Strengthen your Research by Telling Its Software Story

1 - Introduction & Context

EVERSE Network Training, Michael Sparks (Nov 2025)

Welcome

- Thank you for joining us today.
- Today we explore how telling your software's story supports your research.

- Support Materials https://tinyurl.com/EVERSE202511-RSS
- https://github.com/sparkslabs/EVERSE-Seminar-Nov2025-ResearchSoftwareStories

Good afternoon, and thank you for joining us today. My name is Michael Sparks, and over the next two hours we're going to explore something that sounds deceptively simple, but which turns out it can have a surprisingly large impact on how effectively we conduct, communicate, and sustain our research - that is, telling the story of our research software.

Agenda

First Part

- ***1 Introduction and context*** **<– We are here**
- 2 What are Research Software Stories?
- 3 Why write them + Q&A
- 4 How we write them - Template walkthrough

Second Part

- Guided Writing Session
- Discussion, Q&A
- Support Materials https://tinyurl.com/EVERSE202511-RSS

So, first the agenda!

I'll come back to this, but there's five mini sessions today:

- Introduction (10 minutes)
- What are What are Research Software Stories (10minutes)
- Why write them + Q&A (20 minutes)
- How we write them - Template walkthrough (15 minutes) (short break)
- Guided writing session (35 mins + 15 mins)

Section 1 - Introduction & Context

Housekeeping

- Session is recorded for later viewing.
- Cameras: on/off as you prefer.
- Microphones: please mute most of the time
- Feel to use the chat
  - to say where you're from etc
  - What you're hoping from this session
- Please put **questions in the Q&A**

Before we get into anything substantial, let me start with a few introductory notes.

This session is being recorded so that participants who cannot attend live can revisit the material later. You are welcome to keep cameras on or off - whatever is most comfortable for you. I'll ask you to mute your microphone, unless we're in a Q&A session. Please DO use the chat - if you can pop where you're from into the chat that would be interesting, and what you're hoping to get from from this session - that would be great. Please DO pop questions into the Q&A, and I'll review the Q&A periodically.

Structure

- Two-hour session: introduction + guided writing.

The session will have two parts: the first is presentation-focused, where I'll introduce the core ideas, and the second hour is hands-on, where you'll begin drafting the outline of a Research Software Story for your own project.

Who am I?

So, who am I?

Well, I'm a Senior Research Software Engineer at the University of Manchester. I'm based in Physics and Astronomy, and work as part of the eScience lab team on the EVERSE project. My background is computer science, and I've been writing code for longer than I can remember, and a professional software engineer for over 27 years – over 23 of those in a R&D context.

This has been from broadcast studio projects, and network systems through concurrency component systems, mass participatory projects through to building the original BBC micro:bit prototype which we trialled in schools to enable children to learn the core of software development.

EVERSE & RSQKit

- Joined EVERSE & University of Manchester earlier this year
- EVERSE'S core goal is to improve the quality of research across the EU science clusters by increasing the quality of the research software that underpins that research
- RSQKit is one of the means to that end

I've been with EVERSE project and the University of Manchester since earlier in the year.

But what is the EVERSE project? Many of you on the call may already know this and may have attended other EVERSE seminars beforehand, in which case Hello! Some of you may have just seen a linkedin message promoting this and thought "that sounds intriguing" and not know anything about EVERSE.

So briefly:

- EVERSE'S core goal is to improve the quality of research across the EU science clusters by increasing the quality of the research software that underpins that research.
- The EVERSE project aims to create a framework for research software and code excellence, collaboratively designed and championed by the research communities, in pursuit of building a European network of Research Software Quality and setting the foundations of a future Virtual Institute for Research Software Excellence.
- One of the work-in-progress outcomes of this project is called "RSQKit" - a Research Software Quality Tookit. This is captures and shares good practices in terms of terms metrics - such as dimensions and indicators, but also in terms of a collection of common tasks and tools, and descriptions of good software practices in the community. This allows sharing of best practices, not just in theory, but also in practice.

Why This Matters

If you care about reproducibility, onboarding, or the ability to restart your research after a gap, then a clear understanding of your software is one of the most practical tools you have.

Now, let me introduce what we're doing today and why this matters.

I want to begin with a very simple claim: if you care about reproducibility, onboarding, or being able to restart your research after a gap, then having a clear, structured understanding of your research software is one of the most practical things you can do to support yourself and your collaborators. This might sound obvious - almost trivial - but in practice, many of us never actually write these things down in a coherent, accessible way.

Research Software Is Embedded

- Analysis, simulation, modelling, workflows.
- Practices often remain hidden.
- Knowledge can become fragile.

Research software is often deeply embedded in the scientific process. It performs analyses, transforms data, runs simulations, generates models, or glues together entire workflows. Yet despite being essential for research outputs, software practices often remain hidden: stored in the memories of a small number of people, in scattered documentation, or in isolated code comments. And when that happens, the knowledge we rely on is always more fragile than we think.

When People Join Your Project

- New PhD students, postdocs, RSEs.
- They ask predictable, reasonable questions:
- What does this do?
- Why does it exist?
- Who relies on it?

Now, consider for a moment the situations many of us find ourselves in regularly. Perhaps you've brought a new PhD student onto your project, or a postdoc, or maybe you've asked for help from an RSE team. What happens next? Well, almost immediately - and quite understandably - they begin to ask you a set of very predictable, very reasonable questions.

Questions People Ask

- What scientific claim depends on it?
- What assumptions lie beneath the surface?
- What must they understand to contribute safely?

Questions like:

What does this software actually do? What scientific claim depends on it? What assumptions lie beneath the surface? Why does it exist? Who relies on it? What should they read or understand before they can contribute safely?

Answering These Questions Takes Time

- Answers often live in emails, issues, memories.
- "Tribal memory" makes projects fragile.
- Context easily becomes separated from code.

If you've ever been in the position of answering these questions on the fly, you'll know that it takes time. Sometimes lots of time, and the answers aren't always written down anywhere. They might be scattered across old emails, issues, notes, or just "tribal memory" within the group.

This Isn't Just About Newcomers

- Adjacent colleagues ask the same questions.
- Grant writing requires clear software context.
- Returning after months away reveals gaps.

This isn't limited to new team members. The same questions come up when you're speaking to colleagues in adjacent areas, or when you're applying for funding, or even when you're trying to remind yourself what you were doing six months ago after a period of leave or attending to other projects. Our research can move at quite a pace, and it's very easy for context to become separated from the code that implements it.

Flip the Perspective

- Imagine *you* are the one brought in to help.
- You immediately ask the same foundational questions.
- Without answers, early work becomes detective work.

So let's flip the perspective for a moment. Suppose that instead of leading a project, you're the one being brought in to help. You're the collaborator or the contractor arriving because something needs attention quickly. Maybe you're an RSE brought in to help. Perhaps the software has reached a bottleneck, or needs reviewing, or needs extending. The very first thing you do in that scenario is ask those same foundational questions - what is this? What problem is it solving? Who depends on it? How does it underpin the science?

Without clear answers, the first days or even weeks of work may become detective work rather than productive work.

Funders Think This Way Too

- Committees need clarity, credibility, relevance.
- They want to see context and community need.
- Plausibility and purpose matter.

Lastly funders - this is something we don't tend to think about - except in terms of how we interacted with them . However, the same sorts arise for them. When you apply for a grant that involves software, committees do want to know whether the software is credible, maintainable, and relevant. They want to know that you understand its context and that other people use it, rely on it, and/or benefit from it. They want to see plausibility and purpose. Without a clear narrative, it becomes much harder to communicate the value of the software: why it matters, whom it serves, and how it contributes to the research aims.

A Simple Reality

Many important conversations about research software depend on having a clear, reliable, quickly-communicable understanding of the project.

All of these examples point to a simple reality: many of the most important conversations about research software rely on having a clear understanding of the project that can be communicated quickly, reliably, and consistently. Without that, we lose time, risk misunderstandings, and make our own work harder to share and sustain.

Introducing Research Software Stories

- A short, structured narrative capturing what the software does, who it serves, how it supports research, and how it is developed.
- A way to answer the fundamental questions collaborators always ask.

This is where the idea of a Research Software Story comes in.

A Research Software Story is a short, structured narrative that captures the essentials of your project: what it does, who it serves, how it is developed, and how it supports your research. It's not a technical manual. It's not a full set of documentation. And it's not a marketing document. Instead, it's a way of answering, in one place, the questions that collaborators, funders, reviewers, and newcomers always ask - the questions that sit beneath the surface of almost every project discussion.

Why Stories Help

- Organise your own thinking.
- Reveal assumptions and gaps.
- Make implicit knowledge explicit.
- Guide sustainable improvement.

It's also a way of organising your own thinking. The process of writing such a story encourages you to reflect on the current state of the project, the assumptions it that brought you there, its sustainability, and its connection to your research objectives. It can reveal gaps or strengths you hadn't noticed. It can show you the difference between what's known implicitly within your team and what's communicated explicitly. It can guide you toward better practices without requiring you to overhaul everything all at once.

RSQKit and Stories

- Stories are part of the Research Software Quality Toolkit.
- RSQKit gathers practices across scientific domains.
- Stories make quality work visible and learnable.

Today we're going to explore how Research Software Stories are used within RSQKit. RSQKit will has started gathering examples of these and will be gathering examples of practices from across scientific domains - from physics and astronomy to life sciences, environmental modelling, digital humanities, and more. These stories are one part of that ecosystem. When shared, they help make visible the work that goes into developing good research software, and they help others learn from your experience.

Why Stories Save Time

- Faster onboarding
- Fewer repeated explanations
- Easier project restarts
- Stronger credibility in grants

In my own work as an RSE and as someone who has spent many years in research environments, I've found that when such information are written well, it can save enormous amounts of time: time onboarding people, time explaining projects, time restarting paused work, and time establishing credibility in grant proposals. They can also help create more reproducible research by making assumptions visible and surfacing unanswered questions.

What This Hour Covers

- What stories are
- Why they matter
- How they support research
- How the template structures them

So the aim of the first hour is to describe:

• what these stories are • why they matter • how they can support your work • and how having standard RSQKit template helps structure them and make them more accessible.

What the Next Hour Will Do

You will begin drafting your own story.

You don't need to finish - the goal is structured thinking and capturing essentials.

In the second hour, you'll begin drafting one for your own project. You won't need to finish it today; the aim is simply to give you the structured space to think through these questions and capture the essential points.

Not Bureaucracy - A Practical Tool

Lastly, this isn't paperwork. It's a tool to make your research easier to conduct, explain, and sustain.

- Support Materials https://tinyurl.com/EVERSE202511-RSS

Lastly, the point I want to emphasise here is that this isn't a bureaucratic exercise. It's not another administrative burden. It's a practical tool designed to make your research easier to conduct, easier to explain, and easier to sustain. It's about strengthening your work, not adding random extra to it.

And with that, let's move into the next section: what Research Software Stories actually are.