Research Software Stories

A Guide for Researchers and Research Software Engineers

EVERSE Network Training – November 2025

NOTE This brief guide is written to accompany the seminar. It is primarily an AI generated summary of the transcript of the key points, with light editting afterwards. This guide is provided "AS IS" and while it's been checked for correctness, minor errors may still be present.

Michael Sparks, michael.sparks@manchester.ac.uk

Contents

1. Introduction and Context
2. What Are Research Software Stories?
3. Why Write Them? (Audiences and Benefits)
4. Template Walkthrough
5. Guided Writing Session
6. Conclusion

1. Introduction and Context

Software underpins an increasing proportion of scientific research. Yet essential knowledge about why specific tools exist, how they are used, and what decisions have shaped them often goes undocumented. This knowledge frequently lives in people's memory, informal notes, or dispersed communications.

Research Software Stories provide a structured way to capture this context. They offer a concise narrative describing:

- the purpose of a software project
- the problem it addresses
- the community that builds and uses it
- the practices and tools that support it
- the scientific impact it enables

Unlike technical documentation, a Research Software Story explains *why* the software matters and *how it fits within the wider research landscape*.

A Research Software Story can:

- make the research impact of the software visible
- support onboarding for researchers, students, and collaborators
- highlight existing good practices and areas for improvement
- provide material for sustainability planning and proposals
- assist funders and evaluators in understanding value and context

This guide introduces the concept, describes the motivation, and explains how to write a Research Software Story using the template developed within the **Research Software Quality Toolkit (RSQKit)**.

2. What Are Research Software Stories?

A Research Software Story is a short, structured narrative that describes a software project in its research context. It is not a technical manual. Instead, it captures the essential information that helps researchers and developers understand:

- **What** the software does - the research it supports
- **Why** it exists
- **Who** uses and maintains it
- **How** it is developed and supported
- **Where** it fits within research workflows
- **Which** practices, tools, and decisions shape it

2.1 Purpose

Research Software Stories help surface implicit knowledge — the reasoning, expectations, and assumptions often missing from formal documentation. They provide clarity not only for software use, but for scientific reproducibility and project continuity.

2.2 Why they are different from documentation

- Documentation explains *how* to run or extend the code.
- A Research Software Story explains *why* the software matters and where it sits in the overall research ecosystem.

2.3 Who benefits

Research Software Stories can support:

- new researchers joining a project
- early career researchers inheriting workflows
- RSEs asked to maintain or extend software
- managers overseeing planning and sustainability
- funders assessing impact and risk
- collaborators evaluating whether the tool fits their needs

They help create shared understanding, reduce onboarding time, and clarify scientific value.

3. Why Write Research Software Stories?

(Audiences and Benefits)

Different groups rely on research software in different ways. Understanding these audiences helps explain why writing a Research Software Story can be useful.

3.1 Researchers

Researchers depend on software for core scientific tasks, but may lack clarity about:

- why certain design choices were made
- what limitations or assumptions exist
- how workflows evolved
- where risks or dependencies lie

A Research Software Story can:

- provide context needed for reproducibility
- clarify assumptions and boundaries
- support grant writing and reporting
- make scientific motivation explicit

It can strengthen research by ensuring shared understanding of key tools.

3.2 Early Career Researchers

ECRs frequently join established projects without full context. A Research Software Story:

- offers a clear starting point
- explains typical user workflows
- reduces onboarding time
- provides confidence in navigating software and data

This supports skill development and encourages meaningful contribution.

3.3 Research Software Engineers

RSEs often need to rapidly understand unfamiliar projects. A Research Software Story can:

- summarise the problem, scope, and purpose
- reveal dependencies and complexity
- highlight areas requiring improvement
- describe current practices and workflows
- reduce time spent reverse-engineering context

This can support more effective, targeted engineering work.

3.4 Managers and Coordinators

Managers must plan around sustainability, staffing, and risk. A Research Software Story:

- provides a clear overview of project value
- highlights gaps or single-person dependencies
- supports decisions about investment and prioritisation
- improves communication with stakeholders

It can support better strategic planning across research groups.

3.5 Funders and Evaluators

Funders increasingly assess software as part of research output. A Research Software Story can:

- demonstrate scientific relevance
- show evidence of community adoption
- outline sustainability and governance
- identify risks early
- support case building in proposals

This can strengthen competitiveness and clarity.

4. Template Walkthrough

The Research Software Story template provides a structured framework for capturing context. Each section addresses a specific aspect of the project.

4.1 Problem

Describe the scientific or practical problem the software addresses.

Explain:

- what bottleneck or issue motivated its creation
- why existing approaches were insufficient
- how this problem relates to the wider research aim

This section sets the foundation for the rest of the story.

4.2 User Community

Describe who builds and uses the software.

Include:

- developers and maintainers
- researchers and domain experts
- typical users or workflows
- skill levels and expertise

This reveals the social and organisational structure around the software.

4.3 Technical Aspects

Summarise the technical nature of the software:

- type of tool
- languages and frameworks

- architecture or design notes
- scale or complexity
- relevant constraints

Provide enough context for contributors to understand the system.

4.4 Libraries and Systems

List key dependencies, scientific libraries, workflow engines, and external systems.

This helps newcomers understand compatibility requirements and domain expectations.

4.5 Software Practices

Describe the practices that shape development:

- version control
- code review
- testing
- release processes
- communication mechanisms

This section helps identify strengths and potential improvements.

4.6 Developer Community: Onboarding and Adoption

Explain how new users and developers begin.

Include:

- typical user journeys
- tutorials and documentation
- learning pathways
- common early obstacles

This can support easier onboarding and reveal maturity.

4.7 Tools

List tools that support development, testing, documentation, or packaging.

This describes the infrastructure used to maintain quality.

4.8 FAIR & Open

Describe how the software aligns with FAIR principles and open-development practices.

Research Software Stories **can enable** better transparency and visibility by clarifying what is already open and what could be strengthened.

4.9 Documentation

Describe what documentation exists, where it lives, and how it is used.

Honest reflection is encouraged — this section supports later improvement.

4.10 Sustainability

Explain governance, maintenance roles, activity levels, and funding.

This identifies long-term plans and potential risks.

4.11 References

List relevant work, documentation, standards, or scientific context.

This situates the software in its ecosystem.

5. Guided Writing Session

This final section supports participants in drafting their own Research Software Story.

For each section:

1. Begin with **two or three full sentences**.
2. Add bullet points to expand detail.
3. Focus on clarity, not perfection.
4. Capture *what is true now*, not an aspirational future.
5. Move steadily through the template without overthinking wording.

This session aims to produce a complete first draft that can be refined later. It can also help identify missing documentation, coordination challenges, or sustainability concerns — insights valuable for both project teams and wider RSQKit engagement.

6. Conclusion

Research Software Stories provide a practical way to make research software context visible. They help researchers, RSEs, and collaborators understand the purpose, value, and structure of a project. By capturing community, practice, tools, and sustainability, they support:

- reproducibility
- clarity
- effective onboarding
- long-term planning
- stronger proposals
- improved communication
- alignment with RSQKit tasks

A Research Software Story does not need to be perfect. It needs to be *honest*, *useful*, and *maintained*. Writing one is an investment in the future of your software and the research it supports.