christophe Fillion-Robin, Erik van Sebille, Ben van Werkhoven, Gijs van den Oord, Stijn Heldens, Finn Årup Nielsen, Matteo Bachetti, John Lees, Nathan G Skene, Ajda Pretnar Žagar, Alexander Skorikov, Joris J. Snellenburg, Reimar Bauer, Michael Behrisch, Elena Ranguelova, Stefan Verhoeven, Egon L. Willighagen, Alexandre M.J.J. Bonvin, Patrik Schönfeldt, and Lourens Veen.

Data availability

Data will be made available on request.

References

[1] M. Barker, N.P.C. Hong, D.S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L.J. Castro, M. Gruenpeter, P.A. Martinez, T. Honeyman, Introducing the FAIR Principles for research software, Sci. Data 9 (1) (2022) 622, http://dx.doi.org/10.1038/s41597-022-01432-0.

[2] A.-L. Lamprecht, L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. Martin Del Pico, V. Dominguez Del Angel, S. Van De Sandt, J. Ison, P.A. Martinez, et al., Towards FAIR principles for research software, Data Sci. 3 (1) (2020) 37–59.

[3] J. Carver, N. Weber, K. Ram, S. Gesing, D. Katz, A survey of the state of the practice for research software in the United States, PeerJ Comput. Sci. 8 (2022) e963, http://dx.doi.org/10.7717/peerj-cs.963.

[4] S.L. Ahrenkrog, W. Haeck, F. Abrams, D. Whelbourn, PMI's organizational project management maturity model, in: Proc. PMI® Global Congress 2003—North America, Newtown Square, PA: Project Management Institute, Baltimore, MD, 2003.

[5] CMMI Product Team, Capability maturity model® integration (CMMI SM), version 1.1, in: CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1), 2, 2002.

[6] I.A. Cosden, K. McHenry, D.S. Katz, Research software engineers: Career entry points and training gaps, Comput. Sci. Eng. 24 (6) (2022) 14–21, http://dx.doi.org/10.1109/mcse.2023.3258630.

[7] Deekshitha, S. Farshidi, J. Maassen, R. Bakhshi, R. Van Nieuwpoort, S. Jansen, FAIRSECO: An extensible framework for impact measurement of research software, in: 2023 IEEE 19th International Conference on E-Science (E-Science), 2023, pp. 1–10, http://dx.doi.org/10.1109/e-Science58273.2023.10254664.

[8] M. Gruenpeter, N. Chue Hong, S. Granger, E. Breitmoser, M. Priddy, M. Antonioletti, P.A. Martinez, T. Honeyman, J.A. Collins, R. Meneses, Research Software Workshop: Guidelines and Metrics for Metadata Curation, 2023, http://dx.doi.org/10.5281/zenodo.8075097.

[9] A.A. Younis, Y. Hu, R. Abdunabi, Analyzing software supply chain security risks in industrial control system protocols: An openssf scorecard approach, in: 2023 10th International Conference on Dependable Systems and their Applications, DSA, IEEE, 2023, pp. 302–311.

[10] F. Goth, J.P. Thiele, Foundational competencies and specializations of a research software engineer, Comput. Sci. Eng. (2025) http://dx.doi.org/10.1109/MCSE.2025.3552156.

[11] I. Puebla, G.A. Ascoli, J. Blume, J. Chodacki, J. Finnell, D.N. Kennedy, B. Mair, M.E. Martone, J. Wittenberg, J.-B. Poline, Ten simple rules for recognizing data and software contributions in hiring, promotion, and tenure, PLoS Comput. Biol. 20 (8) (2024) 1–8, http://dx.doi.org/10.1371/journal.pcbi.1012296.

[12] I.A. Cosden, K. McHenry, D.S. Katz, Research software engineers: Career entry points and training gaps, Comput. Sci. Eng. 24 (6) (2022) 14–21, http://dx.doi.org/10.1109/MCSE.2023.3258630.

[13] C. Strasser, K. Hertweck, J. Greenberg, D. Taraborelli, E. Vu, Ten simple rules for funding scientific open source software, PLoS Comput. Biol. 18 (11) (2022) e1010627.

[14] J.C. Carver, I.A. Cosden, C. Hill, S. Gesing, D.S. Katz, Sustaining research software via research software engineers and professional associations, in: 2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability, BoKSS, IEEE, 2021, pp. 23–24.

[15] D.S. Katz, S. Druskat, R. Haines, C. Jay, A. Struck, The state of sustainable research software: Results from the workshop on sustainable software for science: Practice and experiences (wssspe5. 1), 2018, arXiv preprint arXiv:1807.07387.

[16] A. Murray, M. Ward, Improving Project Performance Using the PRINCE2 Maturity Model (P2MM), The Stationery Office, 2007.

[17] S. Jansen, A focus area maturity model for software ecosystem governance, IST 118 (2020) 106219.

[18] M. Overeem, M. Mathijssen, S. Jansen, API-m-FAMM: A focus area maturity model for API Management, IST 147 (2022) 106890.

[19] E. van der Velden, CMasher: Scientific colormaps for making accessible, informative and 'cmashing' plots, J. Open Source Softw. 5 (46) (2020) 2004, http://dx.doi.org/10.21105/joss.02004, arXiv:2003.01069.

[20] J. Caffaro, S. Kaplun, Invenio: A Modern Digital Library for Grey Literature, Technical Report CERN-OPEN-2010-027, CERN, 2010.

[21] C. Martinez-Ortiz, R. Bakhshi, Y. Dzigan, N. Renaud, F. Diblen, B. Weel, M. van Meersbergen, N. Drost, S. van der Burg, F. Alidoost, Structured and unstructured teams for research software development at The Netherlands eScience center, Comput. Sci. Eng. 24 (3) (2022) 25–32, http://dx.doi.org/10.1109/MCSE.2022.3167448.

[22] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, C.C. Venters, Sustainability design and software: The karlskrona manifesto, in: Proceedings of the 37th International Conference on Software Engineering, ICSE, vol. 2, IEEE, 2015, pp. 467–476, http://dx.doi.org/10.1109/ICSE.2015.179.

[23] Deekshitha, R. Bakhshi, J. Maassen, C. Martinez-Ortiz, R. van Nieuwpoort, S. Jansen, Research Software Focus Area maturity model (RSMM) dataset, 2025, http://dx.doi.org/10.5281/zenodo.14827462, Data set.

[24] T. Mettler, P. Rohner, R. Winter, Towards a classification of maturity models in information systems, in: A. D'Atri, M. De Marco, A.M. Braccini, F. Cabiddu (Eds.), Management of the Interconnected World, Physica-Verlag HD, Heidelberg, 2010, pp. 333–340.

[25] J. Poeppelbuss, B. Niehaves, A. Simons, J. Becker, Maturity models in information systems research: literature search and analysis, Commun. Assoc. Inf. Syst. 29 (1) (2011) 27, http://dx.doi.org/10.17705/1CAIS.02927.

[26] W. Pullen, A public sector HPT maturity model, Perform. Improv. 46 (4) (2007) 9–15, http://dx.doi.org/10.1002/pfi.119, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/pfi.119.

[27] T. De Bruin, M. Rosemann, R. Freeze, U. Kaulkarni, Understanding the main phases of developing a maturity assessment model, in: Proc. Australasian Conf. on Information Systems, ACIS, AAIS, 2005, pp. 8–19.

[28] M. van Steenbergen, R. Bos, S. Brinkkemper, I. van de Weerd, W. Bekkers, The Design of Focus Area Maturity Models, in: Proc. Conf. Global Perspectives on Design Science Research (DESRIST 2010), Springer, Springer Berlin Heidelberg, 2010, pp. 317–332.

[29] J. Pöppelbuß, M. Röglinger, What makes a useful maturity model? a framework of general design principles for maturity models and its demonstration in business process management, in: Proc. European Conf. on Information Systems, ECIS, 2011, p. 28, URL: http://aisel.aisnet.org/ecis2011/28.

[30] C. Okoli, A guide to conducting a standalone systematic literature review, Commun. Assoc. Inf. Syst. 37 (2015).

[31] Deekshitha, Research Software focus area Maturity Model: Method, 2026, http://dx.doi.org/10.5281/zenodo.19412427, (Accessed 4 April 2026).

[32] J. Nielsen, Card Sorting to Discover the Users' Model of the Information Space, Nielsen Norman Group, 1995.

[33] 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, H. Zach, **Research Software Directory**, 2020, http://dx.doi.org/10.5281/zenodo.1154130, version: 3.0.1, URL: https://github.com/research-software-directory/research-software-directory.

[34] R. van Nieuwpoort, D.S. Katz, Defining the roles of research software, Upstream (2023) http://dx.doi.org/10.54900/9akm9y5-5ject5y.

[35] V. van Hees, Z. Fang, E. Mirkes, J. Heywood, J.H. Zhao, C.P. Joan, S. Sabia, J.H. Migueles, **GGIR**, 2022, http://dx.doi.org/10.5281/zenodo.7043054, version 2.7-6.

[36] B. Andela, B. Broetz, L. de Mora, N. Drost, V. Eyring, N. Koldunov, A. Lauer, B. Mueller, et al., **ESMValTool**, 2023, http://dx.doi.org/10.5281/zenodo.3401363, URL: https://github.com/ESMValGroup/ESMValTool/.

[37] N. Prat, I. Comyn-Wattiau, J. Akoka, A taxonomy of evaluation methods for information systems artifacts, J. Manage. Inf. Syst. 32 (3) (2015) 229–267, http://dx.doi.org/10.1080/07421222.2015.1099390.

[38] J. Jurison, Software project management: the manager's view, Commun. Assoc. Inf. Syst. 2 (1) (1999) 17.

[39] C. Goble, Better software, better research, IEEE Internet Comput. 18 (5) (2014) 4–8, http://dx.doi.org/10.1109/MIC.2014.88.

[40] C. Martinez-Ortiz, P. Martinez Lavanchy, L. Sesink, B.G. Olivier, J. Meakin, M. de Jong, M. Cruz, Practical guide to software management plans, 2023, http://dx.doi.org/10.5281/zenodo.7589725, URL: https://doi.org/10.5281/zenodo.7589725.

[41] C.C. Venters, R. Capilla, E.Y. Nakagawa, S. Betz, B. Penzenstadler, T. Crick, I. Brooks, Sustainable software engineering: Reflections on advances in research and practice, Inf. Softw. Technol. 164 (2023) 107316, http://dx.doi.org/10.1016/j.infsof.2023.107316, URL: https://www.sciencedirect.com/science/article/pii/S0950584923001714.

[42] A.M. Thakur, R. Milewicz, M. Jahanshahi, L. Paganini, B. Vasilescu, A. Mockus, Scientific open-source software is less likely to become abandoned than one might think! Lessons from curating a catalog of maintained scientific software, Proc. ACM Softw. Eng. 2 (FSE) (2025) 2216–2239, http://dx.doi.org/10.1145/3729369.

[43] R.C. Seacord, J. Elm, W. Goethert, G.A. Lewis, D. Plakosh, J. Robert, L. Wrage, M. Lindvall, Measuring software sustainability, in: Proceedings of the International Conference on Software Maintenance, ICSM, IEEE, 2003, pp. 450–459, http://dx.doi.org/10.1109/ICSM.2003.1235455.

[44] D.S. Katz, L.C. McInnes, D.E. Bernholdt, A.C. Mayes, N.P.C. Hong, et al., Community Organizations: Changing the Culture in Which Research Software Is Developed and Sustained, Comput. Sci. Eng. 21 (2) (2019) 8–24, http://dx.doi.org/10.1109/MCSE.2018.2883051.

[45] S. Hermann, J. Fehr, Documenting research software in engineering science, Scirep 12 (1) (2022) 6567.

[46] GitHub, Documenting your project with wikis, 2021, (Accessed 19 February 2024), URL: https://docs.github.com/en/communities/documenting-your-project-with-wikis.

[47] C. Argyris, D.A. Schön, Theory in Practice: Increasing Professional Effectiveness, Jossey-Bass, San Francisco, CA, 1974.

[48] M. Niazi, D. Wilson, D. Zowghi, A maturity model for the implementation of software process improvement: an empirical study, J. Syst. Softw. 74 (2) (2005) 155–172.

[49] M.A. Heroux, E. Gonsiorowski, R. Gupta, R. Milewicz, J.D. Moulton, G.R. Watson, J. Willenbring, R.J. Zamora, E.M. Raybourn, Lightweight software process improvement using productivity and sustainability improvement planning (PSIP), in: Annual Workshop on HPC User Support Tools, Springer, 2019, pp. 98–110.

[50] V. Clarke, V. Braun, Thematic analysis, J. Posit. Psychol. 12 (3) (2017) 297–298.

[51] D.S. Katz, N.P.C. Hong, T. Clark, A. Muench, S. Stall, D. Bouquin, M. Cannon, S. Edmunds, T. Faez, P. Feeney, et al., Recognizing the value of software: a software citation guide, F1000Research 9 (2020).

[52] S. Riverman, N. Burgan-Illig, XZ Utils, the xz Backdoor and What We Can Learn from Open Source CVEs, 2024, (Accessed 09 January 2025), URL: https://www.puppet.com/blog/xz-backdoor.

[53] T. Vergoulis, M. Makaronidou, S. Amodeo, T. Stavropoulos, F. Quaglia, K.E.A. Bouhraoua, S. Roiser, D. Garijo, D. Piovesan, F. Psomopoulos, N. Pechlivanis, A. Orfanou, E. Bakoglidou, D5.1 Landscape analysis of existing rewards and mechanisms for research software and training activities. , 2025, http://dx.doi.org/10.5281/zenodo.14978474.