



The LifeWatch ERIC Biodiversity
& Ecosystem eScience Conference

PLOVDIV, BULGARIA

7-10 JULY 2026

EVERSE RSQKit indicators and tools in Science clusters

Zhiming Zhao

University of Amsterdam

LifeWatch ERIC - Virtual Lab & Innovation Center (VLIC)

July 2026



UNIVERSITEIT
VAN AMSTERDAM



EVERSE

Software in research activities

What?...

How?...

Observe

Measure

Remote sensing

UAV/Lidar

Satellite

....

inverse models

Complex system modelling

Model composition

Multi scale modelling

Data centric modelling..

....

Why?...

What-if

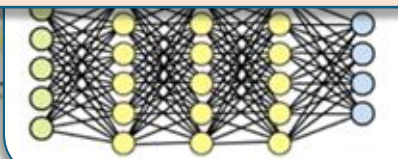
Scenarios

Workflows

Digital twins

...

$$\begin{aligned} \rho \frac{Dv_x}{Dt} &= -\frac{\partial p}{\partial x} + \mu \Delta v_x + f_x \\ \rho \frac{Dv_y}{Dt} &= -\frac{\partial p}{\partial y} + \mu \Delta v_y + f_y \\ \rho \frac{Dv_z}{Dt} &= -\frac{\partial p}{\partial z} + \mu \Delta v_z + f_z \\ \frac{D}{Dt} &= \frac{\partial}{\partial t} + v_x \frac{\partial}{\partial x} + v_y \frac{\partial}{\partial y} + v_z \frac{\partial}{\partial z} \end{aligned}$$





Expected learning outcomes

- **Understand** research software quality: the model and assessment process
- **Understand** the EVERSE research software toolkits and their application in a scientific domain
- **Gain practical familiarity** with research software quality for Python and Jupyter notebooks



Outline

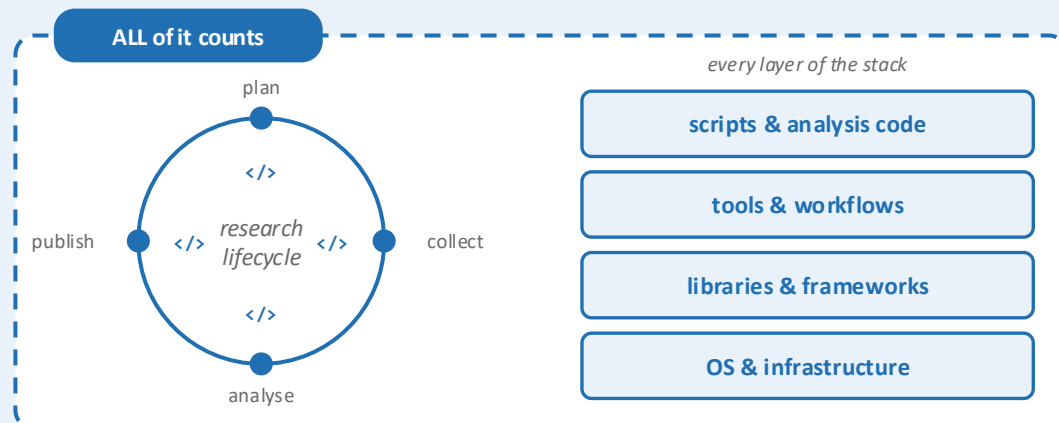
1. Research software
2. Software quality
3. EVERSE Research Software Quality Toolkit (RSQKit)
4. Case study
5. *Hands on*

1. Research software

What Counts as Research Software?

Two ways to draw the boundary — an inclusive and an exclusive view

The inclusive view *software in research — the whole environment counts*



- All code & software artefacts used or produced in one or more stages of the research lifecycle
- Regardless of the layer of the software stack
- Includes software not necessarily developed to be part of research

EXAMPLES — RESEARCH-SPECIFIC AND GENERAL-PURPOSE ALIKE

Python / R scripts

Jupyter notebooks

Excel & macros

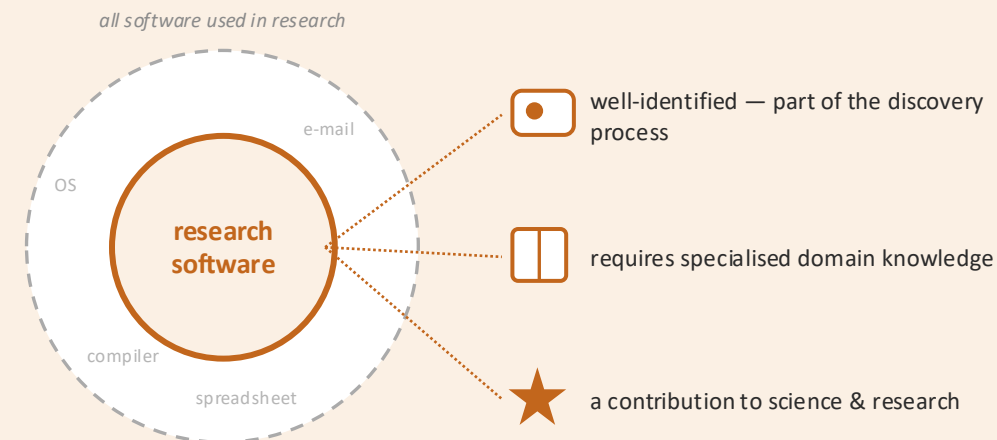
Git & CI pipelines

NumPy & libraries

Linux / HPC stack

VS

The exclusive view *research software — only the scientific core counts*



- Well-identified software that is part of the research discovery process
- Requires specialised domain knowledge
- A contribution to science and research

EXAMPLES — DOMAIN CODES AT THE SCIENTIFIC CORE

GROMACS · molecular dynamics

CESM · climate modelling

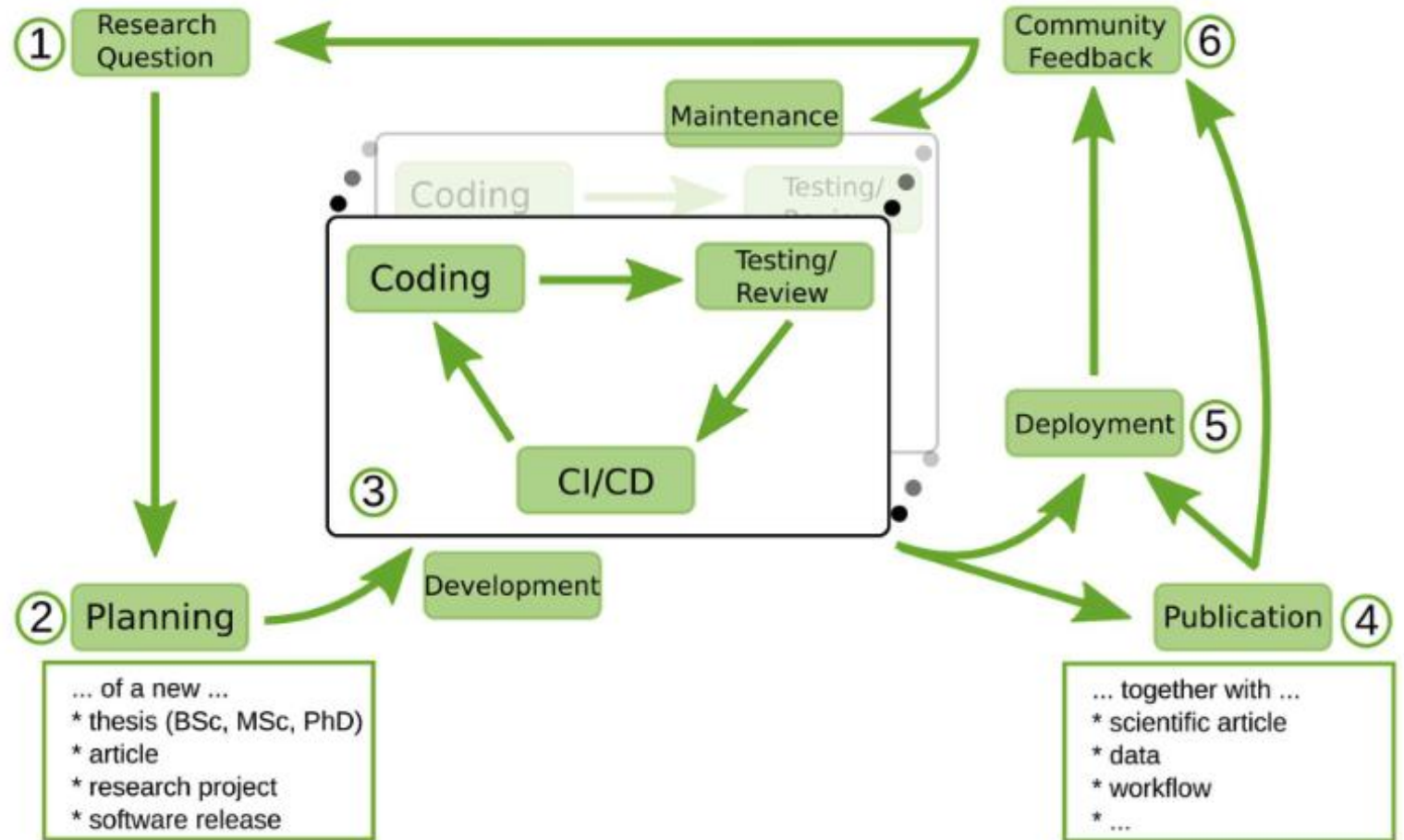
BLAST · sequence analysis

AstroPy · astronomy

Same landscape, different boundary: where the line is drawn determines what gets cited, credited, funded and sustained.

Software in the research lifecycle

- Different from industria
- Lifecycle
- Research software tiers
- Open science





Three Tiers of Research Software

From personal analysis code to shared infrastructure — rising reuse, audience and sustainability demands

T3

Research software infrastructure

Broadly accepted, widely used ideas, methods & models

community codes · warrants close, ongoing researcher involvement



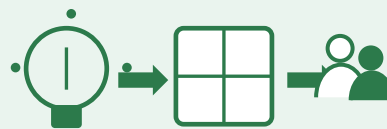
shared foundation · whole community

T2

Prototype tools

Demonstrates a new idea, method or model for others

proof of concept · substantive intellectual contribution · used beyond its origin project



new idea → shareable tool → others

T1

Analysis code

Captures the computational research process & methodology

simulation · data generation · preparation · analysis · visualisation



data → analysis code → result

How the tiers compare

	T1	T2	T3
Primary audience	the author / team	peers beyond the project	the research community
Main purpose	get results	demonstrate an idea	provide a dependable tool
Reuse & reach	low 	medium 	high
Engineering rigour	light 	moderate 	high · sustained
Typical form	scripts, notebooks	proof-of-concept tool	community code / library

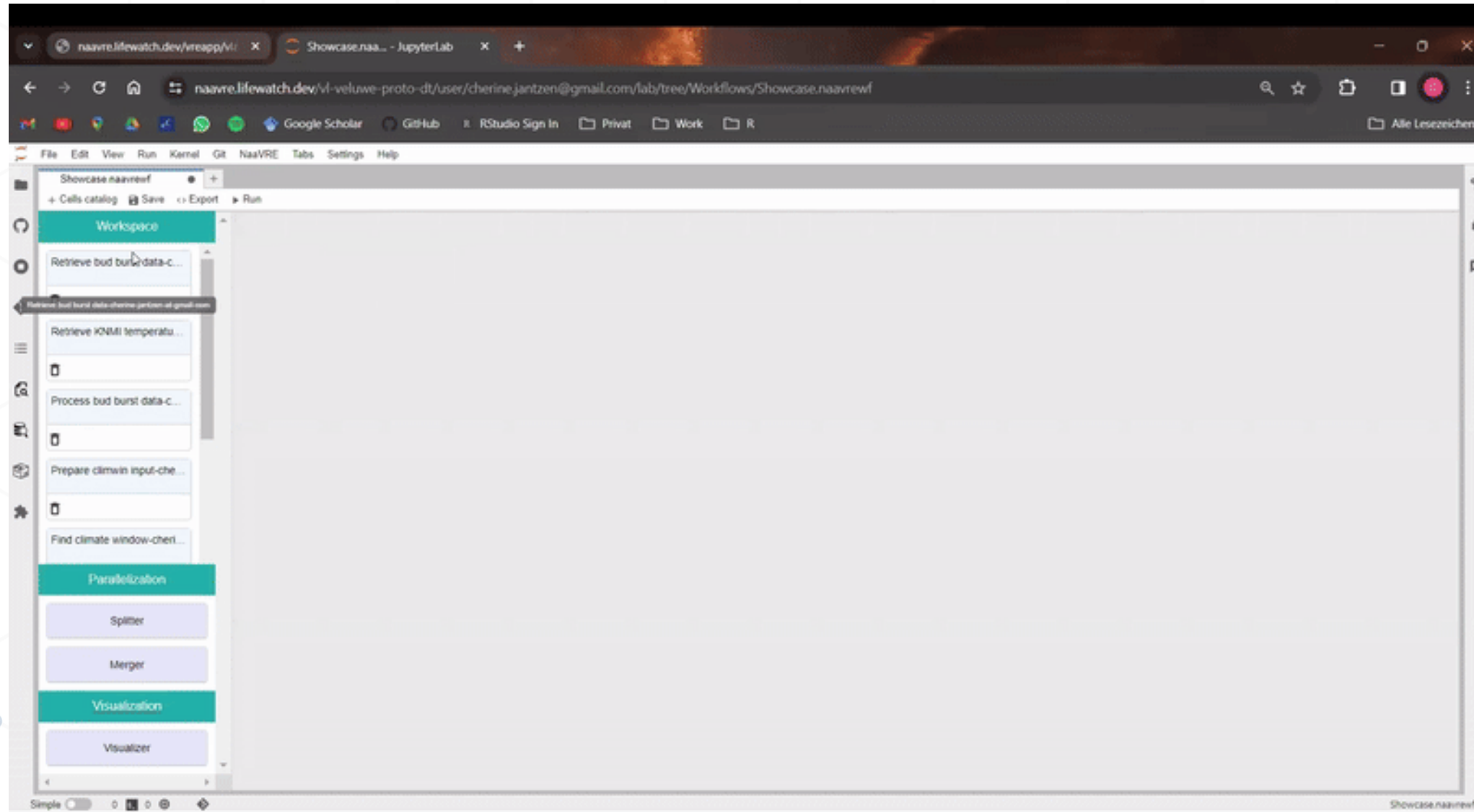
each tier builds on the one below

Same code DNA, rising responsibility: as software moves from Tier 1 to Tier 3, its audience widens and the need for identification, credit and sustained engineering grows.



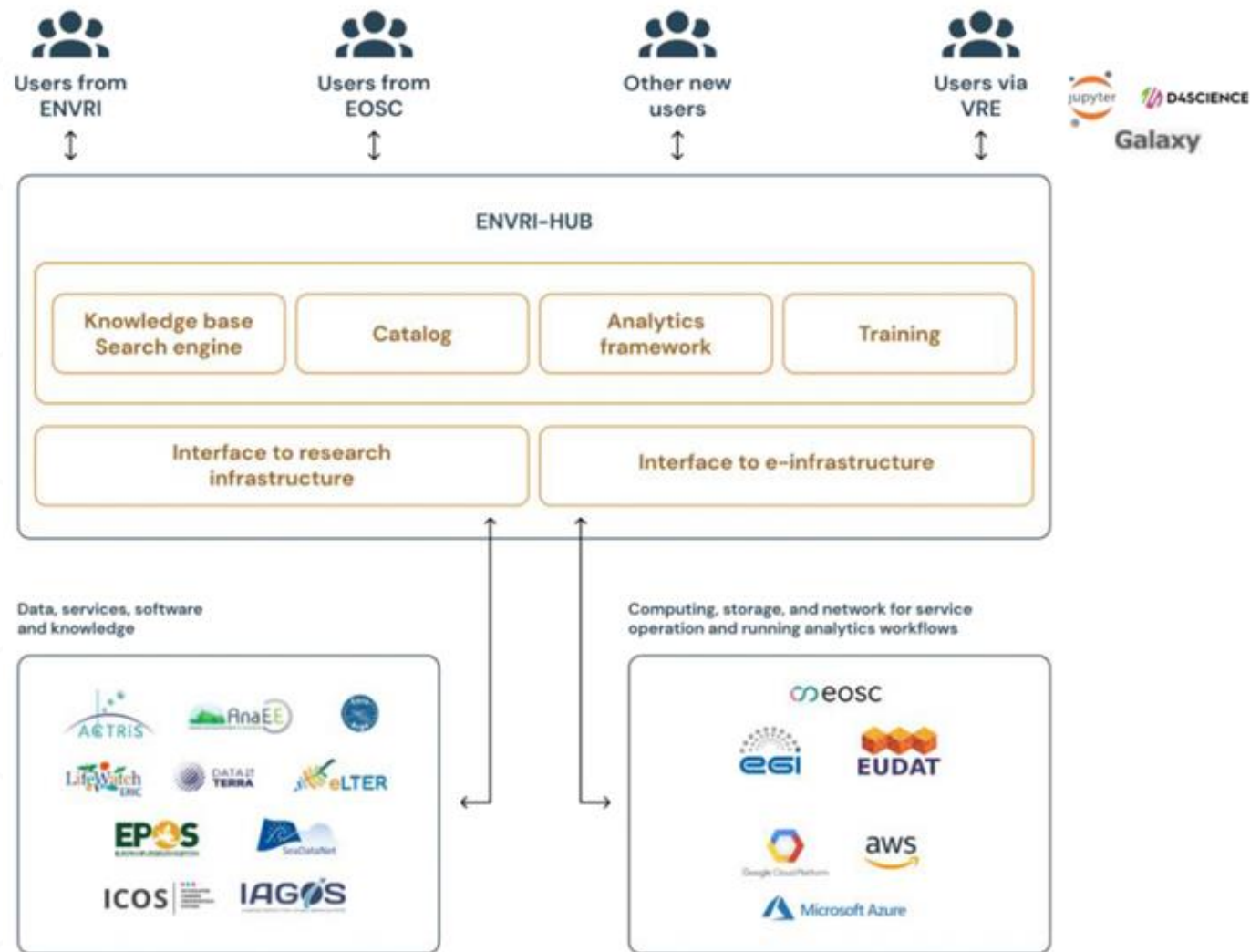
Tier 3 research software example: VRE

An experiment is a workflow of services, with customized input data and infrastructure



Tier 3 research software example: ENVRI-HUB framework

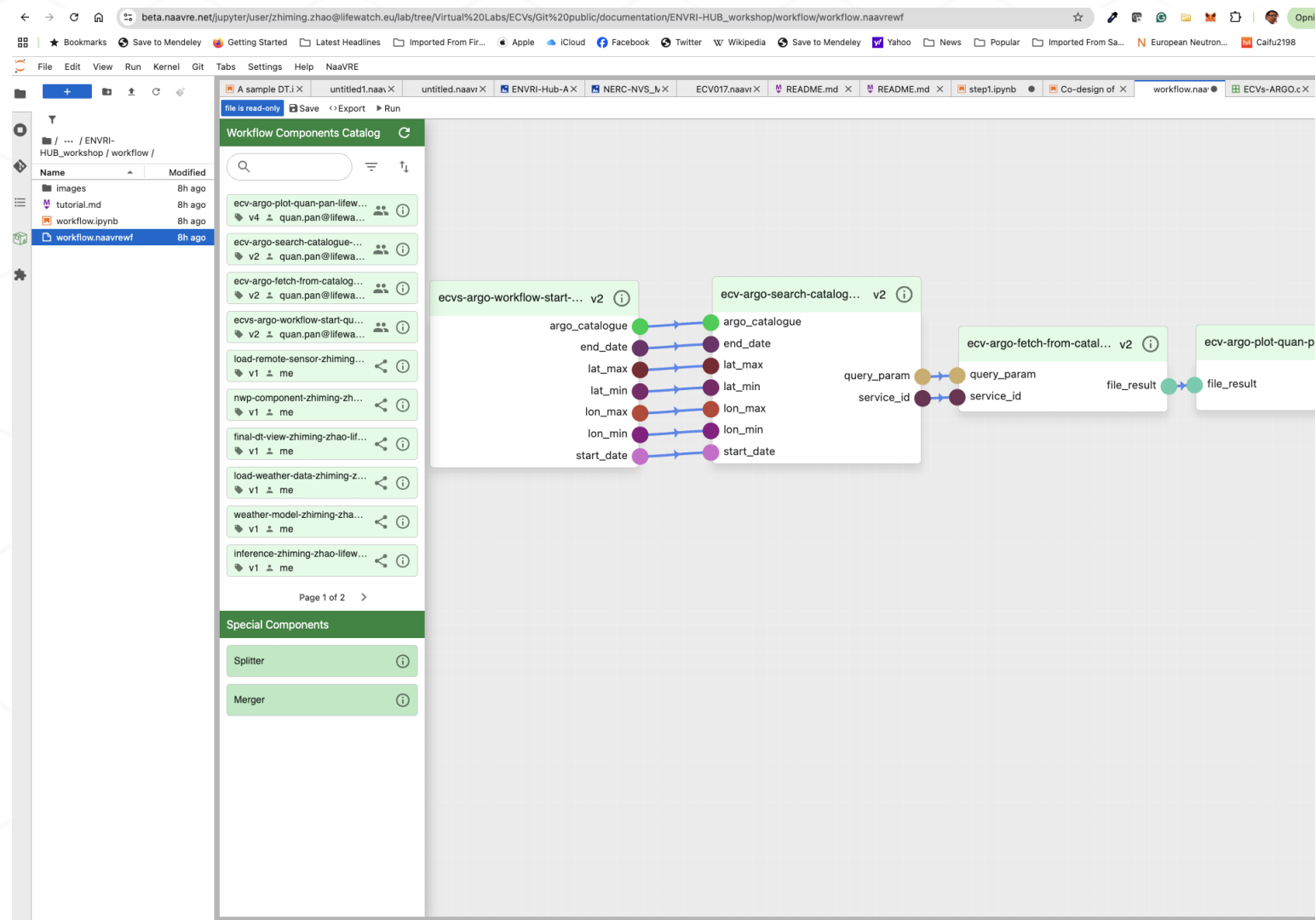
- ENVRI –HUB
- Catalog
- Knowledge base and Search engine
- Analytics Framework
- Training
- Interface to VRE
- Interface to underlying RI and Cloud





Tier 2 research software example: ECV workflow

■ Reusable workflows





Research software challenges

- Findability, Accessibility, Interoperability and Reusability of Research Software
- Reproducibility and Portability
- Sustainability and funding
- Usability
- Maintenance
-

Thematic community
nodes



EUROPEAN OPEN
SCIENCE CLOUD

National nodes

EVERSE 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**.



European e-
infrastructures

Reference EOSC EU
Node

Partners, associates, and affiliated entities



CERTH
CENTRE FOR
RESEARCH & TECHNOLOGY
HELLAS



HZDR
HELMHOLTZ ZENTRUM
DRESDEN ROSSENDORF



POLITÉCNICA



SKAO



Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación



THE UNIVERSITY
of EDINBURGH



UNIVERSITÀ
DEL SALENTO
The University of the two seas



Friedrich-Alexander-Universität
Erlangen-Nürnberg



UNIVERSITY
OF AMSTERDAM

netherlands

eScience center



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



★ **Vision:** establish research software as a first-class citizen of the scientific process — quality you can measure, tools you can reuse, and engineers who get the credit.

Core Project Outputs



RSQkit — Quality Toolkit

Community-driven knowledge base of best practices, guidelines & quality indicators across disciplines.

RSQkit



Automated Assessment

Automatically evaluates FAIRness, maintainability & sustainability of software.

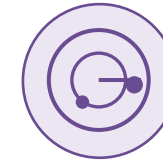
resqui-Pipeline



Analytics Dashboards

Track and explore software-quality metrics visually.

DashVERSE



Technology Monitoring

Monitors and advances emerging research-software technologies.

TechRadar

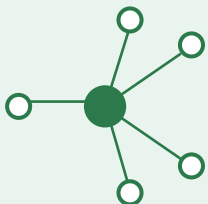


Recognition & Reward

Frameworks giving RSEs academic credit & career progression.

RSE careers

Community & Network Impact



European Network of Research Software Quality

A collaborative network to share knowledge, cross-fertilise research domains, and raise awareness of software as a core enabler of European research.



Scientific Use-Cases across EOSC Science Clusters

Tools integrated with major EOSC Science Clusters to demonstrate real gains in software quality and code reusability:

Life sciences

Astronomy

Environmental

Social sciences

The throughline: quality, recognition and community — turning research software into dependable, credited, reusable research output.

2. Research software quality

Software Quality — Concepts, Standards & References

What software quality is, how it is structured, and the standards that define it

Q What is it? The degree to which software satisfies stated and implied needs — uniting **conformance to requirements** (built right) with **fitness for purpose** (the right thing built).

Basic concepts — four complementary lenses

Two views



Two requirement types



Two measurement points



Two checks

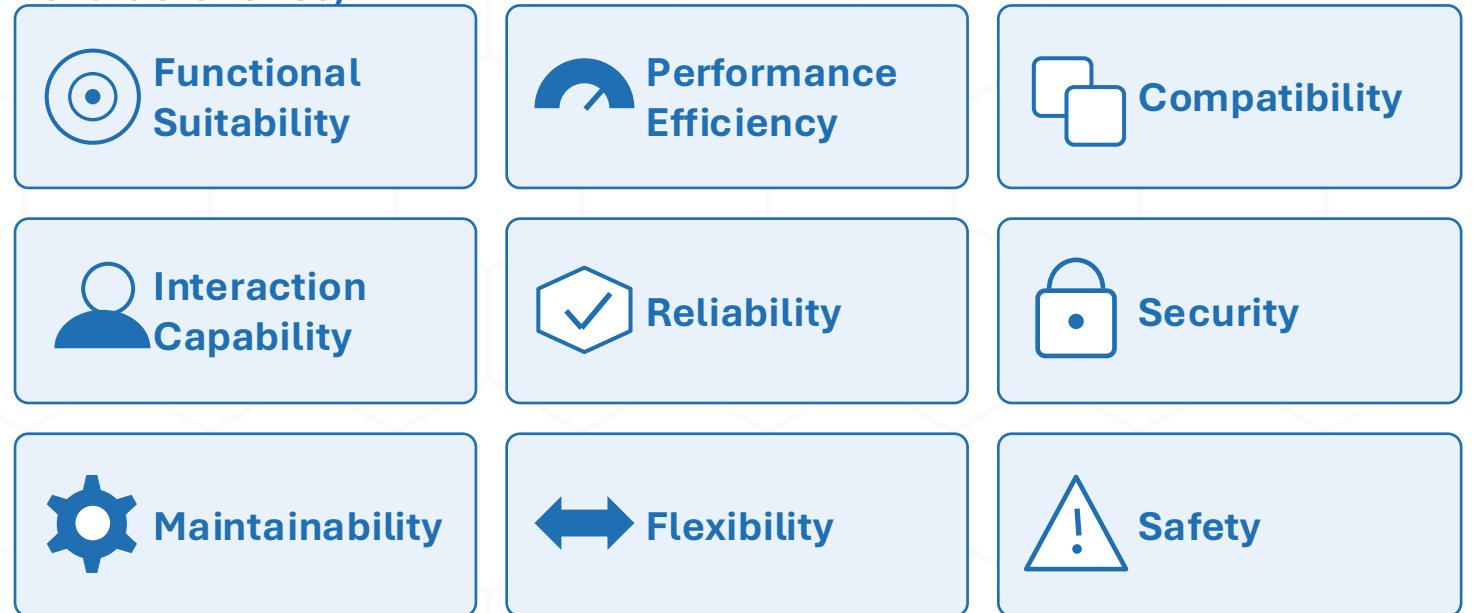


Standards & references



In practice: quality is defined by a model, made measurable through characteristics & sub-characteristics, and evaluated against standards — not judged by opinion.

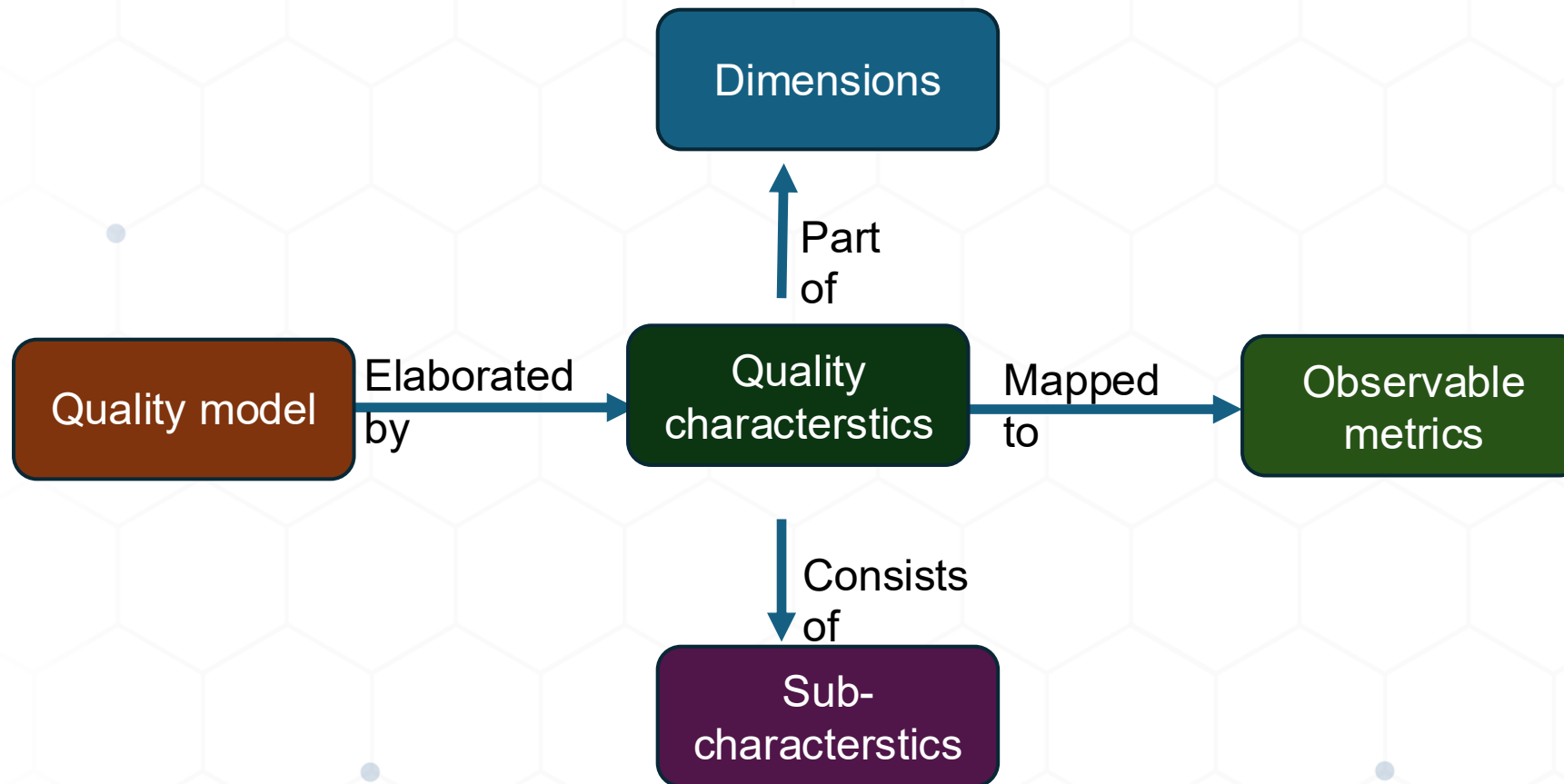
The reference model — ISO/IEC 25010:2023 product quality model (9 characteristics)



+ a companion Quality-in-Use model (effectiveness, efficiency, satisfaction, freedom from risk, context coverage) — ISO/IEC 25019

Software quality: a taxonomy

■ In ISO/IEC 25010



- Quality characteristics:
 - Functional suitability
 - Performance efficiency
 - Compatibility
 - Interaction capability
 - Reliability
 - Security
 - Maintainability
 - Flexibility
 - Safety

3. EVERSE Research software quality ToolKit (RSQKit)

Research software & quality >

Research software stories >

Research clusters or infrastructures >

Roles

Researcher who codes

Research Software Engineer

Principal Investigator

Project manager

Product owner

Policy maker

Trainer

Tasks (alphabetical) >

All indicators

All tools and resources

Get involved >

Site Statistics

Find your page...

 roles

Policy maker

Software development and quality guidance for policy makers

 roles

Principal Investigator (PI)

Software development and quality guidance for Principal Investigators (PIs)

 roles

Product owner

Software development and quality guidance for research software product owners

 roles

Project manager

Software development and quality guidance for research software project managers

 roles

Research Software Engineer (RSE)

Software development and quality guidance for Research Software Engineers (RSEs)

 roles

Researcher who codes

Software development and quality
guidance for researchers who code roles

Trainer

Software development and quality guidance for trainers



BEEs 2026 Research Software Tasks & Their Dependencies

34 RSQKit tasks, clustered by shared keywords into six lifecycle stages — arrows show how the outputs of one stage feed the next

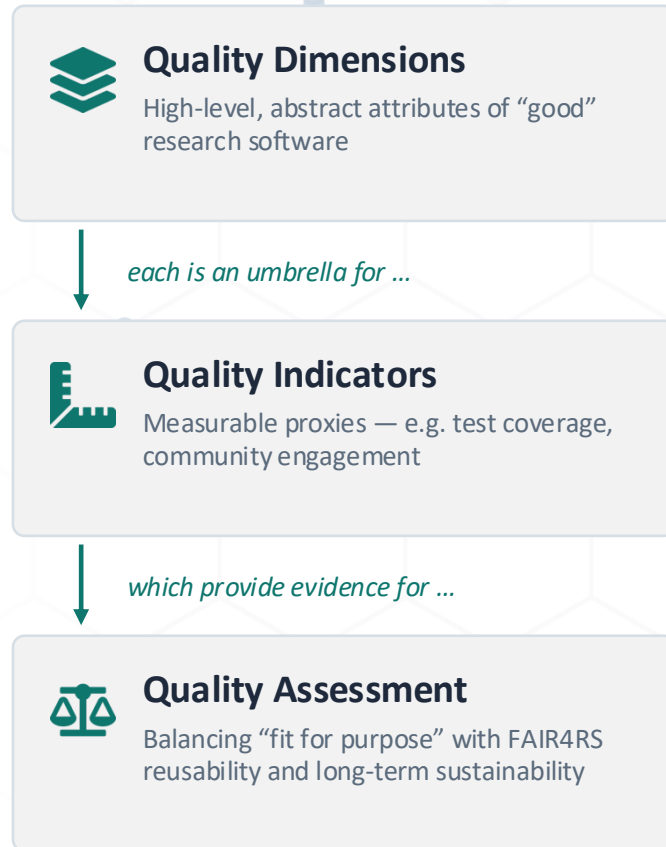


 **Adopting FAIR research software practices** — cross-cutting task underpinning every stage *keywords: FAIRness · openness · reproducibility · reuse*

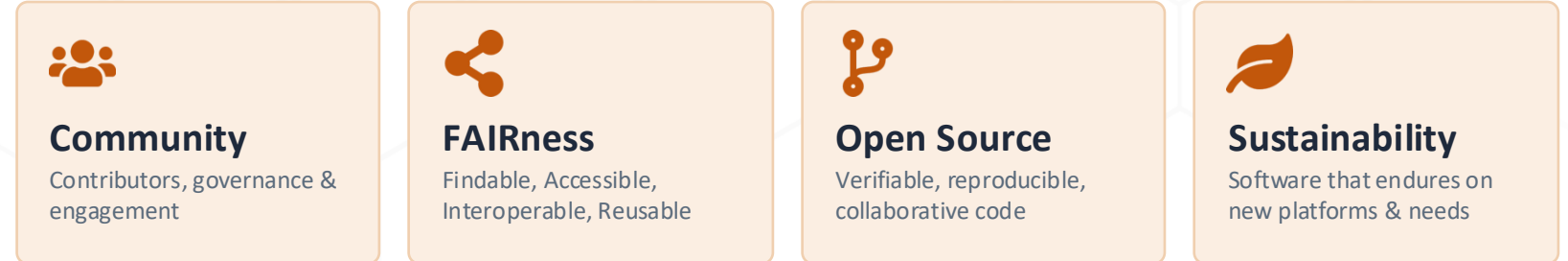
The Quality Model of Research Software

13 quality dimensions — each an umbrella for measurable quality indicators · EVERSE RSQKit (https://everse.software/RSQKit/quality_dimensions)

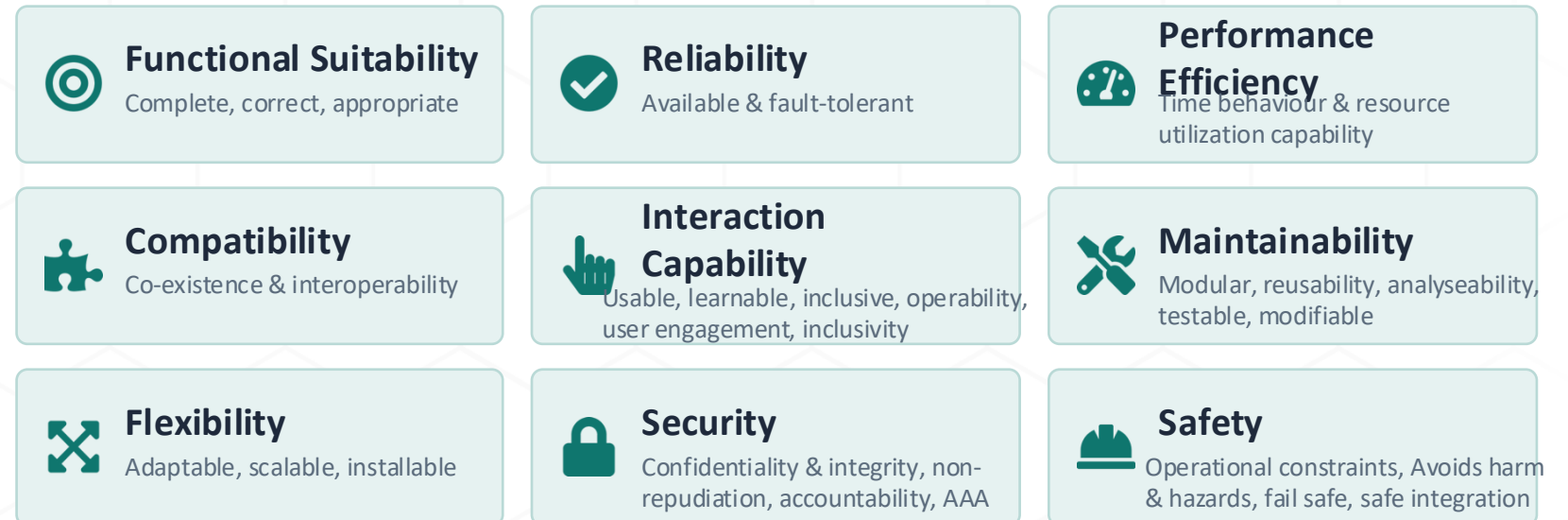
HOW THE MODEL WORKS



RESEARCH-SOFTWARE-SPECIFIC DIMENSIONS *the scientific context*

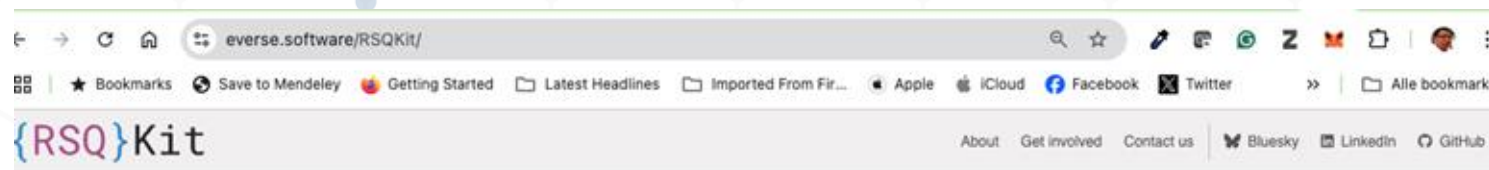


ENGINEERING DIMENSIONS *from ISO/IEC 25010*



What will EVERSE offer?

- Community "good enough" practices
- Research a software quality model
- Tools to measure the quality characteristics
- Support and training



Research Software Quality Toolkit for Sciences

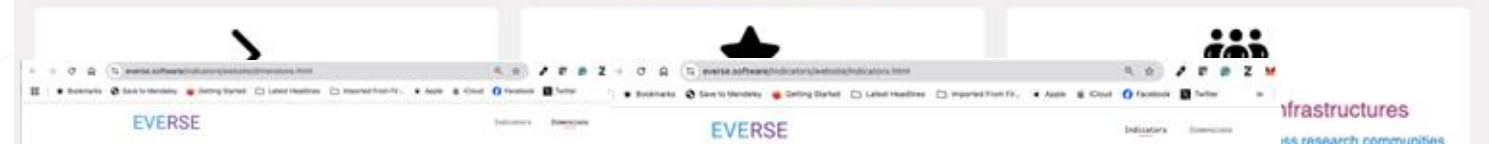
Research Software Quality Toolkit (RSQKit) lists curated best practices, tools and resources for improving the quality of research software

What can we help you find?



Search RSQKit ...

Browse all topics by



Research Software Quality Dimensions

* `research_software_quality_dimensions` represents criteria relevant for assessing software quality (e.g., Reusability, Performance, etc.).

Topic	Description	Source
Reusability	Reusability refers to the degree to which software components can be reused in other projects, systems or environments, which requires the implementation of clear interfaces and documentation.	Research Software Quality Toolkit
Performance	Performance refers to the degree to which software components are able to process data efficiently, accurately, and reliably. This includes, among other things, the ability to handle large amounts of data and to process it quickly.	Research Software Quality Toolkit
Reliability	Reliability refers to the degree to which a product can be expected to perform its intended function in a consistent and predictable manner.	Research Software Quality Toolkit
Interoperability	Interoperability refers to the degree to which a product or system can be expected to interact with other products or systems in a consistent and predictable manner.	Research Software Quality Toolkit
Security	Security refers to the degree to which a product or system is able to protect its data and resources from unauthorized access, use, or disclosure.	Research Software Quality Toolkit
Accessibility	Accessibility refers to the degree to which a product or system is able to be used by people with disabilities.	Research Software Quality Toolkit
Portability	Portability refers to the degree to which a product or system is able to be used on different platforms or operating systems.	Research Software Quality Toolkit
Compliance	Compliance refers to the degree to which a product or system is able to meet the requirements of relevant standards, regulations, or policies.	Research Software Quality Toolkit
Documentation	Documentation refers to the degree to which a product or system is accompanied by clear and comprehensive documentation.	Research Software Quality Toolkit
Support	Support refers to the degree to which a product or system is accompanied by clear and comprehensive documentation.	Research Software Quality Toolkit

Research Software Quality Indicators

* `research_software_quality_indicators` represents a specific software aspect that can be measured (e.g., Reusability, Test coverage, Documentation coverage, etc.).

Topic	Keywords	Quality Dimension	Description	Source
Code quality	code quality, code review, code coverage	Performance	Indicator that checks the compliance of a codebase with the rules of a code quality framework that are defined and maintained. This indicator does not assess the quality of the research software itself.	No source available
Software development management	Software development management, software development	Interoperability	Indicator that assesses the compliance of a codebase with the rules of a software development management framework that are defined and maintained. This indicator does not assess the quality of the research software itself.	Software Quality Indicators from the European Union (EU) - Software Quality Indicators from the European Union (EU)
No external dependencies	security, external dependencies	Security	Checks if external dependencies are used in a secure and reliable manner.	Spotify Best practices for external dependencies
No known vulnerabilities	security, vulnerabilities, security, security, security	Security	Checks if known vulnerabilities (e.g., CVEs, etc.) are listed in the public CVE database.	Spotify Best practices for external dependencies

This project has received funding from the European Union's Horizon Europe Programme under the Marie Skłodowska Curie Grant Agreement No. 101019760.

This page was built on a Free template designed by [Freemove](#) on 10/05/2024 10:00:00



EVERSE
NETWORK
RESEARCH SOFTWARE QUALITY

 Editor Patrick Bos Netherlands eScience Center	 Contributor SC Barcelona Supercomputing Centre (BSC)	 Editor Shoaib Sufi The University of Manchester / Software Sustainability Institute	 Contributor Valentin Churavy JGU Mainz & University of Augsburg	 Editor Aleksandra Nenadic The University of Manchester / Software Sustainability Institute	 Author Azza Gamgani	 Contributor Carole Goble The University of Manchester / ELIXIR-UK	 Author Christian Hüser HZDR	 Editor Daniel Garijo Universidad Politécnica de Madrid	 Contributor Elena Breitmöser University of Edinburgh
 Editor Shraddha Bajare Square Kilometre Array Observatory	 Author Simon Christ Leibniz University Hannover	 Author Thomas Vuillaume LAPP, University Savoie Mont-Blanc, CNRS	 Editor Zhiming Zhao Universiteit van Amsterdam	 Author Esteban González	 Author Eva Martín del Pico	 Editor Fotis Psomopoulos Institute of Applied Biosciences (IABICERTH) / ELIXIR-GR	 Author Giacomo Perù	 Contributor Graeme A. Stewart CERN	 Contributor Jan Küster Universität Bremen
				 Editor Jason Maassen Netherlands eScience Center	 Contributor Kenneth Riqua CERN	 Author Kirsty Fringle	 Editor Laura Porcell-Silva Barcelona Supercomputing Center / ELIXIR-ES	 Editor Michael Sparks The University of Manchester / Software Sustainability Institute	 Editor Pablo Martínez-Díaz Barcelona Supercomputing Center / ELIXIR-ES

Editorial Board –
rsqkit@lists.certh.gr

<https://everse.software/services/rsqkit/>

RSQKit All Contributors



Quality check tools

everse.software/TechRadar/

Save to Mendeley Getting Started Latest Headlines Imported From Fir... Apple iCloud Facebook Twitter Wikipedia

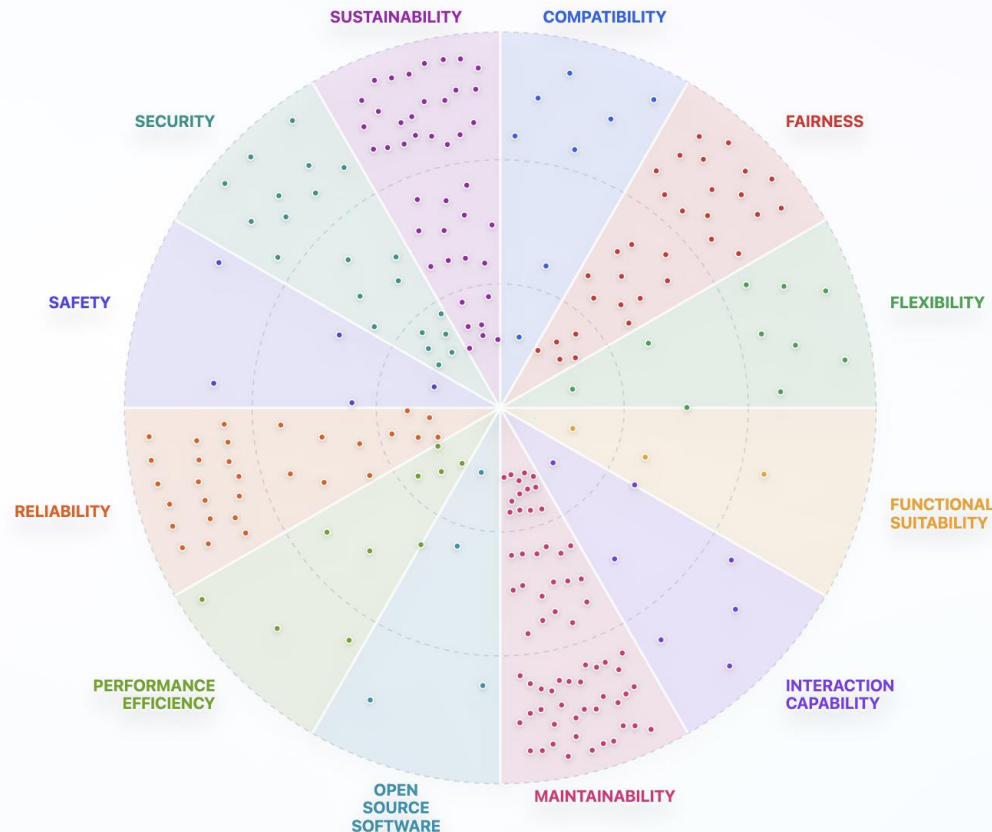
eosc | EVERSE TechRadar

About Suggest a new tool

Research Software Quality Tools

A curated catalog of tools and services to measure and improve research software quality.

Browse Catalog



eosc | EVERSE TechRadar

About Suggest a new tool

Filters

Your software's targets

- Communities
- Research Teams
- Individuals

Your Software's Programming Language

- ☐ #
- ☐ C++
- ☐ Dart
- ☐ Elixir
- ☐ Elm
- ☐ Fortran
- ☐ Go
- ☐ IDL
- ☐ ...

Apptainer

Free

High-performance container platform designed for scientific computing and HPC...

compatibility flexibility reliability

Auto-CodeMeta Generator

Free

This repository contains a (client-side) web application to automatically generate...

fairness

bandit

Free

Security-focused static analysis tool for Python that identifies common security vulnerabilities...

security

bearer

Free

Bearer CLI is a static application security testing (SAST) tool that scans your source code and...

security

Black

Free

Deterministic Python code formatter that enforces consistent coding style across...

maintainability

cffinit

Free

Web-based tool for creating standardized CITATION.cff files that enable proper citation of...

fairness

Checkstyle

Free

Java code style checker that enforces coding standards and best practices in research...

maintainability

Choose a License

Free

Choose a License is a website that helps developers understand and select open...

sustainability

CMake

Free

Cross-platform build system generator that simplifies the build process for research...

maintainability compatibility

EVERSE TechRadar

Back to Catalog

About Suggest a new tool

F-UJI

fairness

Visit Website

Automated assessment tool for data FAIRness with extended version for research software based on FAIR4RS principles, enabling systematic evaluation and improvement of research output compliance with FAIR standards.

MEASURES QUALITY INDICATORS

Software has license

DETAILS

- License: 0BSD
- Usage: CI/CD, command-line
- Software Tier: ResearchInfrastructureSoftware



Training materials

← → ↺ 🏠 🌐 everse-training.app.cern.ch/materials ☆



Opnieuw starten om te updaten ⋮



★ Bookmarks



Save to Mendeley



Getting Started



Latest Headlines



Imported From Fir...



Apple



iCloud



Facebook



Twitter



Wikipedia



Alle bookmarks

EVERSE
TRAINING

Materials

Events

Trainers

About



zmzhao ▾

Home / Materials

Sort



Most recent ▾

Filter



Date added ▾



Scientific topic ▾



Content provider ▾



Keyword ▾



Competency level ▾



Licence ▾



Target audience ▾

Training materials

E.g., web pages (github, blog, ...), presentations, videos, tutorials, etc., about training on any RSE-related software, standard or pipeline. Materials can be manually added or automatically harvested from a provider's website.

If you are unsure whether your training material is a good fit for the current page, we warmly welcome you to contact us.

Search materials...



Subscribe ▾

Register training material

118 materials found

Results per page: 10 ▾

GITHUB PAGE

Industry Skills Agile Training

organising software project

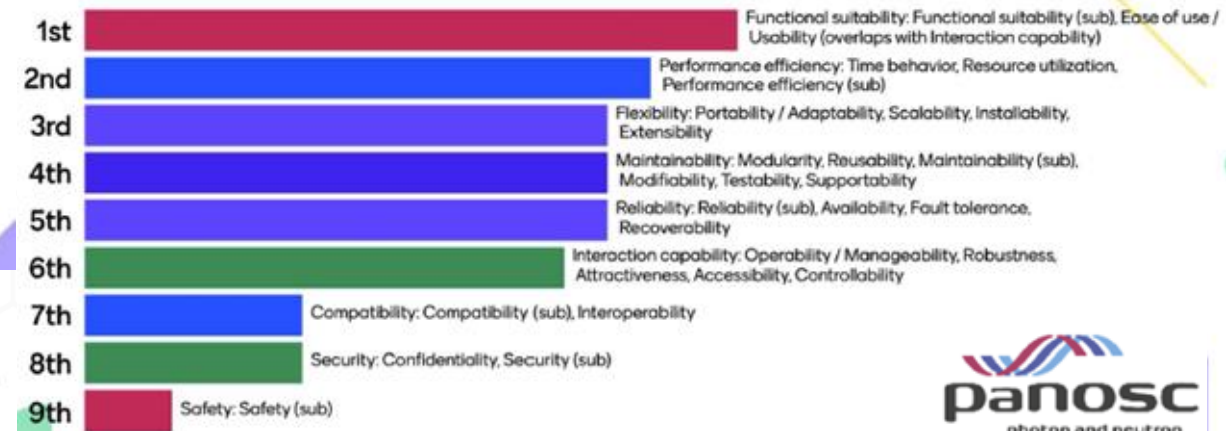
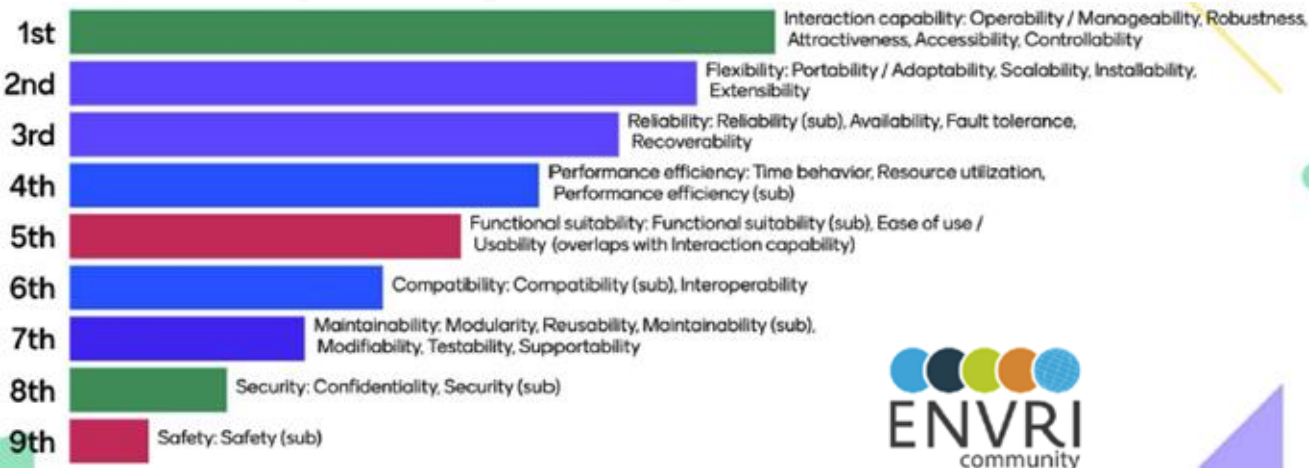


GITHUB PAGE

Ai Fundamentals For Researchers

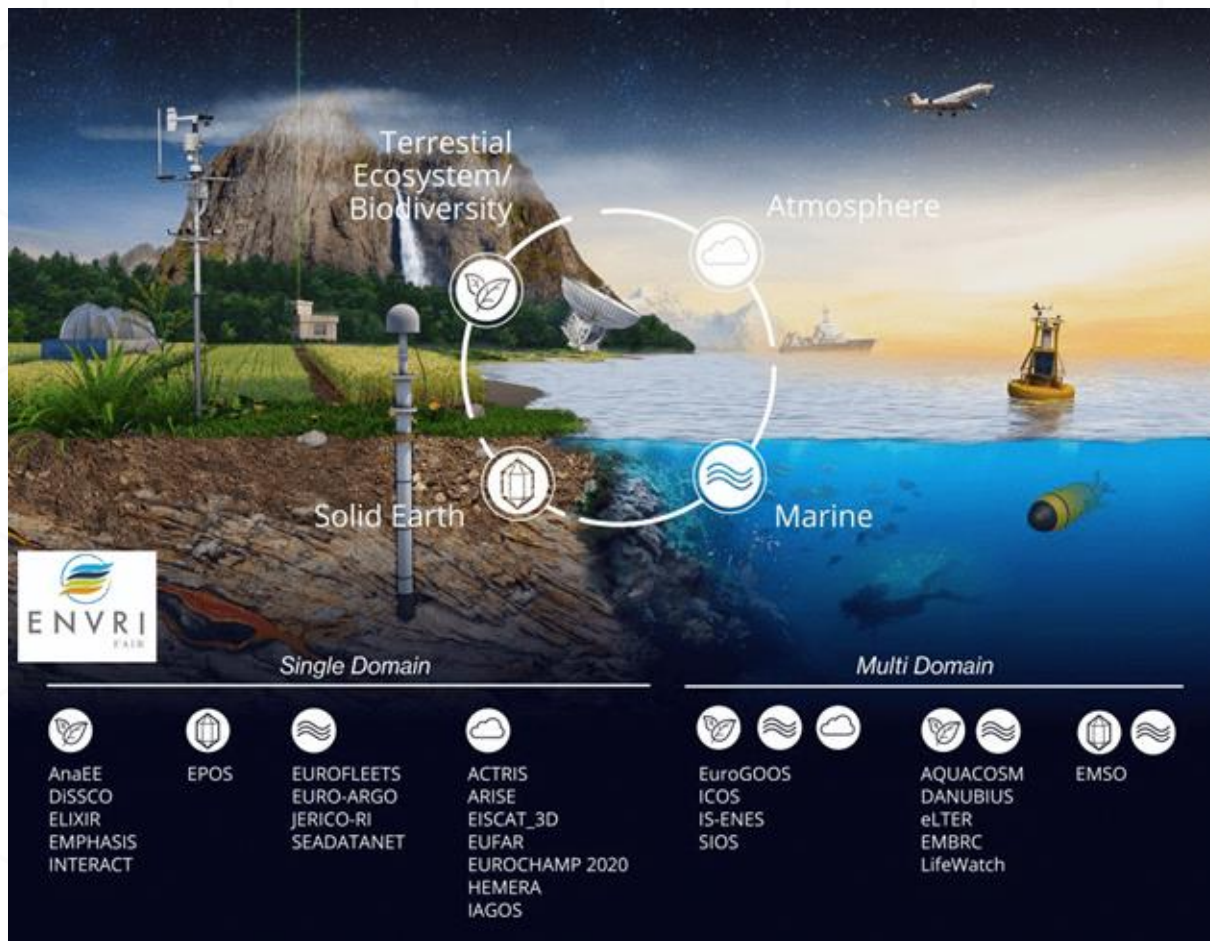


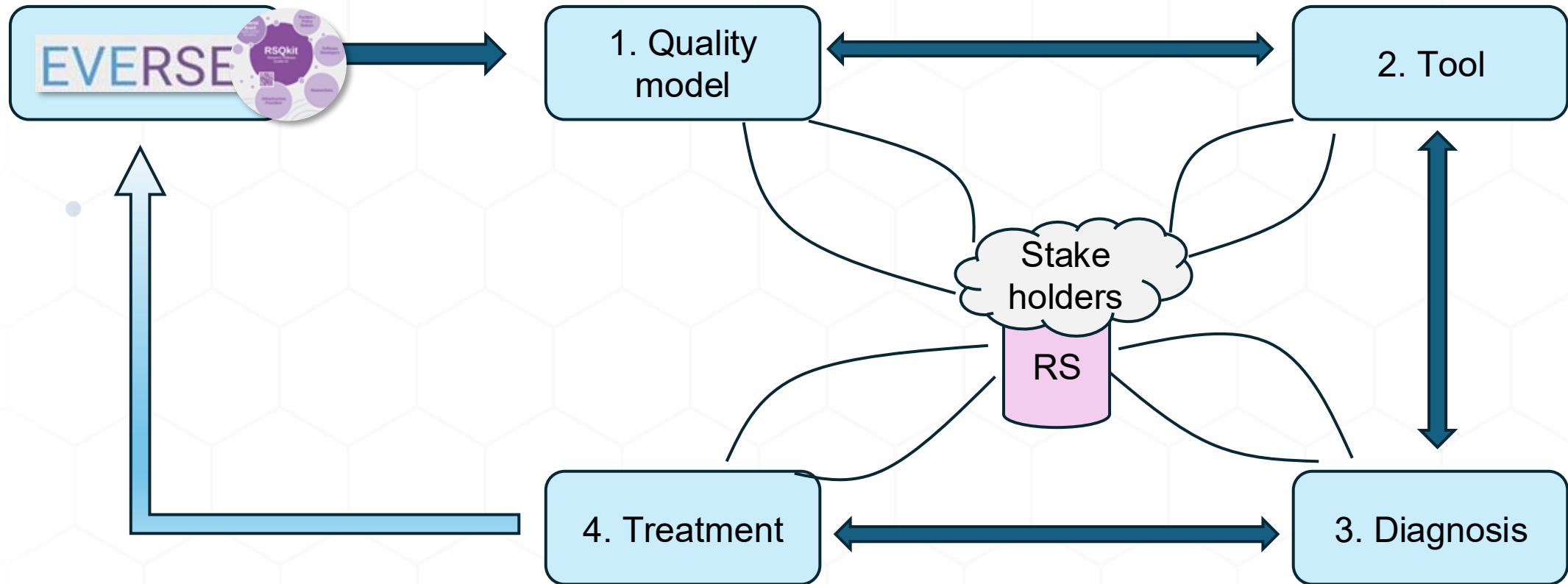
Prioritisation of quality dimensions



4. Case study: Using EVERSE in ENVRI

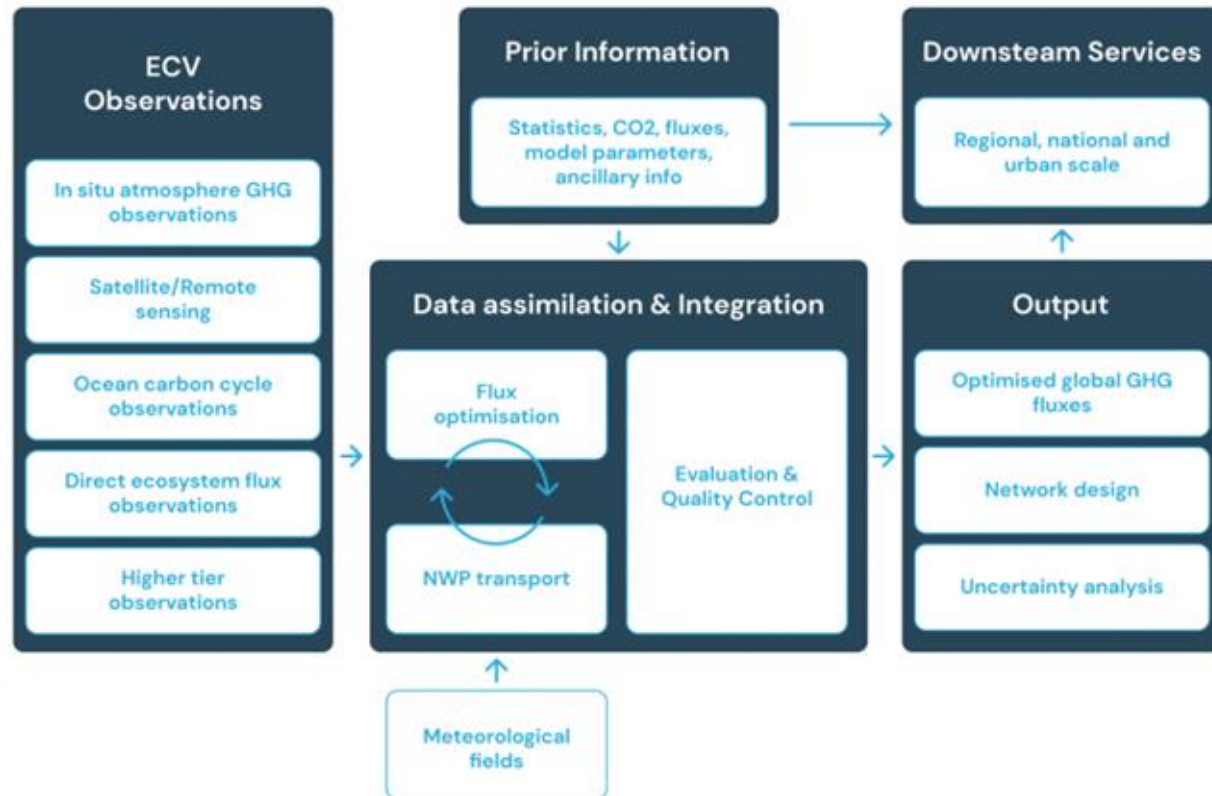
- RIs in environmental and earth sciences





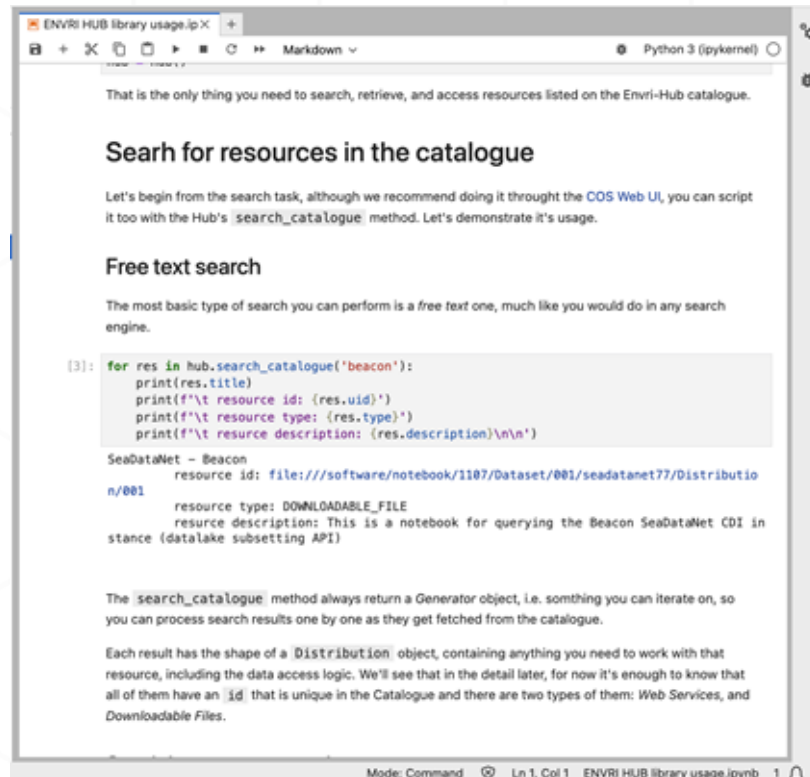
Essential climate variables

- *Essential Climate Variables as defined by the Global Climate Observing System; there are 54 focused on the atmosphere, cryosphere, oceans, anthroposphere, biosphere and the hydrosphere*



Essential climate variables

- Variables are critical for observing and monitoring a given facet of the Earth system.



That is the only thing you need to search, retrieve, and access resources listed on the Envri-Hub catalogue.

Search for resources in the catalogue

Let's begin from the search task, although we recommend doing it through the [COS Web UI](#), you can script it too with the Hub's `search_catalogue` method. Let's demonstrate its usage.

Free text search

The most basic type of search you can perform is a *free text* one, much like you would do in any search engine.

```
[3]: for res in hub.search_catalogue('beacon'):
    print(res.title)
    print(f'\t resource id: {res.uid}')
    print(f'\t resource type: {res.type}')
    print(f'\t resource description: {res.description}\n\n')
```

SeaDataNet - Beacon
 resource id: file:///software/notebook/1107/Dataset/001/seadatanet77/Distribution/001
 resource type: DOWNLOADABLE_FILE
 resource description: This is a notebook for querying the Beacon SeaDataNet CDI in stance (datalake subsetting API)

The `search_catalogue` method always return a Generator object, i.e. something you can iterate on, so you can process search results one by one as they get fetched from the catalogue.

Each result has the shape of a `Distribution` object, containing anything you need to work with that resource, including the data access logic. We'll see that in the detail later, for now it's enough to know that all of them have an `id` that is unique in the Catalogue and there are two types of them: Web Services, and Downloadable Files.

Tier one research software

Sou 1



Sou 2



Sou 3



Pilots & Drivers

**Environmental Sciences:** *Integration of Science Cluster ENVRI through ENVRI-HUB*

- Integrate EVERSE framework into the ENVRI-HUB Knowledgebase and Virtual Research Environment
- Apply to the development of the Essential Climate Variable computing program and cloud workflows

**Life Sciences:** *Integration of Science Cluster EOSC-Life through ELIXIR*

- Make RO-Crate actionable by incorporating the five safes concept into WfExS for secure and federated workflow orchestration
- Use of community-led standards for materialising research software packaged using container technologies and mobilising encrypted data whenever needed

**Astronomy and particle physics:** *Integration of Science Cluster ESCAPE through the Dark Matter Test Science Project*

- ML for scientific data compression (standalone code, python)
- A Common Tracking Software
- Choose an ATLAS trigger algorithm as an option for the collaboration

**Proton and neutron science:** *Integration of Science Cluster PaNOSC through LEAPS/LENS*

Transition software to high performance computing (HPC) and heterogeneous computing architectures

**Social sciences:** *Integration of Science Cluster SSHOC*

Develop a multilanguage textual analysis pipeline of tools that use a combination of open source tools and own code to create an integrated SotA tool capable of deploying locally or as a service



Software quality in VRE

From personal analysis code to shared infrastructure — rising reuse, audience and sustainability demands

T3

Research software infrastructure



Virtual research environment

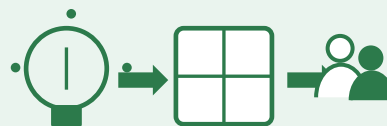


shared foundation · whole community

T2

Prototype tools

Virtual Labs



new idea → shareable tool → others

T1

Analysis code

Jupyter notebooks



data → analysis code → result



each tier builds on the one below

Same code DNA, rising responsibility: as software moves from Tier 1 to Tier 3, its audience widens and the need for identification, credit and sustained engineering grows.



Methodology: three iterative steps.

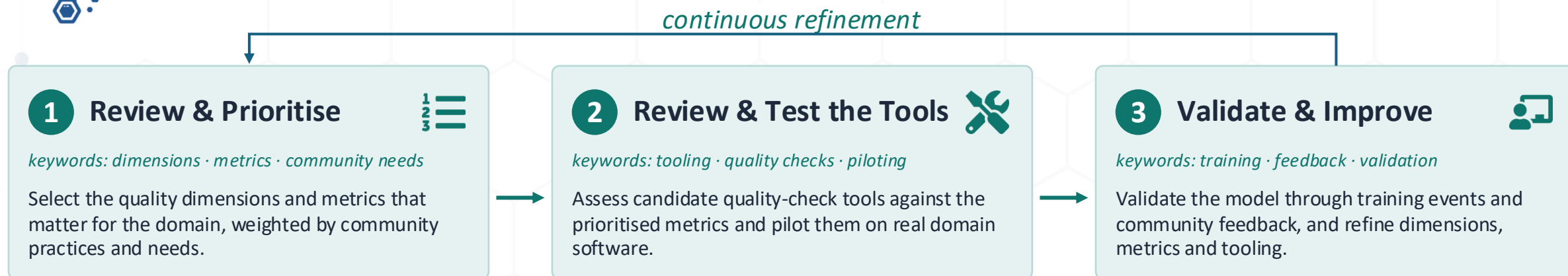
- Quality model customization
- Quality tool selection and integration
- Community training, utilization and feedback



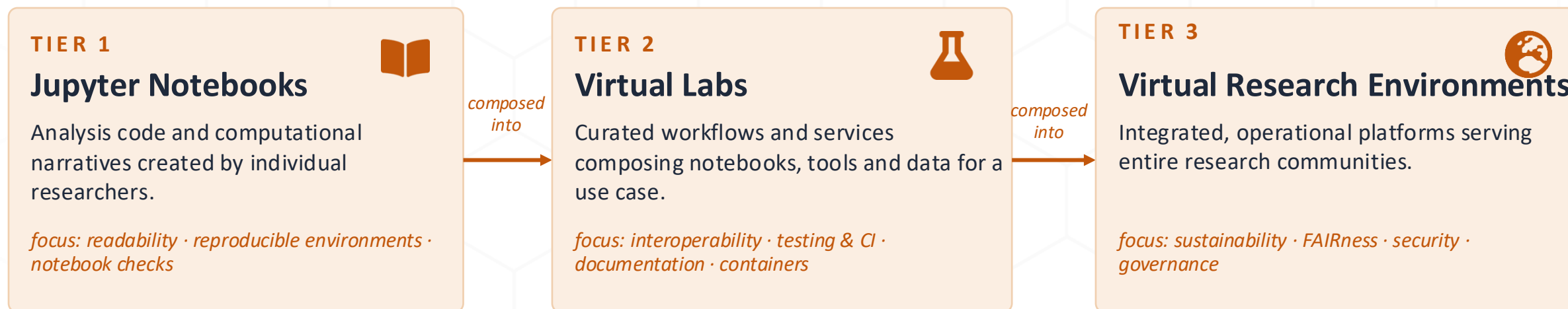
BEEs
2026

Customising the Research Software Quality Model for a Domain

A three-step customisation cycle, instantiated at each of the **three tiers of research software** in the domain



the same three-step cycle is applied at every tier



increasing scale, integration and community involvement

4.1. Quality model customization

5 Dimensions:

1. Maintainability
2. Security
3. FAIRness
4. Functional Suitability
5. Sustainability

25 Final Metrics (selected from Volentir [1] & EVERSE [2])

Examples:

1. Maintainability Index
2. Code Duplication
3. Comment Density
4. Dependency Management
5. Presence of License
6. No Leaked Credentials

Mapped to 4 Lifecycle Stages:

1. Planning & Design
2. Development
3. Testing
4. Maintenance

Quality Dimension	Metric	Lifecycle Stage
Maintainability	Modularity	Planning & Design
	Cohesion	Planning & Design
	Architectural Complexity	Planning & Design
	Code Smells	Development
	Maintainability Index	Development
	Cognitive Complexity	Development
	Cyclomatic Complexity	Development
	Code Duplication	Development
	Technical Debt	Development
	Comment Density	Development
	Software Size (LoC)	Development
Security	Security Vulnerabilities	Maintenance
	No Leaked Private Credentials	Maintenance
FAIRness	Presence of License	Maintenance
	Publicly Accessible Repository	Maintenance
	Rich Metadata	Maintenance
	Documentation Quality	Maintenance
Functional Suitability	Requirement Traceability	Planning & Design
	Unit Tests	Testing
	Test Success Rate	Testing
	Code Reproducibility	Testing
	Defect Rate	Testing
	Percentage of Assertions	Testing
Sustainability	Dependency Management	Development
	User Satisfaction	Maintenance

[1] A. Volentir, "Customizable software quality guidance in the jupyter environment," M.S. thesis, University of Amsterdam, 2024.

[2] EVERSE Project, European virtual institute for research software excellence, <https://everse.software>, Accessed: 2025-04-13, 2025

4.2. Quality check tools

Metric	Tool(s)	Feasibility	Notes
Modularity	None	Manual Review	Structural modularity is difficult to capture without human interpretation.
Cohesion	None	Manual Review	Requires conceptual understanding of logical grouping across cells.
Requirement Traceability	None	Manual Review	Depends on how well code links to research goals.
Architectural Complexity	None	Manual Review	Requires human assessment of overall flow and structure.
Code Smells	Pylint	Automatic	Pylint flags typical Python anti-patterns and poor code practices.
Maintainability Index	Radon	Automatic	Radon computes maintainability score based on complexity and size.
Cognitive Complexity	None	Manual Review	No lightweight tools available for Python.
Cyclomatic Complexity	Radon	Automatic	Measured via Radon using AST-based analysis.
Code Duplication	JSCPD	Automatic	Detects duplicated code fragments in notebooks and scripts.
Technical Debt	None	Indirect	Inferred from code smells and complexity, not measured directly.
Dependency Management	Custom Script	Partial	Compares imports with <code>requirements.txt</code> ; doesn't account for all package sources.

4.3. Integrate tools with VRE

jupyter Untitled Last Checkpoint: last month

File Edit View Run Kernel Settings Help

JupyterLab Python 3 (ipykernel)

[1]: %run extension.py

Stage: Development

Target Path: tools

Run Quality Scan

Selected Stage: Development

Development Metrics Being Checked:

- ✓ **Code Smells** (measured)
Automatically checked via pylint.
- ✓ **Maintainability Index** (measured)
Automatically checked via radon.
- ✗ **Cognitive Complexity** (requires human evaluation)
Not automatically measurable. Requires human judgment to assess nested structures, logic flow, and mental overhead.
- ✓ **Cyclomatic Complexity** (measured)
Automatically checked via radon.
- ✓ **Code Duplication** (measured)
Automatically checked via jscpd.
- ~ **Technical Debt** (partially measured)
Estimated indirectly using indicators like code smells (pylint), cyclomatic complexity (radon), code duplication (jscpd), and maintainability index (radon). These issues often lead to technical debt. While no standard tool calculates technical debt for research notebooks, this approximation gives insight into future refactoring.
- ~ **Dependency Management** (partially measured)
Partially measured by checking whether required libraries are declared in requirements.txt and used in code. Helps detect missing or unused dependencies.
- ✓ **Comment Density** (measured)
Automatically checked via custom parser.

Project-Level Results

✓ Dependency Management

Imported third-party packages found in your code: 2

Packages listed in requirements.txt: 8

Imported packages:

- IPython, ipywidgets

Declared packages:

- IPython, bandit, howfairis, ipywidgets, jscpd, notebook, pylint, radon

✓ Declared and used:

- IPython, ipywidgets

⚠ Declared but not imported:

- bandit, howfairis, jscpd, notebook, pylint, radon

Note: Some packages may be used indirectly (e.g., by CLI tools or notebooks) and won't appear in Python imports.

File: tools/bandit_runner.py

- ✗ **Code Smells:** tools/bandit_runner.py:19:0: C0301: Line too long (104/100) (line-too-long)
tools/bandit_runner.py:40:0: C0301: Line too long (113/100) (line-too-long)
tools/bandit_runner.py:108:0: C0301: Line too long (101/100) (line-too-long)
tools/bandit_runner.py:1:0: C0114: Missing module docstring (missing-module-docstring)
tools/bandit_runner.py:5:0: R0914: Too many local variables (20/15) (too-many-locals)
tools/bandit_runner.py:74:9: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
Your code has been rated at 8.50/10 (previous run: 8.50/10, +0.00)

✓ Maintainability Index: MI Score: 77.54, Grade: A

Very high maintainability.

Tip: Your code demonstrates excellent structure, clarity, and low complexity. To maintain this level, continue enforcing consistent naming conventions, modular design principles, and clean separation of concerns. Ensure all functions remain concise and well-documented. Conduct occasional code reviews to catch early signs of complexity.

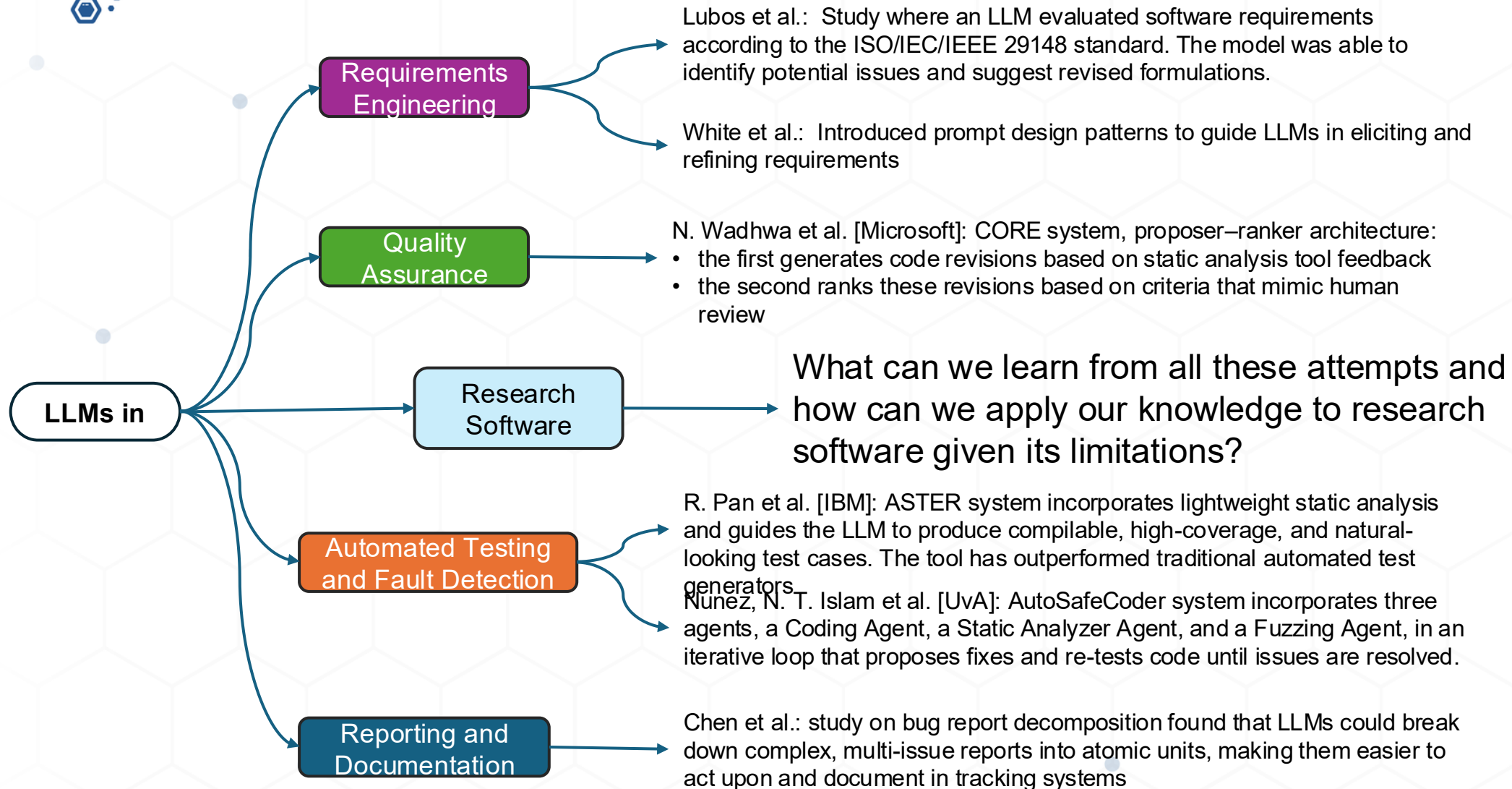
✓ Cyclomatic Complexity: Avg. Cyclomatic Complexity: 11.00, Worst Rank: C

Moderate complexity – slightly complex.

Tip: Consider reviewing areas with nested conditionals, long methods, or large switch/case statements. Break complex logic into helper functions where possible.

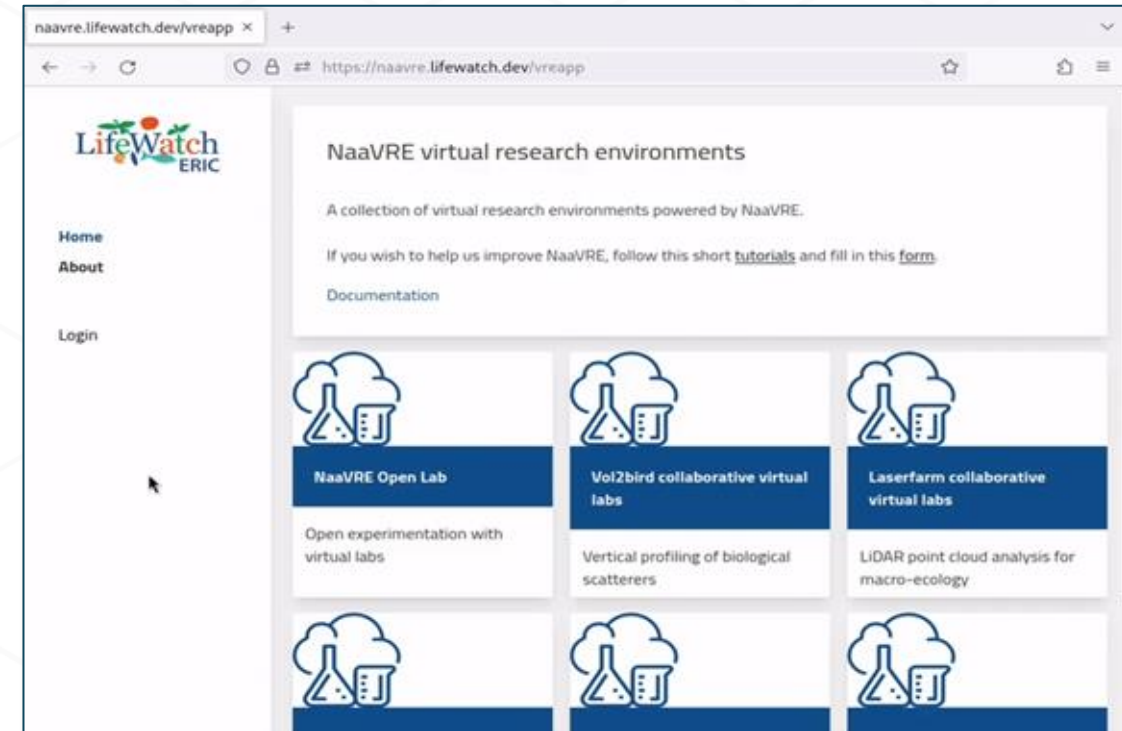
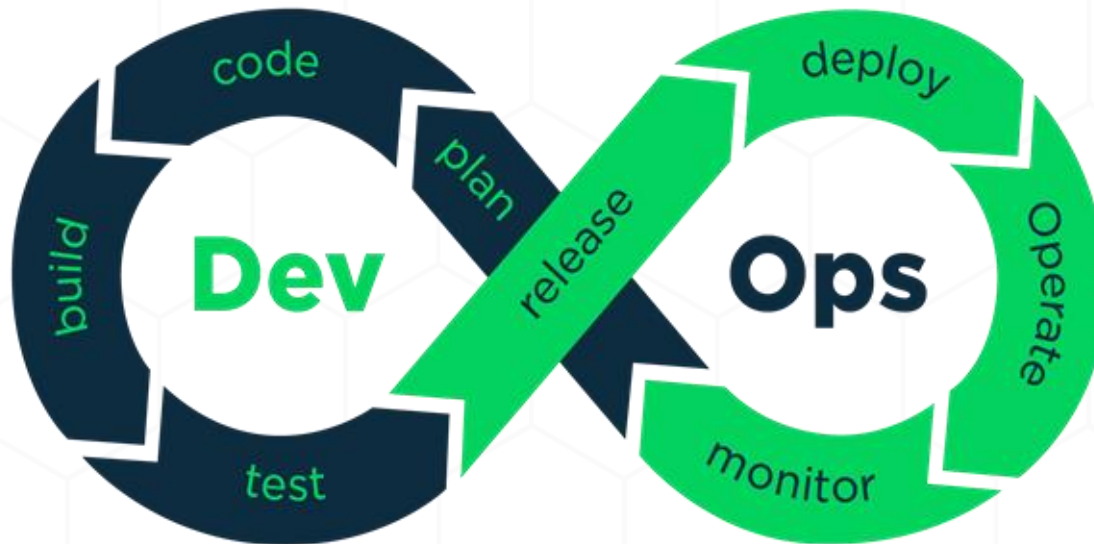
Note: Cyclomatic Complexity ranks range from A (low/simple) to F (very high/complex).

4.4 LLM in research software quality



DevOps pipeline for services (tier three RS)

Research software quality



Initial Mapping of Characteristics

DevOps Phase	Focus	Characteristic	Sub-characteristics
Plan	Ensure all required functionality is captured early through accurate and complete requirements.	Functional Suitability	Functional Completeness, Functional Appropriateness
Code	Implement secure coding practices to protect data and prevent unauthorized access.	Security	Confidentiality, Integrity
	Write analyzable code to make future changes easier.	Maintainability	Modularity, Reusability, Analysability, Modifiability
	Design code to be adaptable for different environments and future requirements.	Flexibility	Adaptability
Build	Ensure the artifacts of the code-base are build and automated testing will be performed to get the code ready for deployment.	Maintainability	Testability
Test	Verify that the system behaves as intended.	Functional Suitability	Functional Appropriateness
	Test the system stability under specified conditions.	Reliability	Faultlessness, Fault Tolerance
Deploy/Release	Ensure deployments are stable and can recover from failure quickly.	Reliability	Availability, Recoverability
	Protect deployments from threats.	Security	Resistance
Operate	Ensure the product remains usable in the live environment.	Interaction Capability	Recognizability, Learnability, Operability, User Error Protection, User Engagement, Inclusivity, User Assistance, Self-descriptiveness
Monitor	Continuously assess system health and uptime to detect and respond to issues.	Reliability	Recoverability, Availability
	Validate that delivered functionality meets expectations in production.	Functional Suitability	Functional Correctness

Different focus per stage

Plan: Requirements checking

Code: Standard code quality metrics

Deploy/Release: Failed deployments

Build: Unused libraries

Test: Coverage and Success

Operate: User Satisfaction

Monitor: Downtime

Characteristics

Defined based on the goal of each phase

Metrics

Defined based on assigned sub-characteristic to each state of the lifecycle



Table with metrics

DevOps Phase	Characteristic	Sub-characteristic	Linked Metric
Plan	Functional Suitability	Functional Completeness	Requirements Completeness
Plan	Functional Suitability	Functional Appropriateness	Requirements Completeness
Plan	Functional Suitability	Functional Completeness	Requirements Volatility
Code	Security	Confidentiality	Secret Detection
Code	Security	Integrity	CVE and CVSS
Code	Maintainability	Modularity	Duplicated Lines Density
Code	Maintainability	Reusability	Duplicated Lines Density
Code	Maintainability	Analysability	Cognitive Complexity and Code Smells
Code	Maintainability	Modifiability	Cyclomatic Complexity
Code	Sustainability	-	Programming Language Impact
Code	Flexibility	Adaptability	Cognitive Complexity
Build	Sustainability	-	Unused Libraries
Build	Maintainability	Testability	Total Coverage
Test	Functional Suitability	Functional Appropriateness	Test Success Density
Test	Reliability	Faultlessness	Test Success Density
Test	Reliability	Fault Tolerance	Test Success Density
Deploy/Release	Reliability	Availability	Change Failure Rate (CFR)
Deploy/Release	Security	Resistance	Secret Detection
Operate	Interaction Capability	Recognizability	User Satisfaction
Operate	Interaction Capability	Learnability	User Satisfaction
Operate	Interaction Capability	Operability	User Satisfaction
Operate	Interaction Capability	User Error Protection	User Satisfaction
Operate	Interaction Capability	User Engagement	User Satisfaction
Operate	Interaction Capability	Inclusivity	User Satisfaction
Operate	Interaction Capability	User Assistance	User Satisfaction
Operate	Interaction Capability	Self-descriptiveness	User Satisfaction
Monitor	Reliability	Recoverability	Mean Time to Recover (MTTR)
Monitor	Functional Suitability	Functional Correctness	Defect Density

Are the metrics suitable for the sub-characteristics in this stage?

12 antwoorden

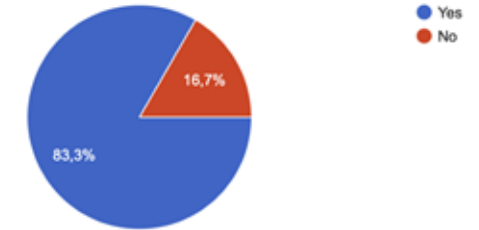


Figure A.6: Planning Metrics answers

Are the metrics suitable for the sub-characteristics in this stage?

12 antwoorden

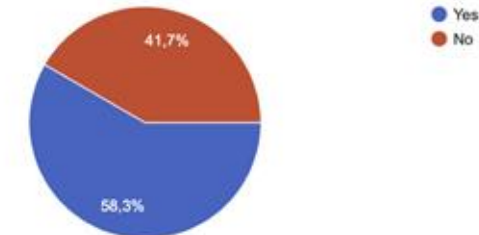


Figure A.8: Code Metrics answers

Are the metrics suitable for the sub-characteristics in this stage?

12 antwoorden

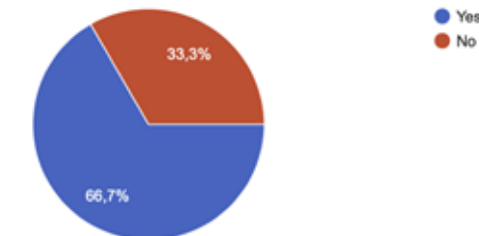


Figure A.17: Monitor Metrics answers

Tools per Metric

Phase	Metric	Tool(s)
Plan	Requirements Completeness	Stakeholder Survey
	Estimated vs Completed Story Points	JIRA REST API
	Security Requirements Coverage	JIRA REST API
Code	Library Freshness	Trivy
	Cyclomatic Complexity	SonarQube
	Cognitive Complexity	SonarQube
	Code Smells	SonarQube
	Duplicated Lines Density	SonarQube
	Programming Language Impact	GitHub API, Couto <i>et al.</i> benchmark table
Build	CVE Identifiers and CVSS Scores	Trivy
	Secret Detection	Gitleaks
	Open Source License Scan	Trivy
	Unused Libraries	Depcheck (JS/TS)
Test	Total Coverage	SonarQube, Jest
	Test Success Density	SonarQube, Jest
	Penetration Testing	OWASP ZAP
Deploy/Release	Deployment Time / Change Time	GitHub API
	Deployment Frequency	GitHub API
Operate/Monitor	Security Incident Reports	JIRA REST API
	Mean Time to Recover (MTTR)	GitHub API
	User Satisfaction	User Survey
	Defect Density	JIRA REST API

Generate Workflow Stage Metrics

DevOps Quality Tool

Tool Selection
Select tools to include in your pipeline

Penetration Testing ✓	Outdated Packages ✓
Unused Libraries ✓	Secret Detection ✓
Test Success Density ✓	SonarQube Metrics ✓
CVE Identifiers and CVSS Scores ✓	Open Source License Scan ✓
Estimated vs Completed Story Points (JIRA) ✓	Security Requirements Coverage (JIRA) ✓
Security Incident Reports (JIRA) ✓	Defect Density (JIRA) ✓
Programming Language Environment Impact ✓	Deployment Frequency ✓
Deployment Time ✓	Mean Time to Recovery ✓

Project Settings [Workflow Instructions](#)

GitHub repository URL
https://github.com/mennokongrunt-test-examples

Branch name that needs to be tested
master

Location of packages (leave empty if root)
Directory path

Local port
8080

Start command
npm run start

Install command
npm install

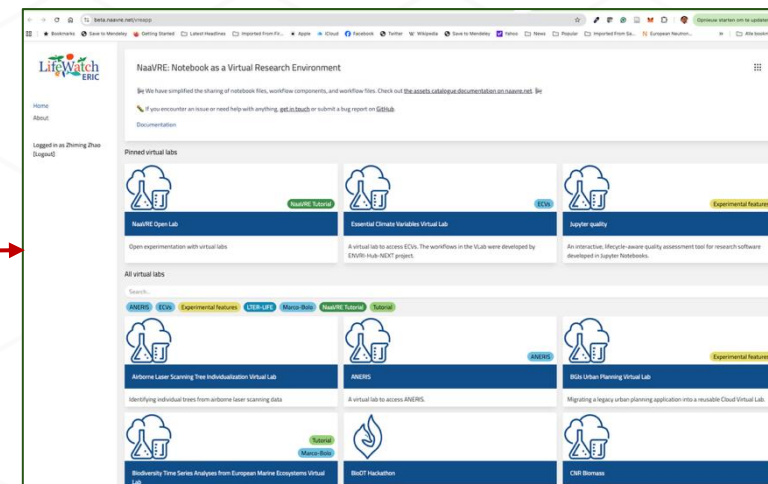
Node.js version
18

Thresholds per Metric

Metric	Explanation	Severity Thresholds
Estimated vs Completed Story Points per Sprint	Based on agile sprint performance standards [55], 85–100% is considered normal. Lower scores indicate planning issues.	$\geq 100\%$: <i>low</i> $85\text{--}99\%$: <i>medium</i> $< 85\%$: <i>high</i>
Security Requirements Coverage	If security requirements are even partially specified, the coverage is considered acceptable.	$> 0\%$: <i>low</i> $= 0\%$: <i>high</i>
Security Incident Reports	The occurrence of any security incident is considered severe.	$= 0$: <i>low</i> > 0 : <i>high</i>
User/Stakeholder Satisfaction (NSI)	Based on the Net Satisfaction Index [56], using a 100-point scale.	≥ 100 : <i>low</i> $75\text{--}99$: <i>low-medium</i> $50\text{--}74$: <i>medium</i> $25\text{--}49$: <i>medium-high</i> < 25 : <i>high</i>
Cyclomatic Complexity	Thresholds follow McCabe's software complexity recommendations [57].	≤ 10 : <i>low</i> $11\text{--}20$: <i>medium</i> > 20 : <i>high</i>
Cognitive Complexity	Campbell suggests 15 as a rough maximum for readability [58].	≤ 15 : <i>low</i> > 15 : <i>high</i>
Code Smells per KLOC	A good baseline is under 5 smells per KLOC and no more than 10 [59].	≤ 5 : <i>low</i> $6\text{--}10$: <i>medium</i> > 10 : <i>high</i>



5. Hands on





How to access the Virtual Lab

beta.naavre.net/vreapp/vi/jupyter-quality

Opnieuw starten om te updaten

LifeWatch ERIC

Jupyter quality

An interactive, lifecycle-aware quality assessment tool for research software developed in Jupyter Notebooks.

Experimental features

Home
About
Login

Jupyter Quality Extension

This tool is designed for research software and helps researchers evaluate notebook quality using relevant metrics based on the software's development stage.

Features

- Assess the quality of Jupyter-based research software projects
- Metrics tailored to different lifecycle stages (e.g., Development, Maintenance)
- Interactive notebook-based interface with pass/fail icons and improvement suggestions
- Automatically converts .ipynb notebooks into .py scripts for metric compatibility
- Modular design for easy extension and maintenance

[Login to see details](#)

New to NaaVRE? Follow the [login tutorial](#)

beta.naavre.net/vreapp

Opnieuw starten om te updaten

LifeWatch ERIC

NaaVRE: Notebook as a Virtual Research Environment

We have simplified the sharing of notebook files, workflow components, and workflow files. Check out the [assets catalogue documentation on naavre.net](#)

If you encounter an issue or need help with anything, [get in touch](#) or submit a bug report on [GitHub](#).

Documentation

Logged in as Zhiming Zhao [Logout]

Pinned virtual labs

NaaVRE Open Lab	Essential Climate Variables Virtual Lab	Jupyter quality
Open experimentation with virtual labs	A virtual lab to access ECVs. The workflows in the VLab were developed by ENVRI-Hub-NEXT project.	An interactive, lifecycle-aware quality assessment tool for research software developed in Jupyter Notebooks.

All virtual labs

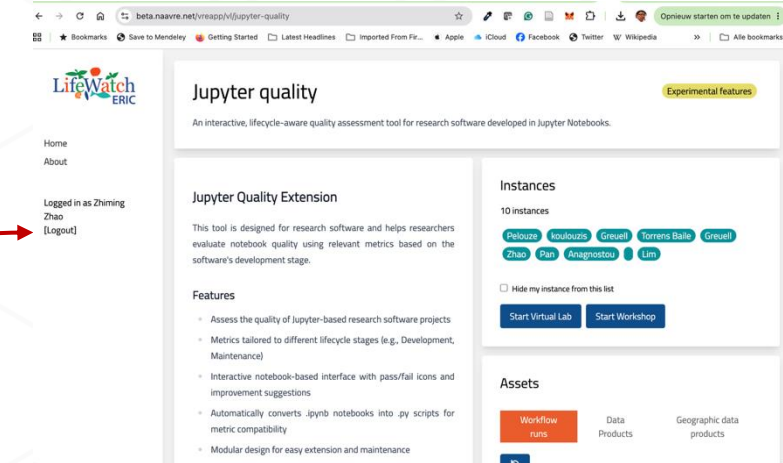
Search...

ANERIS ECVs Experimental features LTER-LIFE Marco-Bolo NaaVRE Tutorial Tutorial

Airborne Laser Scanning Tree Individualization Virtual Lab	ANERIS	BGIS Urban Planning Virtual Lab
Identifying individual trees from airborne laser scanning data	A virtual lab to access ANERIS.	Migrating a legacy urban planning application into a reusable Cloud Virtual Lab.
Biodiversity Time Series Analyses from European Marine Ecosystems Virtual Lab	BioDT Hackathon	CNR Biomass



How to access the Virtual Lab



Jupyter quality

An interactive, lifecycle-aware quality assessment tool for research software developed in Jupyter Notebooks.

Experimental features

[Login to see details](#)

New to NaavRE? Follow the [login tutorial](#)

Jupyter Quality Extension

This tool is designed for research software and helps researchers evaluate notebook quality using relevant metrics based on the software's development stage.

Features

- Assess the quality of Jupyter-based research software projects
- Metrics tailored to different lifecycle stages (e.g., Development, Maintenance)
- Interactive notebook-based interface with pass/fail icons and improvement suggestions
- Automatically converts .ipynb notebooks into .py scripts for metric compatibility
- Modular design for easy extension and maintenance

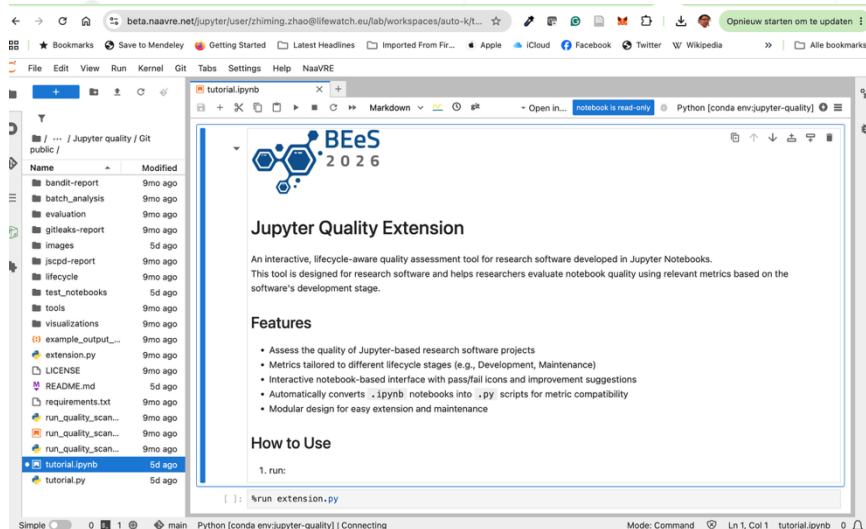
Home

About

Login



How to access the Virtual Lab





Home
About

Logged in as Zhiming Zhao
[Logout]

Jupyter quality

An interactive, lifecycle-aware quality assessment tool for research software developed in Jupyter Notebooks.

Experimental features

Jupyter Quality Extension

This tool is designed for research software and helps researchers evaluate notebook quality using relevant metrics based on the software's development stage.

Features

- Assess the quality of Jupyter-based research software projects
- Metrics tailored to different lifecycle stages (e.g., Development, Maintenance)
- Interactive notebook-based interface with pass/fail icons and improvement suggestions
- Automatically converts .ipynb notebooks into .py scripts for metric compatibility
- Modular design for easy extension and maintenance

Instances

10 instances

Pelouze Koulouzis Greuell Torrens Baile Greuell Zhao Pan Anagnostou Lim

☐ Hide my instance from this list

Start Virtual Lab Start Workshop

Assets

Workflow runs Data Products Geographic data products



How to access the Virtual Lab

beta.naavre.net/jupyter/user/zhiming.zhao@lifewatch.eu/lab/workspaces/auto-k/...

File Edit View Run Kernel Git Tabs Settings Help NaaVRE

... / Jupyter quality / Git public /

Name	Modified
bandit-report	9mo ago
batch_analysis	9mo ago
evaluation	9mo ago
gitleaks-report	9mo ago
images	5d ago
jscpd-report	9mo ago
lifecycle	9mo ago
test_notebooks	5d ago
tools	9mo ago
visualizations	9mo ago
example_output_result...	9mo ago
extension.py	9mo ago
LICENSE	9mo ago
README.md	5d ago
requirements.txt	9mo ago
run_quality_scan...	9mo ago
run_quality_scan...	9mo ago
run_quality_scan...	9mo ago
tutorial.ipynb	5d ago
tutorial.py	5d ago

Jupyter Quality Extension

An interactive, lifecycle-aware quality assessment tool for research software developed in Jupyter Notebooks. This tool is designed for research software and helps researchers evaluate notebook quality using relevant metrics based on the software's development stage.

Features

- Assess the quality of Jupyter-based research software projects
- Metrics tailored to different lifecycle stages (e.g., Development, Maintenance)
- Interactive notebook-based interface with pass/fail icons and improvement suggestions
- Automatically converts .ipynb notebooks into .py scripts for metric compatibility
- Modular design for easy extension and maintenance

How to Use

- run:

```
[ ]: %run extension.py
```

Simple 0 1 main Python [conda env:jupyter-quality] | Connecting Mode: Command Ln 1, Col 1 tutorial.ipynb 0

beta.naavre.net/jupyter/user/zhiming.zhao@lifewatch.eu/lab/workspaces/auto-k/...

File Edit View Run Kernel Git Tabs Settings Help NaaVRE

... / Jupyter quality / Git public /

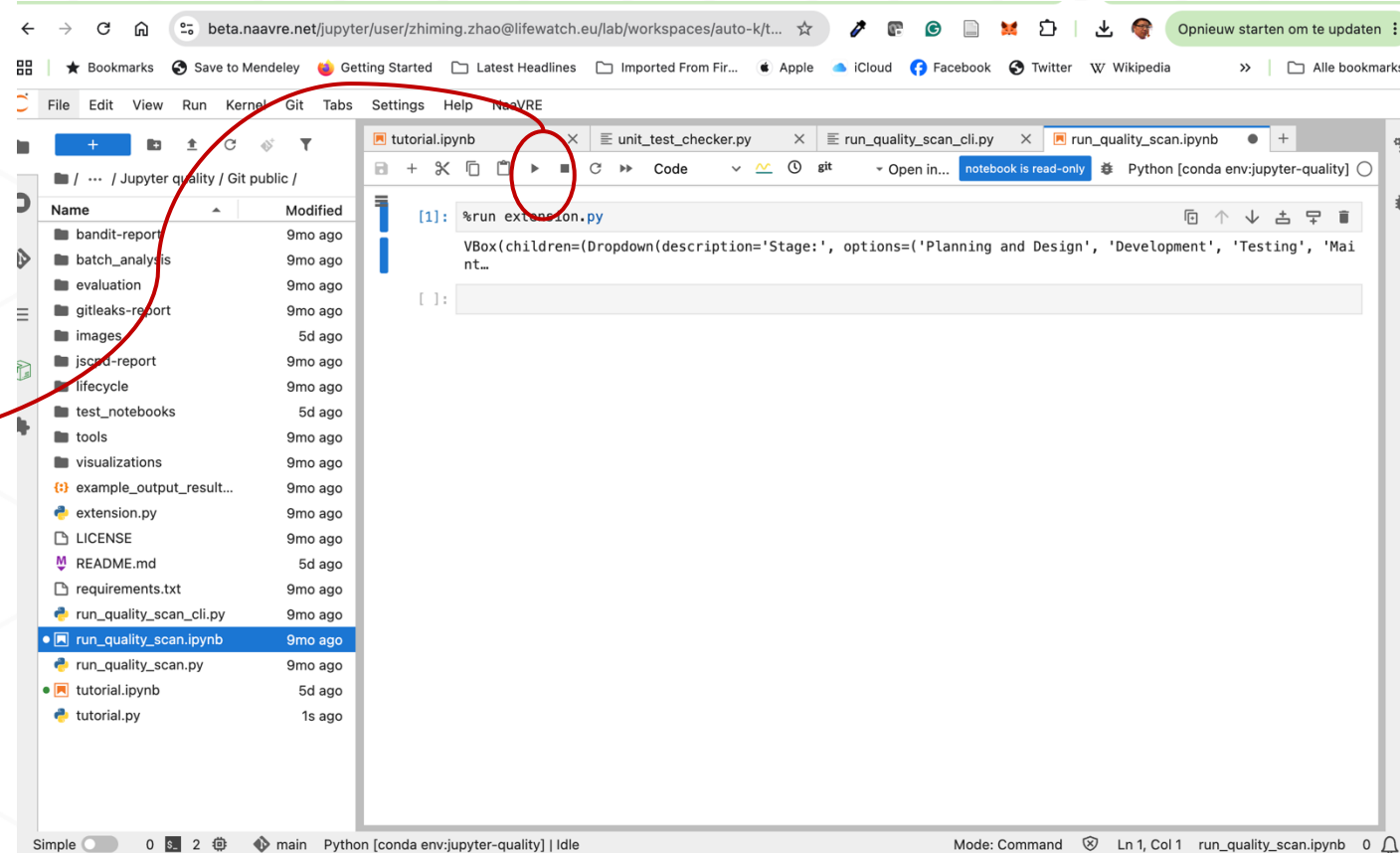
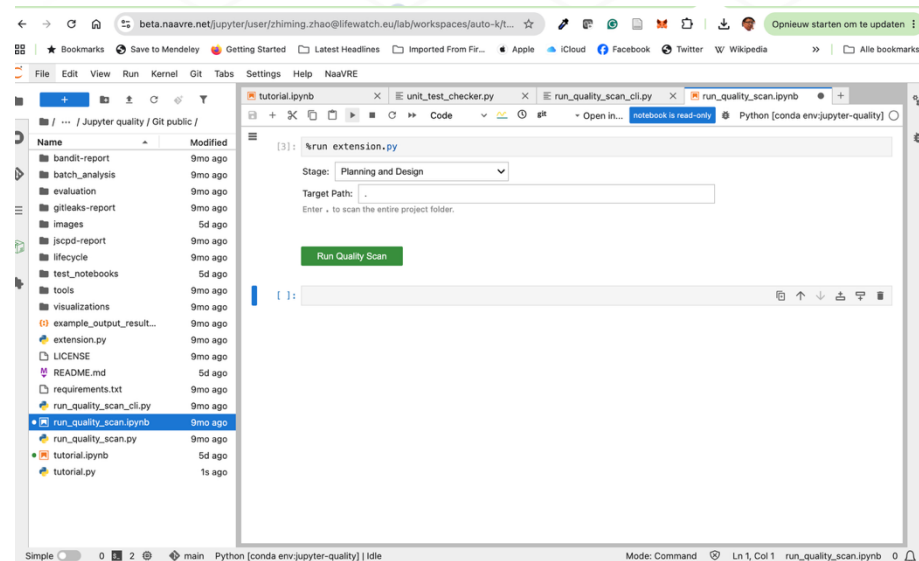
Name	Modified
bandit-report	9mo ago
batch_analysis	9mo ago
evaluation	9mo ago
gitleaks-report	9mo ago
images	5d ago
jscpd-report	9mo ago
lifecycle	9mo ago
test_notebooks	5d ago
tools	9mo ago
visualizations	9mo ago
example_output_result...	9mo ago
extension.py	9mo ago
LICENSE	9mo ago
README.md	5d ago
requirements.txt	9mo ago
run_quality_scan_cli.py	9mo ago
run_quality_scan.ipynb	9mo ago
run_quality_scan.py	9mo ago
tutorial.ipynb	5d ago
tutorial.py	1s ago

```
[ ]: %run extension.py
VBox(children=(Dropdown(description='Stage:', options=('Planning and Design', 'Development', 'Testing', 'Main...))
```

Simple 0 2 0 main Python [conda env:jupyter-quality] | Idle Mode: Command Ln 1, Col 1 run_quality_scan.ipynb 0



How to access the Virtual Lab





How to access the Virtual Lab

beta.naavre.net/jupyter/user/zhiming.zhao@lifewatch.eu/lab/workspaces/auto-k/t...

Opnieuw starten om te updaten

Bookmarks: Save to Mendeley, Getting Started, Latest Headlines, Imported From Fir..., Apple, iCloud, Facebook, Twitter, Wikipedia

File Edit View Run Kernel Git Tabs Settings Help NaaVRE

... / Jupyter quality / Git public /

Name	Modified
bandit-report	9mo ago
batch_analysis	9mo ago
evaluation	9mo ago
gitleaks-report	9mo ago
images	5d ago
jscpd-report	9mo ago
lifecycle	9mo ago
test_notebooks	5d ago
tools	9mo ago
visualizations	9mo ago
example_output_result...	9mo ago
extension.py	9mo ago
LICENSE	9mo ago
README.md	5d ago
requirements.txt	9mo ago
run_quality_scan_cli.py	9mo ago
run_quality_scan.ipynb	9mo ago
run_quality_scan.py	9mo ago
tutorial.ipynb	5d ago
tutorial.py	1s ago

tutorial.ipynb unit_test_checker.py run_quality_scan_cli.py run_quality_scan.ipynb

Code git Open in... notebook is read-on

[3]: %run extension.py

Stage: Planning and Design

Target Path: .

Enter . to scan the entire project folder.

Run Quality Scan

[1]:

Simple 0 2 main Python [conda env:jupyter-quality] | Idle Mode: Command Ln 1, Col 1 run_quality_scan.ipynb 0

%run

Stage: Development

Target:

Enter

Run Quality Scan



How to access the Virtual Lab

beta.naavre.net/jupyter/user/zhiming.zhao@lifewatch.eu/lab/workspaces/auto-k/t...

Opnieuw starten om te updaten

Bookmarks: Save to Mendeley, Getting Started, Latest Headlines, Imported From Fir..., Apple, iCloud, Facebook, Twitter, Wikipedia, Alle bookmarks

File Edit View Run Kernel Git Tabs Settings Help NaaVRE

/ ... / Jupyter quality / Git public /

Name	Modified
bandit-report	9mo ago
batch_analysis	9mo ago
evaluation	9mo ago
gitleaks-report	9mo ago
images	5d ago
jscpd-report	9mo ago
lifecycle	9mo ago
test_notebooks	5d ago
tools	9mo ago
visualizations	9mo ago
example_output_result...	9mo ago
extension.py	9mo ago
LICENSE	9mo ago
README.md	5d ago
requirements.txt	9mo ago
run_quality_scan_cli.py	9mo ago
run_quality_scan.ipynb	9mo ago
run_quality_scan.py	9mo ago
tutorial.ipynb	5d ago
tutorial.py	1s ago

tutorial.ipynb unit_test_checker.py run_quality_scan_cli.py run_quality_scan.ipynb

Code git Open in... notebook is read-only Python [conda env:jupyter-quality]

[3]: %run extension.py

Stage: Planning and Design

Target Path: .

Enter . to scan the entire project folder.

Run Quality Scan

[]:

Target Path: test_notebooks

Simple 0 2 main Python [conda env:jupyter-quality] | Idle Mode: Command Ln 1, Col 1 run_quality_scan.ipynb 0

6. Feedback

- The quality of Research Software (RS) is essential for open science and research infrastructure.
- The EVERSE RSQkit offers a community-based approach to RS quality control.
- Customization of the RS quality model is necessary for specific communities.
- Generative AI (GenAI) shows significant potential for diagnosing RS quality and providing feedback.



Thank you!

Contact: z.zhao@uva.nl

ENVRI: www.envri.eu

EVERSE: <https://everse.software/network/>

MNS Research group: <https://mns-research.nl/>



Funded by
the European Union





To open the workshop in the NaaVRE virtual lab:

- Go to beta.naavre.net
- Click on "Jupyter quality"
- Log in (using a google or EOSC account)
- Click on "Start workshop"