

Use Cases and the Research Software Quality Kit (RSQKit)

Michael Sparks

University of Manchester

Senior Research Software Engineer

michael.sparks@manchester.ac.uk



(Slido is coming)

tinyurl.com/WLCGRSQKit

ELIXIR-STEERS WP2 meeting

24 April 2025, Online



**Funded by
the European Union**

This project has received funding from the European Union's Horizon Europe Programme under GA 101129744 – EVERSE – HORIZON-INFRA-2023-EOSC-01-02



Use Cases:
Analysis Code
Prototype Tools
Software Infrastructure





Why?



Why?



“Improve software quality”

“Make software results dependable

Correctness, Reusable, Citable, Reproducible

-- now and in the future



Why?



“Improve software quality”

“Make software results dependable

Correctness, Reusable, Citable, Reproducible

-- now and in the future

Can someone reproduce your work, using your software, 20 years from now?



ESCAPE

Astronomy
and
Particle
Physics



The five science clusters

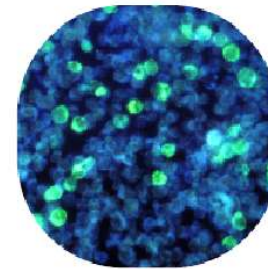
<https://science-clusters.eu>



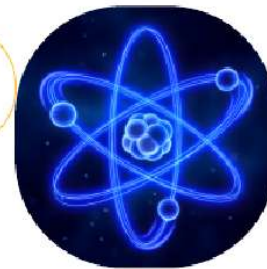
Environmental
Sciences



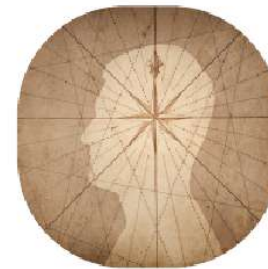
Life
Sciences



Photon and
Neutron
Sciences



Social sciences
and
Humanities





... in ESCAPE





To find & share what actually is helpful
in practice, not “in theory”

To capture useful knowledge – From **YOU** & Beyond

To share that knowledge widely

To distil common, reusable lessons

To create a positive feedback loop of improvement

I know the HEP Community already
takes this seriously...



I know the HEP Community already takes this seriously...

Affiliated Projects and Software Guidelines

In a spirit of openness and flexibility, the HSF maintains an evolving checklist of best practices for HSF Affiliated Projects and Software, rather than a set of requirements. [HSF Affiliated Projects and Software](#) need to abide by (at least a subset of) the guidelines, which are used for their endorsement and attribution of Bronze, Silver or Gold levels of recognition.

The guidelines have been created to:

- Encourage projects to follow best practices;
- Help new projects discover what those practices are; and
- Help users know which projects are following best practices, meant as a guarantee of quality.

The guidelines can be updated in light of updates or release of community practices, such as those emerging from e.g. the [EVERSE project](#).

The guidelines take inspiration from the [“old HSF page”](#) and from the [Open Source Security Foundation \(OpenSSF\)’s page](#).

Best-practice Guidelines

General guidelines

Any software library should strive to the following:

- **Availability of the code in a public repository.** The code should be accessible in anonymous read-only mode by anybody. GitHub is a very popular solution.
- **A suitable name.** It is good practice to choose a new name (a unique name is better, but often difficult) or at least a name that



I know the HEP Community already

takes this seriously, Endorsement Badge Levels

Three endorsement levels are defined to distinguish mainly the level of maturity, developer support, community support and engagement: Bronze, Silver and Gold. For each level a number of requisites need to be demonstrated. Each level adds more requisites to the previous level. These requisites lay in three major categories:

- Software engineering practices followed by the project (as described in the Best-practice Guideline section and with specifics for the programming language ecosystem).
- Sustainability and support structures of the project (e.g. number of active developers, discussion fora, documentation, training events, time to respond to issues, etc.)
- Level of adoption of the software by experiments and other projects, hence the impact of the project.

The keywords *MUST*, *SHOULD*, *MAY* that appear in the criteria in this document are to be interpreted as described in [RFC 2119](#):

- The term *MUST* is an absolute requirement, and *MUST NOT* is an absolute prohibition.
- The term *SHOULD* indicates a criterion that is normally required, but there may exist valid reasons in particular circumstances to ignore it.
- The term *MAY* provides one way something can be done, e.g., to make it clear that the described implementation may differ and be acceptable.

The purpose of this entry level is for a young endeavour, likely evolving from and within a collaboration, with the potential for other communities or experiments to use. At this level the category of best software practices is what is required.

Basics

- The project SHOULD follow general and language-specific best practices as given for example in the HSF's Best Practice Guidelines.
- The project source code MUST reside on a version-controlled repository that is publicly readable and has a URL (e.g., GitHub, GitLab).
- The README file at the top level MUST describe what the software does (what problem does it solve?).
- The project website or repository MUST provide information on how to: obtain, provide feedback (as bug reports or enhancements), and contribute to the software.
- The contribution process MUST be explained (e.g., are pull requests used?).
- The software MUST be released on an Open Software license.
- The project MUST post the license(s) of its results in a standard location in their source repository.
- Support structures of the project (e.g. number of active developers, discussion fora, documentation, training events, time to respond to issues, etc.)
- Level of adoption of the software by experiments and other projects, hence the impact of the project.

The keywords *MUST*, *SHOULD*, *MAY* that appear in the criteria in this document are to be interpreted as described in [RFC 2119](#):

- The term **MUST** is an absolute requirement, and **MUST NOT** is an absolute prohibition.
- The term **SHOULD** indicates a criterion that is normally required, but there may exist valid reasons in particular circumstances to ignore it.
- The term **MAY** provides one way something can be done, e.g., to make it clear that the described implementation may differ and be acceptable.

The purpose of this entry level is for a young endeavour, likely evolving from and within a collaborative environment, with the potential for other communities or experiments to use. At this level the category of best software practices is what is required.

Silver

Are for projects aiming for Gold but in an earlier phase towards strong community support and adoption (e.g., adoption is still relatively shy, maintenance is not secured at least in the medium term by more than a single person). High standards of software engineering should be met. All the criteria for Bronze **MUST** be fulfilled, with the addition of the following criteria.

Basics

- The project **MUST** follow general and language-specific best practices as given for example in the HSF's Best Practice Guidelines.

Documentation

- The project **MUST** provide different kinds of documentation:
 - Basic documentation.
 - User documentation.
 - Reference manual.
 - Tutorials (e.g. notebooks).
 - The project **SHOULD** provide specific training for users to use the software product.
- The term **MUST** is an absolute requirement, and **MUST NOT** is an absolute prohibition.
- The term **SHOULD** indicates a criterion that is normally required, but there may exist valid reasons in particular circumstances to ignore it.
- The term **MAY** provides one way something can be done, e.g., to make it clear that the described implementation may differ and be acceptable.

level is for a young endeavour, likely evolving from and within a collaborative effort. It is a level of best practices or experiments to use. At this level the category of best software practices is what

Gold

Project-specific best practices as given for example in the HSF's Best Practice

HSF endorsement level for projects that are adopted by several collaborations and/or experiments with a strong and long-term community support model. All the criteria for Silver MUST be fulfilled, with the addition of the following criteria.

Documentation

- The project MUST provide specific training for users to use the software product or tool on a frequency of once a year.
- The project SHOULD organise user workshops or engage in community events as a means to collect feedback from the user community.

Sustainability

- The project MUST be actively maintained and produce at least one new release a year if there are changes, and produce patch releases as relevant to include fixes that have been accumulated.
- The project MUST have at least 3 long-term maintainers with a future commitment to the software of at least 2 years.
- The project MUST acknowledge a majority of bug reports submitted in the last 1-4 weeks (inclusive); the response need not include a fix.
- The project MUST respond to a majority (>50%) of enhancement requests in the last 1-4 weeks (inclusive).
 - The r...

- The term MUST is an absolute requirement.
- The term SHOULD indicates a criterion that should be followed, but it is acceptable to ignore it.
- The term MAY provides one way something can be done, e.g., it is recommended and be acceptable.

I know the HEP Community already
takes this seriously...

... so RSQKit is learning from there too...





To find & share what actually is helpful in practice, not “in theory”

To capture u
To
To d
To create a

That includes in
this talk...
Slido is coming

YOU & Beyond
ely
sons
Improvement



“Analysis Code” use case

Event Data analysis helper code for analysing
Atlas Runs of proton smashing physics
Aims to make it easier to perform these analyses

What practices work well? What pain points?
Works For Me vs Minimum Viable Reproducibility



“Prototype Tool” use case

ML tool for managing very big data sets
Used by Early Career Researchers

What practices work well? What pain points?

Good as a teaching tool, Potential for software infrastructure. Potential good practice “client”
Dependencies version lag? Onboarding?



“Software Infrastructure” use case

Experiment-independent toolkit for (charged) particle track reconstruction in (high energy) physics experiments

What practices work well? What pain points?

Good exemplar of many processes.

Reproducibility in the long term?

First Question for you

Where do you see your codebase sitting?

Analysis? Prototype tools? Infrastructure? Other?



<https://tinyurl.com/WLCGRSQKit>

- The Research Software Quality Kit:
- Task driven guide – how to improve you research software quality (RSQ)
- Tools related to improving RSQ
- Training related to improving RSQ
- Exemplars of good practice – Research Software Stories (WIP)
- Roles involved in research software
- Research Infrastructure and clusters with heavy use of Research Software
- A model for understanding the research software process for different types of research software

{RSQ}Kit

About Get involved Contact us Bluesky LinkedIn GitHub Search...

Research Software Quality Toolkit for Sciences

Research Software Quality Toolkit (RSQKit) lists curated best practices, tools and resources for improving the quality of research software

What can we help you find?

Search RSQKit ...

Browse all topics by

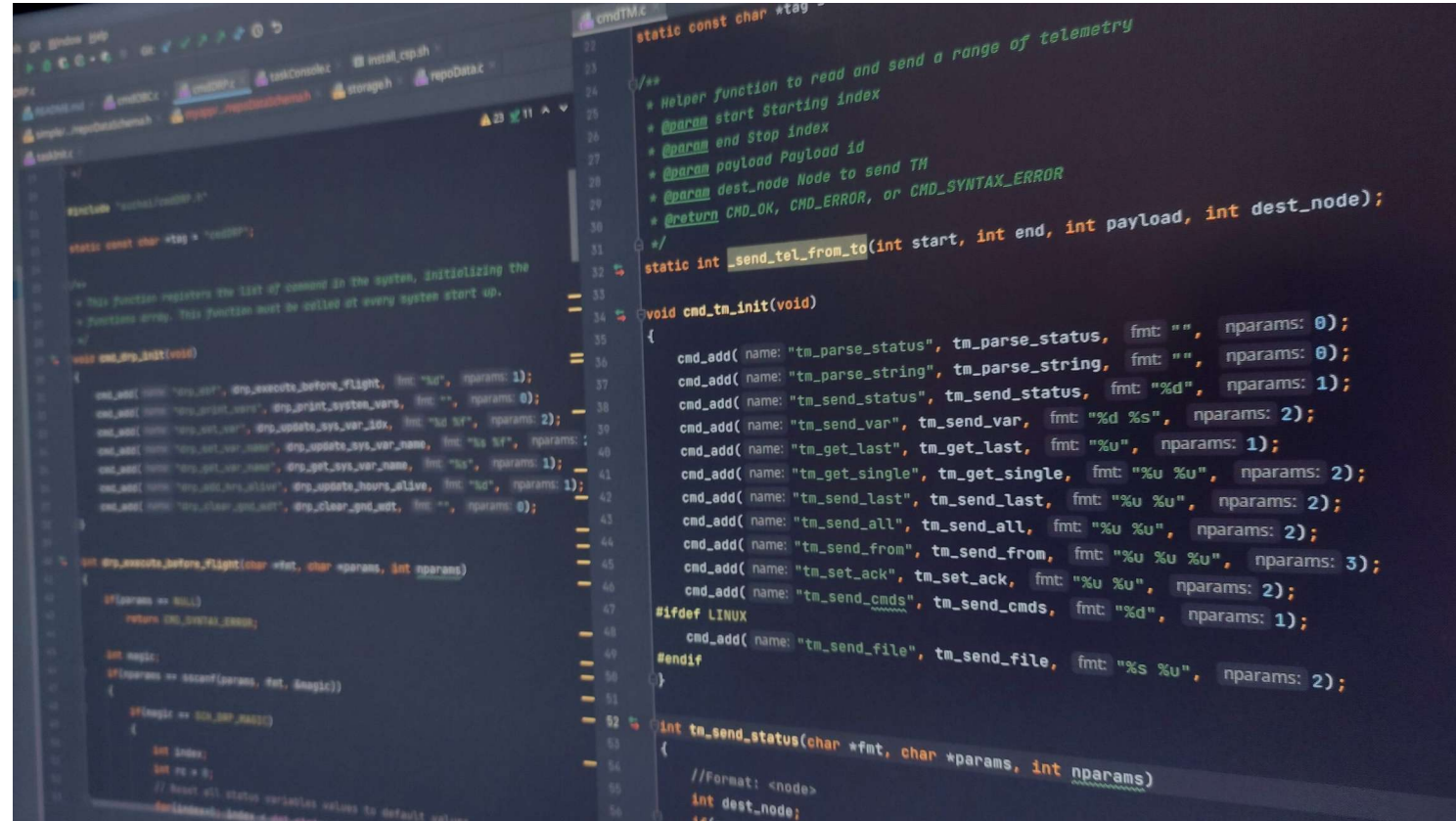
- Research software**
Different perspectives on research software
- Research software quality**
Indicators & principles for research software quality
- Research communities**
Use cases and software practices across research communities
- Your role**
Resources tailored to different roles in research software funding & development
- Your tasks**
Guidelines and solutions for tackling common software tasks
- All tools and resources**
Browse tools and resources for research software quality

{RSQ}Kit

A product of EVERSE



- Researchers who code
- Research software engineers
- Those managing and/or procuring funding for project with a large software component
- Those running research infrastructures using software
- Those developing software related policy at organisations and in projects



```
static const char *tag = "CMDSP";

/* This function registers the list of command in the system, initializing the
 * functions array. This function must be called at every system start up.
 */
void cmd_tm_init(void)
{
    cmd_add(name: "tm_parse_status", tm_parse_status, fmt: "", nparams: 0);
    cmd_add(name: "tm_parse_string", tm_parse_string, fmt: "", nparams: 0);
    cmd_add(name: "tm_send_status", tm_send_status, fmt: "%d", nparams: 1);
    cmd_add(name: "tm_send_var", tm_send_var, fmt: "%d %s", nparams: 2);
    cmd_add(name: "tm_get_last", tm_get_last, fmt: "%u", nparams: 1);
    cmd_add(name: "tm_get_single", tm_get_single, fmt: "%u %u", nparams: 2);
    cmd_add(name: "tm_send_last", tm_send_last, fmt: "%u %u", nparams: 2);
    cmd_add(name: "tm_send_all", tm_send_all, fmt: "%u %u", nparams: 2);
    cmd_add(name: "tm_send_from", tm_send_from, fmt: "%u %u %u", nparams: 3);
    cmd_add(name: "tm_set_ack", tm_set_ack, fmt: "%u %u", nparams: 2);
    cmd_add(name: "tm_send_cmds", tm_send_cmds, fmt: "%d", nparams: 1);
#ifdef LINUX
    cmd_add(name: "tm_send_file", tm_send_file, fmt: "%s %u", nparams: 2);
#endif
}

int tm_send_status(char *fmt, char *params, int nparams)
{
    //Format: <node>
    int dest_node;
}
```

<https://unsplash.com/photos/black-flat-screen-computer-monitor-Mm>



Tasks

- Common software tasks that improve quality
- Quality of practices vs quality of product

☰ Your tasks

Choosing languages, tools & infrastructures

How to decide which programming languages, tools and infrastructures to use?

☰ Your tasks

Citing software

How can people cite your software?

☰ Your tasks

Continuous Integration and Continuous Delivery/Deployment

How can you use CI/CD in software development?

☰ Your tasks

Creating a good README

How to create a good README document for software projects?

☰ Your tasks

Documenting software

How to document your software project?

☰ Your tasks

Documenting software using 'Read The Docs'

How to use 'Read The Docs' tool for software documentation?

☰ Your tasks

Improving environmental sustainability

How to measure and improve environmental sustainability of software?

☰ Your tasks

Licensing software

How to license your software for reuse?

☰ Your tasks

Organising software projects

How to organise your software project?

☰ Your tasks

Packaging & releasing software

How to package and release your software for distribution and reuse?

☰ Your tasks

Releasing software

How to release your software for reuse?

☰ Your tasks

Reproducible software environments

How to create a development environment for your software so others can run and contribute to your software?

☰ Your tasks

Software documentation

How to write clear and useful software documentation for developers and end-users

☰ Your tasks

Software identifiers

How to uniquely identify your software and its versions?

☰ Your tasks

Software metadata

How to describe your software using metadata?

☰ Your tasks

Task automation using GitHub Actions

How to set up GitHub Actions on software repositories for task automation

☰ Your tasks

Testing software

How to test your software?

☰ Your tasks

Using version control

How to version control your software?



☰ Your tasks
Choosing languages, tools & infrastructures

How to decide which programming languages, tools and infrastructures to use?

☰ Your tasks
Citing software

How can people cite your software?

☰ Your tasks
Continuous Integration and Continuous Delivery/Deployment

How can you use CI/CD in software development?

☰ Your tasks
Creating a good README

How to create a good README document for software projects?

☰ Your tasks
Documenting software

How to document your software project?

☰ Your tasks
Documenting software using 'Read The Docs'

How to use 'Read The Docs' tool for software documentation?

☰ Your tasks
Improving environmental sustainability

How to measure and improve environmental sustainability of software?

☰ Your tasks
Licensing software

How to license your software for reuse?

☰ Your tasks
Organising software projects

How to organise your software project?

☰ Your tasks
Packaging & releasing software

How to package and release your software for distribution and reuse?

☰ Your tasks
Releasing software

How to release your software for reuse?

☰ Your tasks
Reproducible software environments

How to create a development environment for your software so others can run and contribute to your software?

☰ Your tasks
Software documentation

How to write clear and useful software documentation for developers and end-users

☰ Your tasks
Software identifiers

How to uniquely identify your software and its versions?

☰ Your tasks
Software metadata

How to describe your software using metadata?

☰ Your tasks
Task automation using GitHub Actions

How to set up GitHub Actions on software repositories for task automation

☰ Your tasks
Testing software

How to test your software?

☰ Your tasks
Using version control

How to version control your software?

Second Question for you

What **tasks** do you think would be most important to detail?



<https://tinyurl.com/WLCGRSQKit>

FAIR Software

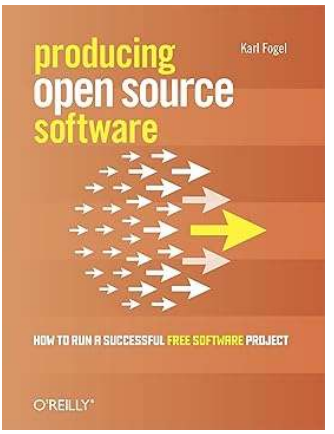
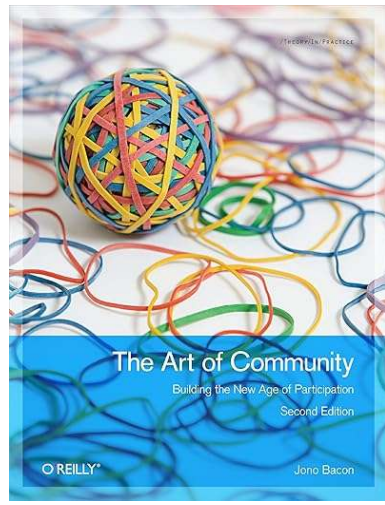
Open Research

Community Development

The FAIR4RS Principles are:

<p>F: Software, and its associated metadata, is easy for both humans and machines to find.</p> <p>F1. Software is assigned a globally unique and persistent identifier.</p> <ul style="list-style-type: none"> F1.1. Components of the software representing levels of granularity are assigned distinct identifiers. F1.2. Different versions of the software are assigned distinct identifiers. <p>F2. Software is described with rich metadata.</p> <p>F3. Metadata clearly and explicitly include the identifier of the software they describe.</p> <p>F4. Metadata are FAIR, searchable and indexable.</p>
<p>A: Software, and its metadata, is retrievable via standardized protocols.</p> <p>A1. Software is retrievable by its identifier using a standardized communications protocol.</p> <ul style="list-style-type: none"> A1.1. The protocol is open, free, and universally implementable. A1.2. The protocol allows for an authentication and authorization procedure, where necessary. <p>A2. Metadata are accessible, even when the software is no longer available.</p>
<p>I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.</p> <p>I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.</p> <p>I2. Software includes qualified references to other objects.</p>
<p>R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).</p> <p>R1. Software is described with a plurality of accurate and relevant attributes.</p> <ul style="list-style-type: none"> R1.1. Software is given a clear and accessible license. R1.2. Software is associated with detailed provenance. <p>R2. Software includes qualified references to other software.</p> <p>R3. Software meets domain-relevant community standards.</p>

Table 1: The FAIR Principles for Research Software



Community Health Analytics in Open Science



By UNESCO.org - Understanding open science (p:6)



abundance



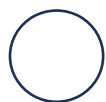
Research software infrastructure

It involves research software that captures more broadly accepted and used ideas, methods and models for use in research, and warrants close researcher involvement in their development.



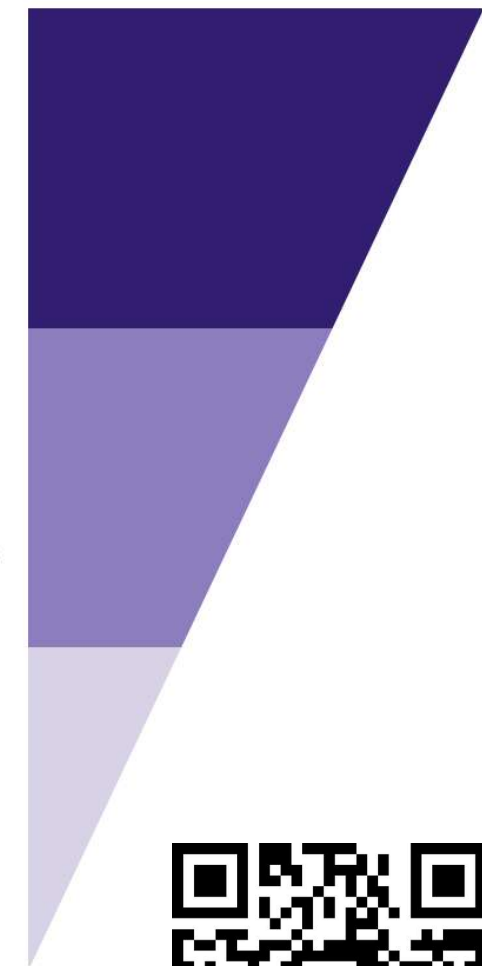
Prototype tools

It refers to research software that demonstrates a new idea, method or model for use by others outside the project within which it originated, often as a substantive intellectual contribution in its own right and often in the form of a proof of concept.



Analysis code

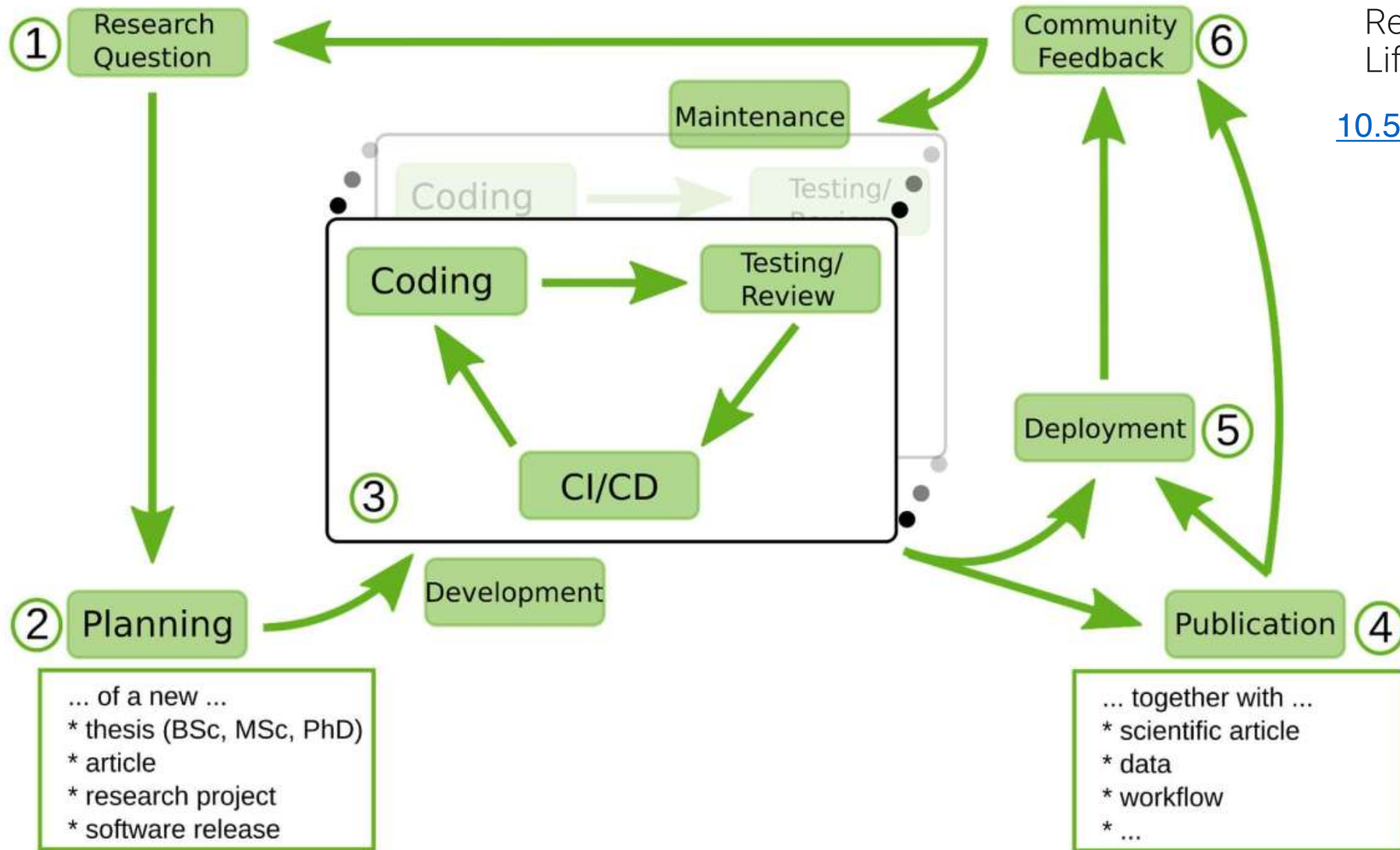
It includes research software that captures computational research processes and methodology, and often occurs in the context of simulation, data generation, preparation, analysis and visualisation.



Importance Reach

Foundational Software





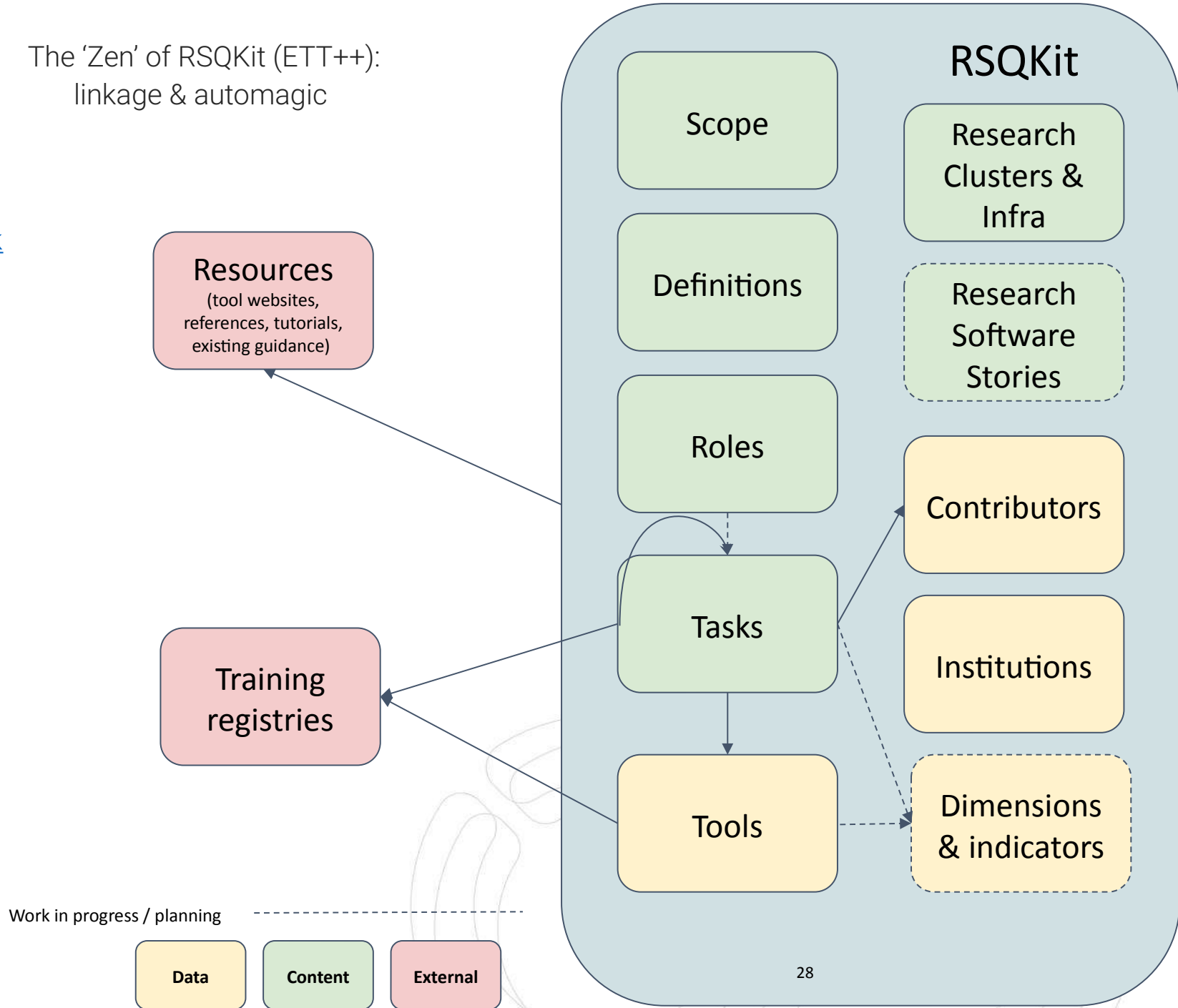
Research Software Lifecycle

[10.5281/zenodo.8324827](https://doi.org/10.5281/zenodo.8324827)



The 'Zen' of RSQKit (ETT++):
linkage & automagic

- From ETT (Elixir Toolkit Theme) - <https://elixir-belgium.github.io/elixir-toolkit-theme/>
 - Tasks to Tools
 - Tools to further information
 - Task & Tool linkage to training registry (limited to TeSS)
 - Related pages
- RSQKit additions (the ++) - Future work
 - Extending the training registries that are supported
 - Presenting the link between dimensions & indicators with tasks and tools in a meaningful way
 - Pathways?
 - Connecting roles with tasks



Events

Guidelines

List of topics

github.com/EVERSE-ResearchSoftware/RSQKit/issues/224

Reaching beyond the project

- deRSE25 (Feb)
 - WLCG/HEP Workshop (May)
 - CW25 (May)
 - EVERSE Network event (June)
 - RSECon25 (in planning) (Sept)
- Contributor Agreement
 - How to Contribute
 - What to Contribute
 - What Not to Contribute
 - Contributor Responsibilities
 - Acknowledgement of Contributions
 - Ownership of Content

everse.software/RSQKit/contribution_guidelines

Topics not yet started

- How to push your Dockerfile to the GitHub registry
- How to package your code for reuse
- How to manage your software project
- How to maintain your research software project
- How to collaborate on an open source software project
- How to develop a software application into a (cloud) software service
- How to establish your software in the community and support this community
- Comparison of tools for automated FAIR software assessment
- Understanding quality metrics and how to use them
- How to refactor your software/code
- How to design software for sustainability
- How to conduct static analysis
- How to conduct dynamic analysis
- How to conduct architectural recovery and reconstruction
- How to add your first GitLab CI
- How to use containers and why should I use them
- How to use design patterns
- How to provide complete bibliographic metadata as a codemeta file
- How to publish your software to contribute to the Scholarly Knowledge Graphs
- Basic security checks for web based software and APIs
- Profiling your software

Topics with an issue

- [Document your Computational Workflow #207](#)
- [Metadata documenting your Code task #206](#)
- [How to publish your software package to a package repository #197](#)
- [How to cite your code repository and maintain a Citation File \(CFF\) #196](#)
- [How to structure the files in your software project #195](#)
- [How to test and automate the testing of your software project #194](#)
- [How to add your first GitHub action #193](#)



eosc | EVERSE Road Map – next 6 months

- Additional prioritised task page content
- Improved navigation
 - Life cycle stages and tasks
 - Marking up task pages with indicators and dimensions
- Integration with software quality tool efforts in EVERSE
 - At what stage to use which tool
- A better description of software roles
- Additional external facing workshops & contentathons
- A newsletter
 - Internal first to reach out across the project
 - Moving to the EVERSE network to gain further input / updates
- Refreshing the layout




























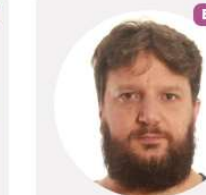
everse.software/network/



Last Question...

Can you rank relative **importance to you** of: tasks, exemplars, tools for software quality metrics, roles, high/minimum guides, glossary of tools



 <p>Editor</p> <p>Patrick Bos Netherlands eScience Center</p> <p>🗨️ ✉️ ID</p>	 <p>Contributor</p> <p>SC</p> <p>Salvador Capella-Gutierrez Barcelona Supercomputing Centre (BSC)</p> <p>✉️</p>	 <p>Editor</p> <p>Shoaib Sufi The University of Manchester / Software Sustainability Institute</p> <p>🗨️ ID</p>	 <p>Contributor</p> <p>Valentin Churavy JGU Mainz & University of Augsburg</p> <p>🗨️ ID</p>	 <p>Editor</p> <p>Aleksandra Nenadic The University of Manchester / Software Sustainability Institute</p> <p>🗨️ ID</p>	 <p>Author</p> <p>Azza Gangami</p> <p>🗨️</p>	 <p>Contributor</p> <p>Carole Goble The University of Manchester / ELIXIR-UK</p> <p>🗨️ ID</p>	 <p>Author</p> <p>Christian Hüser HZDR</p> <p>🗨️</p>	 <p>Editor</p> <p>Daniel Garijo Universidad Politecnica de Madrid</p> <p>🗨️ ✉️ ID</p>	 <p>Contributor</p> <p>Elena Breitmoser University of Edinburgh</p> <p>🗨️ ✉️ ID</p>
 <p>Editor</p> <p>Shraddha Bajare Square Kilometre Array Observatory</p> <p>🗨️ ✉️</p>	 <p>Author</p> <p>Simon Christ Leibniz University Hannover</p> <p>🗨️ ✉️ ID</p>	 <p>Author</p> <p>Thomas Vuillaume LAPP, University Savoie Mont-Blanc, CNRS</p> <p>🗨️ ✉️ ID</p>	 <p>Editor</p> <p>Zhiming Zhao Universiteit van Amsterdam</p> <p>🗨️ ✉️</p>	 <p>Author</p> <p>Esteban González</p> <p>🗨️</p>	 <p>Author</p> <p>Eva Martín del Pico</p> <p>🗨️ ID</p>	 <p>Editor</p> <p>Fotis Psomopoulos Institute of Applied Biosciences (INABICERTH) / ELIXIR-GR</p> <p>🗨️ ✉️ ID</p>	 <p>Author</p> <p>Giacomo Peru</p> <p>🗨️</p>	 <p>Contributor</p> <p>Graeme A Stewart CERN</p> <p>🗨️ ✉️ ID</p>	 <p>Contributor</p> <p>Jan Küster Universität Bremen</p> <p>🗨️ ID</p>
 <p>Editor</p> <p>Jason Maassen Netherlands eScience Center</p> <p>🗨️ ✉️ ID</p>	 <p>Contributor</p> <p>Kenneth Rioja CERN</p> <p>🗨️ ✉️ ID</p>	 <p>Author</p> <p>Kirsty Pringle</p> <p>🗨️</p>	 <p>Editor</p> <p>Laura Portell-Silva Barcelona Supercomputing Center / ELIXIR-ES</p> <p>🗨️ ✉️ ID</p>	 <p>Editor</p> <p>Michael Sparks The University of Manchester / Software Sustainability Institute</p> <p>🗨️ ✉️</p>	 <p>Editor</p> <p>Pablo Martinez-Diaz Barcelona Supercomputing Center / ELIXIR-ES</p> <p>🗨️ ✉️ ID</p>				

Editorial Board –
rsqkit@lists.certh.gr

Thank you!

Contact: contact@everse.software

michael.sparks@manchester.ac.uk

EVERSE@WLCG-HSF:

- EVERSE Overview – last talk
- Use Cases and the Research Software Quality Kit – this talk
- The EVERSE Training and Recognition Plan – in the training session this afternoon

EVERSE Network

<https://ec.europa.eu/eusurvey/runner/EVERSENetworkJoinIndividual>

Website:

<https://www.everse.software/>

BlueSky:

<https://bsky.app/profile/eosc-everse.bsky.social>

LinkedIn:

<https://www.linkedin.com/company/eosc-everse/>

FOSSTodon:

https://fosstodon.org/@eosc_everse



**Funded by
the European Union**

This project has received funding from the European Union's Horizon Europe Programme under GA 101129744 – EVERSE – HORIZON-INFRA-2023-EOSC-01-02

