D5.2 - Metrics for automated FAIR software assessment in a disciplinary context

Neil Chue Hong, Elena Breitmoser, Mario Antonioletti, Joy Davidson, Daniel Garijo, Alejandra Gonzalez-Beltran, Morane Gruenpeter, Robert Huber, Clement Jonquet, Mike Priddy, et al.

Project Title	Expanding FAIR solutions across EOSC
Project Acronym	FAIR-IMPACT
Grant Agreement No.	101057344
Start Date of Project	2022-06-01
Duration of Project	36 months
Project Website	https://fair-impact.eu/

D5.2 - Metrics for automated FAIR software assessment in a disciplinary context

Work Package	**WP5 - Metrics, certification, and guidelines**
Lead Author (Org)	**Neil Chue Hong (UEDIN-SSI), Elena Breitmoser (UEDIN-SSI)**
Contributing Author(s) (Org)	**Mario Antonioletti (UEDIN-SSI), Joy Davidson (UEDIN-DCC), Daniel Garijo (UPM), Alejandra Gonzalez-Beltran (UKRI-STFC), Morane Gruenpeter (INRIA), Robert Huber (UBremen), Clement Jonquet (INRAE), Mike Priddy (KNAW-DANS), John Shepherdson (CESSDA), Maaike Verburg (KNAW-DANS), Chris Wood (UEDIN-SSI)**
Due Date	**2023-10-31**
Date	**2023-10-27**
Version	**V1.0**

Dissemination Level	
X	PU: Public
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

Versioning and contribution history

Version	Date	Author	Notes
0.1	2023.05.05	Neil Chue Hong, Elena Breitmoser	TOC and V0.1, main structure
0.2	2023.05.12	Elena Breitmoser, Mike Priddy, Neil Chue Hong, Joy Davidson	Writing sprint 1
0.3	2023.05.26	Alejandra Gonzalez-Beltran, Mike Priddy, Morane Gruenpeter, Elena Breitmoser, Neil Chue Hong, Mario Antonioletti	Writing sprint 2
0.4	2023.07.05	Mario Antonioletti, Neil Chue Hong, Joy Davidson, Daniel Garijo, Robert Huber, Mike Priddy, Morane Gruenpeter, Elena Breitmoser, Chris Wood	Writing sprint 3
0.5	2023.08.01	Mario Antonioletti, Elena Breitmoser, Neil Chue Hong	Draft for internal task reviewers
0.6	2023.09.18	Neil Chue Hong, Elena Breitmoser, Daniel Garijo, Robert Huber, Mike Priddy	Restructured draft for domain reviewers
0.8	2023.09.30	Neil Chue Hong, Elena Breitmoser, Mike Priddy, Maaike Verburg, John Shepherdson	Submission to PCO and internal FAIR-IMPACT review
0.9	2023.10.25	Neil Chue Hong, Elena Breitmoser, Clement Jonquet. Reviewers: Michelle Barker, Yann Le Franc	Final internal review
1.0	2023.10.31	Neil Chue Hong, Elena Breitmoser, Maiike Verburg	Submission to EC

Disclaimer

FAIR-IMPACT has received funding from the European Commission's Horizon Europe funding programme for research and innovation programme under the Grant Agreement no. 101057344. The content of this document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of such content.

Table of Contents

List of Tables

Glossary

Term	Description
API	Application Programming Interface
ARK	Archival Resource Key
CESSDA	Consortium of European Social Science Data Archives
CFF	Citation File Format
CMA	CESSDA software Maturity Assessment
Digital Object	A machine-independent data structure consisting of one or more elements in digital form that can be parsed by different information systems; the structure helps to enable interoperability among diverse information systems in the Internet.
DOI	Digital Object Identifier
EC	European Commission
EOSC	European Open Science Cloud
ESIP	Earth Science Information Partners
FAIR	Findable, Accessible, Interoperable, Reusable
FAIR4RS	FAIR for Research Software
FLOSS	Free/Libre and Open Source Software
Forge (software)	Platform used for the collaborative development and sharing of software (often used as a synonym for code repository)
FRSM	FAIR Research Software Metric
GL	Granularity Level (as defined in Gruenpeter et al. (2021))
GUID	Globally Unique IDentifier (synonymous with UUID)
IETF	Internet Engineering Task Force
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
JSON-LD	JavaScript Object Notation for Linked Data
Licence, Software licence	An agreement between the copyright owner and the end-user on the use and distribution of software
Metadata	Data that define and describe the characteristics of other data, used to improve both business and technical understanding of data and data-related processes. Metadata is also used to describe other digital objects, such as software.
Metric	A set of criteria or conditions that should be met in order to determine the extent to which a principle has been satisfied.
OpenAIRE	Open Access Infrastructure for Research in Europe
ORCID	Open Researcher and Contributor ID
PID	Persistent IDentifier
POM	Project Object Model
PROV	Provenance
RDA	Research Data Alliance

Term	Description
Repository, code/source code/software	The collection of software source code files and associated metadata (such as the history of changes) that constitutes the development history for a piece of software. The term is also sometimes also used to describe a software forge, which is the platform that hosts code repositories to aid collaborative development and sharing.
Repository, scholarly	Digital repository used for depositing, publishing and long term preservation of digital objects, including software.
Research Software	Includes source code files, algorithms, scripts, computational workflows and executables that were created during the research process or for a research purpose. (Gruenpeter et al., 2021) - different from ‘software in research’, may vary between disciplines.
ReSA	Research Software Alliance
REST	Representational state transfer
RRID	Research Resource Identifier
RS	Research Software
RSMD	Research Software Metadata
Scholarly ecosystem	An ecosystem with scholarly repositories where research software may be deposited explicitly, publishers that may link publications with the source code of the associated software, and aggregators that offer researchers a broader view of the available information. (European Commission, 2020)
SKOS	Simple Knowledge Organisation System
SML	Software Maturity Level
Software in research	Software components (e.g. operating systems, libraries, dependencies, packages, scripts, etc.) that are used for research but were not created during or with a clear research intent. (Gruenpeter et al., 2021) - different from ‘Research Software’, may vary between disciplines.
Software	A set of instructions for a computer to execute (often in the form of source code) and associated documentation and data. A type of digital object.
Source code	The version of a piece of software as originally written in a human-readable form (e.g. using a programming language).
SPDX	Software Package Data Exchange
SWHID	Software Heritage Identifiers
TuRTLe	Terse RDF Triple Language
URI	Uniform Resource Identifier
URN	Uniform Resource Name
Use case	A specific situation in which a product or service could potentially be used.
UUID	Universally Unique IDentifier (synonymous with GUID)

Executive Summary

This deliverable from Task 5.2 (FAIR metrics for research software) on *"Metrics for automated FAIR software assessment in a disciplinary context"* is part of Work Package 5 on "Metrics, Certification and Guidelines" within the FAIR-IMPACT project. It builds on the outputs of the RDA/ReSA/FORCE11 *FAIR for Research Software WG* and existing guidelines and metrics for research software to define metrics for the assessment of the "FAIR Principles for Research Software (FAIR4RS Principles)". *FAIR software* can be defined as research software which adheres to these principles, and the extent to which a principle has been satisfied can be measured against the criteria in a metric. This work on software metrics was coordinated with Task 4.3 (Standard metadata for research software) from Work Package 4 on "Metadata and Ontologies", which focuses on *"Guidelines for recommended metadata standard for research software within EOSC",* to ensure that metrics are related to their recommended metadata properties.

The deliverable defines 17 metrics that can be used to automate the assessment of research software against the FAIR4RS Principles, and provides examples of how these might be implemented in one exemplar disciplinary context of the social sciences. The FAIR-IMPACT project will then work to implement the metrics as practical tests by extending existing assessment tools such as F-UJI; this work will be reported in Q2 2024. Feedback will be sought from the community, through webinars and an open request for comments. The information from all these sources will be used to publish a revised version of the metrics.

1 Introduction

The overall goal of FAIR-IMPACT is to identify practices, policies, tools and technical specifications to guide researchers, repository managers, research performing organisations, policy makers and citizen scientists towards a FAIR data management cycle. The focus of the project is on persistent identifiers (PIDs), metadata, ontologies, metrics, certification and interoperability, applying these to real-life use cases starting with examples from social sciences and humanities, the photon and neutron sciences, life sciences and agri-food and environmental sciences.

While the FAIR principles, originally defined by Wilkinson et al. (2016) as the FAIR Data Principles, may be applied to any digital objects, this deliverable is concerned with the subset of digital objects represented by research software. The RDA/ReSA/FORCE11 *FAIR for Research Software WG*[1] provides a definition of research software that is used in this deliverable:

"Research Software includes source code files, algorithms, scripts, computational workflows and executables that were created during the research process or for a research purpose. Software components (e.g., operating systems, libraries, dependencies, packages, scripts, etc.) that are used for research but were not created during or with a clear research intent should be considered software in research and not Research Software. This differentiation may vary between disciplines." (Gruenpeter et al., 2021)

Software quality has long been discussed in scientific literature (e.g. Kan, 2002, Zuser et al., 2005). Standards for software code quality such as the ISO/IEC Systems and software Quality Requirements and Evaluation (SQuaRE) (ISO, 2011) and the IEEE Computer Society's Software Engineering Body of Knowledge (SWEBoK) (Bourque and Fairley, 2014) discuss metrics for software that are related with the FAIR principles (e.g. usability). While some of these metrics overlap with the FAIR principles, they are mostly targeted towards the industrial development and applications of software code.

The open source software community has also developed guidance and metrics for assessing software. The Community Health Analytics in Open Source Software (CHAOSS) initiative[2] is a Linux Foundation project focused on creating metrics and metrics models,[3] as well as software tools,[4] to better understand the open source community health on a global scale. Some metrics can be measured directly by the tools, but others may require manual assessment. The Open Source Security Foundation[5] has developed a set of best practices

[1] https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg
[2] Community Health Analytics in Open Source Software: https://chaoss.community/
[3] CHAOSS Metrics: https://chaoss.community/kb-metrics-and-metrics-models/
[4] CHAOSS Software: https://chaoss.community/software/
[5] Open Source Security Foundation: https://openssf.org/

applicable to all Free/Libre and Open Source Software (FLOSS) projects and released these in the form of a checklist of criteria[6] and badging that encompass different levels of practice. In this case, each metric corresponds to a different check, which is assessed manually by the developers aiming to obtain the badge.

With the extended application of the FAIR principles to research software in "FAIR Principles for Research Software version 1.0 (FAIR4RS Principles v1.0)" (Chue Hong et al., 2022, Barker et al., 2022), a number of guidelines and best practices have been developed by the community to promote their adoption (Gruenpeter *et al.*, 2023; Martinez *et al.*, 2019). In parallel, the Horizon 2020 EOSC Synergy project has developed software quality guidelines for projects in the European research ecosystem (Ortiz et al, 2022) which include relevant metrics: for example, software documentation should be version controlled, have a PID and provide a licence. Together, these provide the foundation for metrics that can be used to automate the assessment of research software against the FAIR4RS Principles.

Metadata-based assessment approaches have also been proposed for other FAIR digital objects, such as ontologies and semantic resources (Amdouni, et al., 2022).

1.1 Purpose and scope

To increase the adoption and uptake of the FAIR principles, this deliverable presents 17 metrics that can be used to translate the FAIR guiding principles into practical tests to measure the *FAIRness* of research software, that can be implemented in an automated fashion via assessment tools for the different infrastructures in the scholarly ecosystem (software aggregators, software publishers, scholarly repositories and software archives).

The metrics are developed to be domain-agnostic, and take into account characteristics which are specific to research software such as its executability, its composite nature and its continuous evolution and versioning. Though most of the FAIR4RS Principles (summarised in Table 1) can be turned into a measurable metric, some (e.g. "F2. Software is described with rich metadata") are much harder to quantify, and hence be assessed by any automated tool in the future. In these cases, it may only be possible to test for existence rather than quality or correctness. Others, such as "R3. Software meets domain-relevant community standards" can be seen to apply to many metrics, and the implementation of a metric will reference these community standards.

These metrics have been developed through reference to existing work on FAIR metrics, software metrics and software metadata. This included the EOSC minimum metadata properties for datasets[7] (Asmi et al., 2017) and the FAIR-IMPACT Deliverable 4.4 *"Guidelines*

[6] https://bestpractices.coreinfrastructure.org/en/criteria
[7] https://eosc-edmi.github.io/

for recommended metadata standard for research software within EOSC" (Gruenpeter et al., 2023) developed by Task 4.3. Wherever feasible, existing metrics and indicators that are currently being used to evaluate the FAIRness of digital objects are reused, such as those defined in *"FAIRsFAIR Data Object Assessment Metrics"* (Devaraju et al, 2022) and *"FAIRsFAIR M2.15 Assessment Report On 'FAIRness of software'"* (Gruenpeter et al., 2020). Community input included a workshop[8] at RDA Plenary 20 in Gothenburg in March 2023 which collected use cases and metrics from participants (Chue Hong et al., 2023).

The FAIR Principles for Research Software (FAIR4RS Principles) are:

Table 1 - The FAIR Principles for Research Software (from Table 1 in Chue Hong et al., 2022)

F: Software, and its associated metadata, is easy for both humans and machines to find.
F1. Software is assigned a globally unique and persistent identifier. ● 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. F2. Software is described with rich metadata. F3. Metadata clearly and explicitly include the identifier of the software they describe. F4. Metadata are FAIR, searchable and indexable.
A: Software, and its metadata, is retrievable via standardized protocols.
A1. Software is retrievable by its identifier using a standardized communications protocol. ● A1.1. The protocol is open, free, and universally implementable. ● A1.2. The protocol allows for an authentication and authorization procedure, where necessary. A2. Metadata are accessible, even when the software is no longer available.
I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.
I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards. I2. Software includes qualified references to other objects.
R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).
R1. Software is described with a plurality of accurate and relevant attributes. ● R1.1. Software is given a clear and accessible license. ● R1.2. Software is associated with detailed provenance. R2. Software includes qualified references to other software. R3. Software meets domain-relevant community standards.

The evolution of the principles from data (Wilkinson et al., 2016) to software can be found in Appendix B of the *"FAIR Principles for Research Software (FAIR4RS Principles) v1.0"* (Chue Hong et al., 2022) and are also presented in Appendix A of this document.

[8] https://fair-impact.eu/events/fairimpact-events/research-software-workshop-guidelines-and-metrics-metadata-curation

FAIRassist[9] is a resource which catalogues resources to measure and improve FAIRness, including automated assessment tools. Some existing FAIR assessment tools can be run against code repositories, e.g. FAIR-Enough,[10] F-UJI (Devaraju and Huber, 2021) and FAIR-Checker (Gaignard et al., 2023), though most were developed to assess FAIRness of data; when used for software they only assess the associated metadata and identifier. One exception is howfairis,[11] which assesses software but against the fair-software.eu recommendations[12] rather than the FAIR principles directly. Typically these tools assess F and A, along with R1.1 (licence) as these are the easiest to automate. Additionally, software quality assessment tools such as SQAaaS[13] provide pipelines that can be integrated with projects to cross-check relevant quality criteria. A more comprehensive evaluation of these tools is in progress and will be reported in MS5.6 *"Practical tests for automated FAIR software assessment in a disciplinary context"*.

1.2 Metric Outline

In general, a distinction can be made between metrics that apply at the "code level" (which measure aspects of the source code), "software project level" (which measure aspects of how the software is developed) and at the "repository" level (which measure aspects of how the software is stored). Some metrics at the repository level cannot be tested at the software level and vice versa. Some metrics related to reuse or reproducibility may require to be applied at multiple levels. Likewise, there are differences between code repositories (also known as forges) and preservation repositories.

In Willkinson et al., (2018), a focus group formed of some of the authors of the original FAIR principles suggest that a good FAIR metric should be:

- *Clear: anyone can understand the purpose of the metric;*
- *Realistic: it should not be unduly complicated for a resource to comply with the metric*
- *Discriminating: the metric should measure something important for FAIRness; distinguish the degree to which that resource meets that objective; and be able to provide instruction as to what would maximise that value;*
- *Measurable: the assessment can be made in an objective, quantitative, machine-interpretable, scalable and reproducible manner, ensuring transparency of what is being measured, and how;*
- *Universal: The metric should be applicable to all digital resources.*

[9] https://fairassist.org/
[10] https://metrics.api.fair-enough.semanticscience.org/docs
[11] https://github.com/fair-software/howfairis
[12] https://fair-software.eu/
[13] https://sqaaas.eosc-synergy.eu/

The last of these criteria suggest that FAIR metrics primarily refer to repository level metrics, for instance to check the presence of metadata, as many code level metrics are necessarily applicable only to source code resources and software project level metrics are defined around the production of a particular type of resource.

There is not a single implementation of a metric that will work for all research software, but there are metrics that can be applied to all types of software. For example, including metadata to describe the hardware requirements may be important for some applications but not other, and may be expressed differently for a software library designed to be recompiled for different architectures. However, if Universal is redefined to mean “the metric should be applicable to all software resources” a framework of metrics can be created for research software that tests the FAIRness of software by using more specific, detailed metrics for some of the FAIR4RS principles which require additional guidance to implement.

The metrics presented in the next sections are specified following the template (Table 2), modified from Wilkinson et al. (2018) and Devaraju et al. (2022). In each metric table, the descriptions and assessment details of the metric are provided, and its alignment with the relevant FAIR4RS principle and Research Software Metadata recommendation (Gruenpeter et al., 2023). There is an expectation that while the metric and assessment methods will remain the same, the criteria for each compliance level will change as adoption of the FAIR principles increases and infrastructure, tools and guidance improve: what is considered essential should reflect an achievable level of compliance at the current time. The list of proposed FAIR metrics for research software is summarised in Table 3.

Table 2 - Modified FAIR Metric template for FAIR Research Software.

Field	Description
Metric Identifier	The local identifier of the metric (FRSM-XX) FRSM: FAIR Research Software Metric.
Metric Name	Metric name in a human readable form.
Description	The definition of the metric, including examples.
FAIR4RS Principle	The FAIR4RS principle(s) most related to the metric.
RSMD Recommendation	The FAIR-IMPACT RSMD recommendation(s) most related to the metric
Assessment	Requirements and methods to perform the assessment against the metric. This includes a suggested compliance level (essential / important / useful), based on the concepts introduced by the FAIR Data Maturity Model Working Group (2020). Criteria at each level will change as adoption of FAIR increases.
Comments	Further notes associated with the implementation of the metric, which may include related resources, constraints and limitations.

Table 3 - List of FAIR Research Software Metrics

Identifier	Name
FRSM-01	Does the software have a globally unique and persistent identifier?
FRSM-02	Do the different components of the software have their own identifiers?
FRSM-03	Does each version of the software have a unique identifier?
FRSM-04	Does the software include descriptive metadata which helps define its purpose?
FRSM-05	Does the software include development metadata which helps define its status?
FRSM-06	Does the software include metadata about the contributors and their roles?
FRSM-07	Does the software metadata include the identifier for the software?
FRSM-08	Does the software have a publicly available, openly accessible and persistent metadata record?
FRSM-09	Is the software developed in a code repository / forge that uses standard communications protocols?
FRSM-10	Are the formats used by the data consumed or produced by the software open and a reference provided to the format?
FRSM-11	Does the software use open APIs that support machine-readable interface definition?
FRSM-12	Does the software provide references to other objects that support its use?
FRSM-13	Does the software describe what is required to use it?
FRSM-14	Does the software come with test cases to demonstrate it is working?
FRSM-15	Does the software source code include licensing information for the software and any bundled external software?
FRSM-16	Does the software metadata record include licensing information?
FRSM-17	Does the software include provenance information that describe the development of the software?

The FAIR Impact project will work to implement the metrics as practical tests by extending existing assessment tools such as F-UJI; this work will be reported in Q2 2024. Feedback will be sought from the community, through webinars and an open request for comments. The information from all these sources will be used to publish a revised version of the metrics.

2. Metric Specification

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Metric Identifier</td><td colspan="2">FRSM-01</td></tr>
<tr><td>Metric Name</td><td colspan="2">Does the software have a globally unique and persistent identifier?</td></tr>
<tr><td>Description</td><td colspan="2">A software object may be assigned with a globally unique identifier such that it can be referenced unambiguously by humans or machines. Globally unique means an identifier should be associated with only one resource at any time. Examples of unique identifiers of data used for software include: Digital Object Identifier (DOI), the Handle System, Uniform Resource Identifier (URI) such as URL and URN, and Software Heritage Identifiers (SWHID). A data repository may assign a globally unique identifier to your data or metadata when you publish and make it available through its curation service.</td></tr>
<tr><td>FAIR4RS Principle</td><td colspan="2">F1: Software is assigned a globally unique and persistent identifier.
R3: Software meets domain-relevant community standards.</td></tr>
<tr><td>RSMD Recommendation</td><td colspan="2">RSMD-3.3</td></tr>
<tr><td rowspan="5">Assessment</td><td>Requirements</td><td>☐ Software identifier
☐ List of globally unique identifier schemes</td></tr>
<tr><td>Method</td><td>Check if the software identifier is based on a suitable identifier scheme, and test it can be resolved.</td></tr>
<tr><td>Essential</td><td>Software has a human and machine-readable unique identifier that is resolvable to a machine-readable landing page and follows a defined unique identifier syntax.</td></tr>
<tr><td>Important</td><td>Identifier uses an identifier scheme that guarantees globally uniqueness and persistence.</td></tr>
<tr><td>Useful</td><td>Identifier scheme is commonly used in the domain.</td></tr>
<tr><td>Comments</td><td colspan="2">The type of identifier assigned will often depend on the type of repository that the software is deposited in, for example a URL for GitHub, DOI for Zenodo, or SWHID for Software Heritage. Note that URLs are not guaranteed to be persistent and by default GitHub only provides permalinks by request.[14] It is not practical to directly test the global uniqueness or persistence of any individual identifier, therefore this metric proposes testing for an identifier scheme that provides guarantees of persistence.

The suitability of an identifier scheme may depend on the domain. If software metadata is available as a separate record, this should be FAIR (see FRSM-08).</td></tr>
</table>

[14] https://docs.github.com/en/repositories/working-with-files/using-files/getting-permanent-links-to-files

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Metric Identifier</td><td colspan="2">FRSM-02</td></tr>
<tr><td>Metric Name</td><td colspan="2">Do the different components of the software have their own identifiers?</td></tr>
<tr><td>Description</td><td colspan="2">Conceptually, it is useful for identifiers to be assigned at a more granular level than just the software project (often synonymous with the “software concept” or “software project”). For instance a software product may consist of different modules, which in turn may be implemented by different files. This metric tests that these different components are not all assigned the same identifier, and that the relationship between components is embodied in the identifier metadata.</td></tr>
<tr><td>FAIR4RS Principle</td><td colspan="2">F1: Software is assigned a globally unique and persistent identifier.
F1.1: Components of the software representing levels of granularity are assigned distinct identifiers.</td></tr>
<tr><td>RSMD Recommendation</td><td colspan="2">RSMD-3.2, RSMD-3.3, RSMD-3.5</td></tr>
<tr><td rowspan="5">Assessment</td><td>Requirements</td><td>☐ Software identifiers</td></tr>
<tr><td>Method</td><td>Check if each software identifier resolves to the appropriate software component and examine identifier metadata.</td></tr>
<tr><td>Essential</td><td>Where the “software” consists of multiple distinct components, each component has a distinct identifier.</td></tr>
<tr><td>Important</td><td>The relationship between components is embodied in the identifier metadata</td></tr>
<tr><td>Useful</td><td>Every component to granularity level GL3 (module) has its own unique identifier</td></tr>
<tr><td>Comments</td><td colspan="2">The granularity levels for software have been defined by the RDA Software Source Code Identifiers WG in Gruenpeter et al. (2021). Identifiers for each software component should be globally unique and persistent (as tested by FRSM-01).

This metric should not be confused with FRSM-10 and FRSM-12 (related to I2) which checks that other related non-software objects are properly described and FRSM-13 (related to R2) which checks that software dependencies which are not considered a part of the software concept of product are described.</td></tr>
</table>

Field	Description
Metric Identifier	FRSM-03
Metric Name	Does each version of the software have a unique identifier?
Description	To make different versions of the same software (or component) findable, each version needs to be assigned a different identifier. The relationship between versions is embodied in the associated metadata.
FAIR4RS Principle	F1: Software is assigned a globally unique and persistent identifier. F1.2: Different versions of the software are assigned distinct identifiers.

	R3: Software meets domain-relevant community standards.	
RSMD Recommendation	RSMD-3.2, RSMD-3.3, RSMD-3.4	
Assessment	*Requirements*	☐ Software Identifiers
	Method	Check if each software identifier resolves to a different version and examine identifier metadata.
	Essential	Each version of the software has a different identifier.
	Important	Relations between the versions are included in the identifier metadata.
	Useful	The version number is included in the identifier metadata.
Comments	What is considered a "version" is defined by the owner of the software: in many cases this will be something that the owner wants to specifically identify and use and/or "release" or "publish" so that others can use and reference/cite. This is something for which there may be disciplinary norms, which may be documented in domain-specific software guidelines e.g. ESIP Software Guidelines[15] in the earth sciences and CESSDA Software Development Guidelines in the social sciences.[16] Identifiers for each software version should be globally unique and persistent (as tested by FRSM-01) and use the same identifier scheme. It may be useful to reference these identifiers in any release documentation or CHANGELOG.	

Field	Description	
Metric Identifier	FRSM-04	
Metric Name	Does the software include descriptive metadata which helps define its purpose?	
Description	Software requires descriptive metadata to support indexing, search and discoverability.	
FAIR4RS Principle	F2: Software is described with rich metadata. R1: Software is described with a plurality of accurate and relevant attributes. R3: Software meets domain-relevant community standards.	
RSMD Recommendation	RSMD-1.1, RSMD-4.1, RSMD-4.2, RSMD-4.3, RSMD-4.4	
Assessment	*Requirements*	☐ Software source code ☐ Software identifier
	Method	Check if the software and/or software identifier has machine-readable descriptive metadata associated with it that describe its purpose.
	Essential	The software includes a README or other file which includes the software title and description.
	Important	The software includes other descriptive metadata such as domain, funder, programming language, date created, and keywords.

[15] https://esipfed.github.io/Software-Assessment-Guidelines/
[16] https://docs.tech.cessda.eu/software/index.html

<table>
<tr><td></td><td>Useful</td><td>The metadata is contained in a format such as CodeMeta or ProjectObjectModel that enables full machine actionability.</td></tr>
<tr><td>Comments</td><td colspan="2">There are several common places for descriptive metadata to be found, including intrinsic metadata that is part of the software source code such as README files, requirements files that describe dependencies, POM, CodeMeta or CFF files, or in the extrinsic metadata available through resolving the software identifier. It may also be directly embedded in software source code files. The implementation of this metric will depend on the coding standards for the programming language as well as community norms for which descriptive metadata is used.
It is hard to check the relevance / correctness of unstructured metadata such as a text description, but it is possible to automatically check for existence.</td></tr>
</table>

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Metric Identifier</td><td colspan="2">FRSM-05</td></tr>
<tr><td>Metric Name</td><td colspan="2">Does the software include development metadata which helps define its status?</td></tr>
<tr><td>Description</td><td colspan="2">Software requires descriptive metadata to support indexing, search and discoverability</td></tr>
<tr><td>FAIR4RS Principle</td><td colspan="2">F2: Software is described with rich metadata.
R1: Software is described with a plurality of accurate and relevant attributes.
R3: Software meets domain-relevant community standards.</td></tr>
<tr><td>RSMD Recommendation</td><td colspan="2">RSMD-4.2, RSMD-4.4, RSMD-4.5</td></tr>
<tr><td rowspan="5">Assessment</td><td>Requirements</td><td>☐ Software source code</td></tr>
<tr><td>Method</td><td>Check if the software has machine-readable descriptive metadata associated with it that describes its development and status.</td></tr>
<tr><td>Essential</td><td>The software includes metadata for contact or support in the README or other intrinsic metadata file according to community standards.</td></tr>
<tr><td>Important</td><td>The software includes metadata for development status, links to documentation</td></tr>
<tr><td>Useful</td><td>The metadata is contained in a format such as CodeMeta or ProjectObjectModel that enables full machine-actionability.</td></tr>
<tr><td>Comments</td><td colspan="2">There are many forms of guidance and community standards for structuring development metadata, such as RepoStatus,[17] Software Release Practice HOWTO,[18] Make a README,[19] and AboutCode.[20]
It is still difficult to check all descriptive metadata around development and status as it is often provided in an unstructured form; machine-readable semantic metadata schema are available but</td></tr>
</table>

[17] https://www.repostatus.org/
[18] https://tldp.org/HOWTO/Software-Release-Practice-HOWTO/index.html
[19] https://www.makeareadme.com/
[20] https://www.aboutcode.org/

	not widely used for this purpose (e.g. RepoStatus, Semantic Versioning[21]) or language specific (e.g. Trove Classifiers[22]).

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Metric Identifier</td><td colspan="2">FRSM-06</td></tr>
<tr><td>Metric Name</td><td colspan="2">Does the software include metadata about the contributors and their roles?</td></tr>
<tr><td>Description</td><td colspan="2">Software should make it easy to recognise and credit all contributors.</td></tr>
<tr><td>FAIR4RS Principle</td><td colspan="2">F2: Software is described with rich metadata.
R3: Software meets domain-relevant community standards.</td></tr>
<tr><td>RSMD Recommendation</td><td colspan="2">RSMD-5.1, RSMD-5.2, RSMD-5.3, RSMD-5.4, RSMD-5.5, RSMD-5.6. RSMD-5.7. RSMD-5.8</td></tr>
<tr><td rowspan="5">Assessment</td><td>Requirements</td><td>☐ Software source code
☐ Software identifier</td></tr>
<tr><td>Method</td><td>Check if the software and/or software identifier has machine readable descriptive metadata associated with it that include contributors and roles.</td></tr>
<tr><td>Essential</td><td>The software includes metadata about the contributors</td></tr>
<tr><td>Important</td><td>The software includes citation metadata that includes all contributors and their roles. This includes ORCIDs when contributors have them.</td></tr>
<tr><td>Useful</td><td>Does the citation metadata include the proportional credit attributed to each contributor?</td></tr>
<tr><td>Comments</td><td colspan="2">There are several common places for contributor metadata to be found, including README files, CodeMeta or CFF files, in the code repository metadata, or in the software identifier metadata. It may also be directly embedded in software source code files.
Criteria for which roles are included is normally defined by the community.</td></tr>
</table>

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Metric Identifier</td><td colspan="2">FRSM-07</td></tr>
<tr><td>Metric Name</td><td colspan="2">Does the software metadata include the identifier for the software?</td></tr>
<tr><td>Description</td><td colspan="2">Software should include its identifier to make it easier to be cited and indexed</td></tr>
<tr><td>FAIR4RS Principle</td><td colspan="2">F3: Metadata clearly and explicitly include the identifier of the software they describe.
R3: Software meets domain-relevant community standards.</td></tr>
<tr><td>RSMD Recommendation</td><td colspan="2">No related RSMD recommendation</td></tr>
<tr><td rowspan="2">Assessment</td><td>Requirements</td><td>☐ Software source code
☐ Software identifier</td></tr>
<tr><td>Method</td><td>Check if the software includes its own software identifier, and that the identifier resolves to that software.</td></tr>
</table>

[21] https://semver.org/
[22] https://pypi.org/classifiers/

	Essential	Does the software include an identifier in the README or citation file?
	Important	Does the identifier resolve to the same instance of the software?
	Useful	N/A
Comments	There are several common places for identifier metadata to be found, including README files, CodeMeta or CFF files. The choice of location may depend on community standards.	

Field	Description	
Metric Identifier	FRSM-08	
Metric Name	Does the software have a publicly available, openly accessible and persistent metadata record?	
Description	Even if the software itself is no longer usable or accessible, its metadata should still be available and accessible.	
FAIR4RS Principle	F4: Metadata are FAIR, searchable and indexable. A2: Metadata are accessible, even when the software is no longer available. R3: Software meets domain-relevant community standards. *May enable compliance to F1, F1.1, F1.2, F2, F3*	
RSMD Recommendation	RSMD-1.2	
Assessment	*Requirements*	☐ Software identifier
	Method	Check if the software identifier includes a reference to a persistent landing page or other metadata record, and if that metadata is still accessible.
	Essential	A metadata record for the software is present on an infrastructure that guarantees persistence.
	Important	The persistent metadata record is available through public search engines. The metadata has a globally unique and persistent identifier.
	Useful	The persistent metadata record is available through multiple, cross-referenced infrastructures.
Comments	Potential locations for persistent metadata records include scholarly repositories (e.g. Zenodo, HAL, OSF), registries or catalogues (e.g. ASCL, bio.tools, swMath), open scholarly infrastructure (e.g. Wikidata, DataCite, IPOL, eLife). The choice of location is dependent on community standards.	

Field	Description
Metric Identifier	FRSM-09
Metric Name	Is the software developed in a code repository / forge that uses standard communications protocols?
Description	Software source code repositories / forges (a.k.a. version control platforms) should use standard communications protocols (such as https / sftp) to enable the widest possible set of contributors.
FAIR4RS Principle	A1: Software is retrievable by its identifier using a standardised communications protocol.

	A1.1: The protocol is open, free, and universally implementable. A1.2: The protocol allows for an authentication and authorization procedure, where necessary. R3: Software meets domain-relevant community standards.	
RSMD Recommendation	RSMD-1.3	
Assessment	*Requirements*	☐ Software source code identifier
	Method	Check if the identifier for the code repository / forge can be accessed using standardised communications protocols such as https or sftp.
	Essential	The code repository / forge can be accessed using the identifier via a standardised protocol.
	Important	If authentication or authorisation are required, these are supported by the communication protocols and the repository / forge.
	Useful	N/A
Comments	Frameworks such as the Internet Protocol suite and Open Systems Interconnection model define different abstraction layers for networked communication. Several bodies, such as the IETF and ISO define standardised communications protocols utilised at each layer. In general, most widely used code repositories / forges use common standardised communications protocols such as https or sftp. In normal use, this test will be implemented by checking that the repository / forge can be accessed using one of these protocols. Using a software forge that is properly indexed by search engines will help with other aspects of findability.	

Field	**Description**	
Metric Identifier	FRSM-10	
Metric Name	Are the formats used by the data consumed or produced by the software open and a reference provided to the format?	
Description	The use of open file formats for data improves the reusability and understandability of the software.	
FAIR4RS Principle	I1: Software reads, writes and exchanges data in a way that meets domain-relevant community standards. I2: Software includes qualified references to other objects.	
RSMD Recommendation	RSMD-7.6	
Assessment	*Requirements*	☐ Software source code ☐ Software documentation
	Method	Check the software source code and documentation for references to the data formats used.
	Essential	The documentation describes the data formats used
	Important	The data formats used are open.
	Useful	A reference to the schema is provided.
Comments	This metric is inherently difficult to implement as at present there is no standardised or common method for describing the data / file formats used by a piece of software in a machine-readable way. Community standards commonly define the data formats in use in a	

<table>
<tr><td></td><td>discipline, and resources such as FAIRsharing.org provide a curated catalogues of standards.</td></tr>
</table>

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Metric Identifier</td><td colspan="2">FRSM-11</td></tr>
<tr><td>Metric Name</td><td colspan="2">Does the software use open APIs that support machine-readable interface definition?</td></tr>
<tr><td>Description</td><td colspan="2">An open Application Programming Interface can be freely accessed by other software or developers, which makes it easier to integrate software and encourages modularity and reuse.</td></tr>
<tr><td>FAIR4RS Principle</td><td colspan="2">I1: Software reads, writes and exchanges data in a way that meets domain-relevant community standards.</td></tr>
<tr><td>RSMD Recommendation</td><td colspan="2">No related RSMD recommendation.</td></tr>
<tr><td rowspan="5">Assessment</td><td>Requirements</td><td>☐ Software source code
☐ Software application</td></tr>
<tr><td>Method</td><td>Call the software API.</td></tr>
<tr><td>Essential</td><td>The software provides documented APIs</td></tr>
<tr><td>Important</td><td>The APIs are open (freely accessible)</td></tr>
<tr><td>Useful</td><td>The APIs include a machine-readable interface definition</td></tr>
<tr><td>Comments</td><td colspan="2">Only applicable if APIs are implemented.

The OpenAPI specification[23] is a machine-readable interface definition language for describing, producing, consuming and visualising web services. Additionally, the SmartAPI[24] project has developed a openAPI-based specification for defining the key API metadata elements and value sets, to maximise the FAIRness of web-based APIs.

This could be extended to test that the API is callable and does not return an error code.</td></tr>
</table>

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Metric Identifier</td><td colspan="2">FRSM-12</td></tr>
<tr><td>Metric Name</td><td colspan="2">Does the software provide references to other objects that support its use?</td></tr>
<tr><td>Description</td><td colspan="2">Determining the usefulness of a piece of software is often aided by understanding what it is used with.</td></tr>
<tr><td>FAIR4RS Principle</td><td colspan="2">I2: Software includes qualified references to other objects.</td></tr>
<tr><td>RSMD Recommendation</td><td colspan="2">RSMD-4.3, RSMD-7.6</td></tr>
<tr><td rowspan="3">Assessment</td><td>Requirements</td><td>☐ Software source code
☐ Software identifier</td></tr>
<tr><td>Method</td><td>Check if the software metadata includes references to other related resources.</td></tr>
<tr><td>Essential</td><td>N/A</td></tr>
</table>

[23] https://www.openapis.org/
[24] https://smart-api.info/

	Important	The software metadata includes machine-readable references to articles describing the software, articles demonstrating use of the software, or to the data it uses.
	Useful	N/A
Comments	This metric is currently difficult to implement as there is no standard machine-readable way to define the relationships at a level of detail that provides suitable meaning, although CodeMeta defines some of these relationships (e.g. supportingData, referencePublication).[25]	

Field	Description	
Metric Identifier	FRSM-13	
Metric Name	Does the software describe what is required to use it?	
Description	Software is made more reusable by providing suitable machine-actionable information on dependencies, build and configuration.	
FAIR4RS Principle	R1: Software is described with a plurality of accurate and relevant attributes. R2: Software includes qualified references to other software.	
RSMD Recommendation	RSMD-7.1, RSMD-7.2, RSMD-7.3, RSMD-7.4, RSMD-7.5	
Assessment	*Requirements*	☐ Software
	Method	Check for machine-readable information that helps support the understanding of how it is to be used
	Essential	The software has build, installation and/or execution instructions
	Important	Dependencies are provided in a machine-readable format and the building and installation of the software is automated.
	Useful	N/A
Comments	Most programming languages provide standardised ways of providing dependency information in a machine-actionable format. Build and package management systems can be used to automate the installation process. It is hard to check the relevance / correctness of this information, but it is possible to automatically check for existence and error-free build. Detailed documentation also aids the reusability of software but it is difficult to automatically test for documentation coverage.	

Field	Description
Metric Identifier	FRSM-14
Metric Name	Does the software come with test cases to demonstrate it is working?
Description	The provision of test cases improves confidence in the software.
FAIR4RS Principle	R1: Software is described with a plurality of accurate and relevant attributes.
RSMD Recommendation	RSMD-7.5

[25] https://codemeta.github.io/terms/

Assessment	*Requirements*	☐ Software source code
	Method	Check for the presence of automated tests
	Essential	Tests and data are provided to check that the software is operating as expected
	Important	Automated unit and system tests are provided
	Useful	Code coverage / test coverage is reported
Comments	Most programming languages have commonly associated test frameworks. The specific definition of what constitutes adequate testing is often defined by community norms. It is hard to check the relevance / correctness of this information, but it is possible to automatically check for existence.	

Field	Description	
Metric Identifier	FRSM-15	
Metric Name	Does the software source code include licensing information for the software and any bundled external software?	
Description	Clear software licensing enables reuse.	
FAIR4RS Principle	R1.1: Software is given a clear and accessible licence.	
RSMD Recommendation	RSMD-6.2, RSMD-6.4, RSMD-6.5, RSMD-6.6	
Assessment	*Requirements*	☐ Software source code ☐ Software
	Method	Check the software and its documentation for the presence of a licence
	Essential	The software includes its LICENCE file
	Important	The source code includes licensing information for all components bundled with that software
	Useful	The software licensing information is in SPDX format
Comments	Each community may have different licences that are popular. It is important that software licences are included with the source code as many tools and processes look for licensing information there to determine licence compatibility. The SPDX License List[26] is a widely used part of the Software Project Data eXchange (SPDX) open standard. Information about the licence for a piece of software can be provided either as a file in the source code repository, or as a short identifier embedded in the source code files.	

Field	Description
Metric Identifier	FRSM-16
Metric Name	Does the software metadata record include licensing information?
Description	It is important for licensing information to be on the publicly searchable and accessible metadata record
FAIR4RS Principle	R1.1: Software is given a clear and accessible licence.
RSMD Recommendation	RSMD-6.3

[26] https://spdx.org/licenses/

Assessment	*Requirements*	☐ Software identifier
	Method	Check if the software identifier or the metadata record referenced by it includes licensing information
	Essential	The identifier or metadata record includes licensing and copyright information
	Important	N/A
	Useful	The software licensing information is in SPDX format, or other machine-readable form.
Comments	This can be defined in different ways, e.g. the "Rights" field in the DOI metadata.	

Field	Description	
Metric Identifier	FRSM-17	
Metric Name	Does the software include provenance information that describe the development of the software?	
Description	Good provenance metadata clarifies the origins and intent behind the development of the software, and establishes authenticity and trust. As a type of metadata this overlaps with the metadata called for in guiding principles F2 and F4.	
FAIR4RS Principle	R1.2: Software is associated with detailed provenance.	
RSMD Recommendation	RSMD-4.5	
Assessment	*Requirements*	☐ Software source code repository / forge
	Method	Check the development metadata available from the code repository / forge for the software
	Essential	The software source code repository / forge includes a commit history
	Important	The software source code repository links commits to issues / tickets
	Useful	The software project uses other tools to capture detailed machine readable provenance information.
Comments	It is hard to check the relevance / correctness of this information, but it is possible to automatically check for existence. It may also be necessary to record information about the way that the software has been developed, such as the development environment used. The methodology for building the software is tested in FRSM-13.	

3. Disciplinary Exemplar

This section provides an example of how the metrics might be used in a disciplinary context, taking the social sciences as an exemplar. There are many community standards and norms that will affect the choice of implementation. For example, checking the type of identifier (FRSM-01) will depend on the identifier schemes commonly in use. In many research fields, DOIs are commonly used, but in some disciplines others may be popular e.g. RRIDs in biomedicine or ARKs in cultural institutions.

By providing an implementation of the metrics defined in Section 2 to a particular disciplinary community, it is possible to test the applicability of the metrics, as well as provide further context for how other communities could utilise them.

3.1 Use case: CESSDA software guidelines

The metrics have been mapped to the CESSDA Technical Guidelines for Social Science.[27] These guidelines define how CESSDA products are developed, by following CESSDA's implementation of the EURISE Network Technical Reference,[28] and include specific guidance on software development and software maturity levels (SMLs). The SMLs provide guidance on the minimum, expected and excellent standards for each of the 12 CESSDA Maturity Assessment criteria (documentation, intellectual property, extensibility, modularity, packaging, portability, standards compliance, maintenance, verification and testing, security, internalisation and localisation, authentication and authorisation) and can be used to suggest what is necessary to meet essential, important and useful compliance levels.

Field	Description	
Metric Identifier	FRSM-01-CESSDA	
Metric Name	Does the software have a globally unique and persistent identifier?	
Assessment	*Requirements*	☐ Software releases of open source components to be published in Zenodo ☐ DOI handle
	Method	Check that an established identifier scheme from the CESSDA Software Publication polices is used to identify software.
	Essential	A version-dependent DOI must be added in the repository's README as the recommended citation
	Important	Releases use the Semantic Versioning 2.0.0 notation
	Useful	Only Major and Minor releases are assigned DOIs
Comments	See the Software Publication[29] of open source components as per CESSDA's Publication Policy & Procedures (CESSDA, 2020).	

[27] https://docs.tech.cessda.eu/index.html
[28] https://technical-reference.readthedocs.io/en/latest/
[29] https://docs.tech.cessda.eu/software/publication.html

	As described in the CESSDA ERIC Persistent Identifier Policy,[30] CESSDA tools and services accept: DOI, Handle (including ePIC-handles), URN, ARK (fulfilling principle 10 of the CESSDA Data Access Policy).

Field	Description	
Metric Identifier	FRSM-02-CESSDA	
Metric Name	Can different components of the software be individually identified?	
Assessment	*Requirements*	☐ Software source code repository
	Method	Check that each software product is split into component microservices, each with its own DOI
	Essential	A separate Git repository is used for the source code of each component (aka microservices). The product deployment scripts assemble the constituent components.
	Important	Each component is deposited in Zenodo with its own DOI.
	Useful	The Zenodo record for each component is tagged with the product(s) that it contributes to.
Comments	CESSDA requirements for modularity are defined in CMA4: Modularity.[31] CESSDA's products are designed and built using a microservices approach. It is expected that a separate Git repository is used for the source code of each component (aka microservice).	

Field	Description	
Metric Identifier	FRSM-03-CESSDA	
Metric Name	Does each version of the software have a unique identifier?	
Assessment	*Requirements*	☐ Repository release tag ☐ Software release identifier
	Method	Check that each release follows CESSDA software publication policies and is deposited in a repository that provides a unique DOI for each release.
	Essential	Each release is published to Zenodo and a DOI obtained. A publication consists of a release tarball matching the release tag in the repository. Release tags exist and adhere to SemVer 2.0.0. The README and CHANGELOG must be up to date prior to release and they must be added to the Zenodo record in addition to the tarball.
	Important	A release checklist is used to ensure that all necessary steps are taken for each release. Releases must be available as Docker images with the release version as tag.

[30] https://zenodo.org/badge/DOI/10.5281/zenodo.6607000.sv
[31] https://docs.tech.cessda.eu/sml/ca4-modularity.html

	Useful	Reserve the DOI in Zenodo, prior to release, to avoid a circularity problem with the CHANGELOG and the tarball.
Comments	These are derived from the CESSDA Software Publication policy and procedures for open source components,[32] as set out in the CESSDA Publication Policy & Procedures (CESSDA, 2020).	

Field	Description	
Metric Identifier	FRSM-04-CESSDA	
Metric Name	Does the software include descriptive metadata which helps define its purpose?	
Assessment	*Requirements*	☐ Software identifier (DOI) provided by Zenodo
	Method	Query the metadata provided by the Zenodo record for the software
	Essential	Zenodo metadata includes the software name and description
	Important	Zenodo metadata includes other descriptive metadata as recommended in CESSDA Software Requirements
	Useful	N/A
Comments	CESSDA technical guidelines on CMA1: Documentation[33] define what is required from end-user documentation, operational documentation, and development documentation but these are not machine-accessible. The CESSDA Software Requirements[34] also demand that all tools and products have a comprehensive README.	

Field	Description	
Metric Identifier	FRSM-05	
Metric Name	Does the software include development metadata which helps define its status?	
Assessment	*Requirements*	☐ Software source code in repository
	Method	Check the README and CHANGELOG files for development status indicators
	Essential	The README and CHANGELOG must be up to date. The README contains release details, version details, links to documentation as described in the EURISE Network Technical Reference.[35]
	Important	Version numbering follows Semantic Versioning 2.0.0 and pre-release versions may be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version

[32] https://docs.tech.cessda.eu/software/publication.html
[33] https://docs.tech.cessda.eu/sml/ca1-documentation.html
[34] https://docs.tech.cessda.eu/software/requirements.html
[35] https://technical-reference.readthedocs.io/en/v0.2/developer-guidelines/02-readme.html

	Useful	N/A
Comments	Some of this metadata is machine readable but requires interpretation. For CESSDA, active status would be defined as there being a recent release (release date) and that it is maintained (recent commits).	

Field	**Description**	
Metric Identifier	FRSM-06-CESSDA	
Metric Name	Does the software include metadata about the authors and their roles?	
Assessment	*Requirements*	☐ Software source code ☐ Software identifier
	Method	Check that the CITATION and/or CONTRIBUTORS files exist and Zenodo metadata is present
	Essential	A CITATION and/or CONTRIBUTORS files is present in the root of the repository.
	Important	Author details (including ORCIDs) are present in the corresponding Zenodo record. ORCIDs are present for authors in the CITATION.cff file.
	Useful	N/A
Comments	Authorship criteria should follow the CESSDA Publication Policy & Procedures (CESSDA, 2020). CESSDA uses Citation File Format[36] for recording authorship, e.g. CDC-Searchkit citation.[37]	

Field	**Description**	
Metric Identifier	FRSM-07-CESSDA	
Metric Name	Does the software metadata include the identifier of the software?	
Assessment	*Requirements*	☐ Software source code
	Method	Check that README and CITATION files exist and include the DOI for the corresponding software release.
	Essential	The README file includes the DOI that represents all versions in Zenodo
	Important	The CITATION.cff file included in the root of the repository includes the appropriate DOI for the corresponding software release in Zenodo.
	Useful	N/A
Comments	The Zenodo DOI representing all versions will always resolve to the latest version in Zenodo. CESSDA uses Citation File Format, which can include a reference to the software identifier.	

Field	**Description**
Metric Identifier	FRSM-08-CESSDA

[36] https://citation-file-format.github.io/
[37] https://github.com/cessda/cessda.cdc.searchkit/blob/main/CITATION.cff

Metric Name	Does the software have a publicly available, openly accessible and persistent metadata record?		
Assessment	*Requirements*	☐ Software identifier	
	Method	Check that a DOI exists for the latest release and resolves to a Zenodo landing page.	
	Essential	The DOI resolves to a Zenodo landing page for the latest release, and metadata can be accessed via the Zenodo API.	
	Important	The Zenodo metadata record is available through public search engines.	
	Useful	The persistent metadata record is available through multiple, cross-referenced infrastructures, including OpenAIRE .	
Comments	Software releases of open source components should be published on Zenodo, as per CESSDA's Publication Policy & Procedures (CESSDA, 2020). Recommended metadata from the CESSDA Technical Guidelines on Software Publication include version, authors, name, description and identifier.		

Field	**Description**	
Metric Identifier	FRSM-09-CESSDA	
Metric Name	Is the software developed in a code repository/forge that uses standard communication protocols?	
Assessment	*Requirements*	☐ Software source code identifier
	Method	Check that the git repository of the component is accessible using standardised communications protocols such as https or sftp.
	Essential	Ensure that repositories containing component software are publicly accessible.
	Important	No authentication is required to view and/or clone CESSDA's public repositories, even so, their contents cannot be modified directly by 3rd parties.
	Useful	Pull requests are used to propose modifications to the contents.
Comments	Development of CESSDA tools and services is carried out using CESSDA-owned git-repositories on Github.[38] If the code is developed publicly elsewhere, mirroring with clear pointers to the upstream are used.[39]	

Field	**Description**	
Metric Identifier	FRSM-10-CESSDA	
Metric Name	Are the data formats used by the software open and a reference provided to the format?	
Assessment	*Requirements*	☐ Software source code

[38] https://github.com/cessda
[39] https://docs.tech.cessda.eu/software/index.html

		☐ Software documentation
	Method	Check that data content used by CESSDA services is machine-readable
	Essential	The data formats used by the software are noted in the documentation.
	Important	The data complies with a recognised standard used by the CESSDA community (typically DDI/XML, RDF/XML, TURTLE, JSON-LD or SKOS).
	Useful	Where a public API is used to access the data content, it complies with the OpenAPI standard.
Comments	CESSDA documents its approach to open data standards in CMA7 - Standards Compliance.[40]	

Field	Description	
Metric Identifier	FRSM-11-CESSDA	
Metric Name	Does the software use open APIs that support machine-readable interface definition?	
Assessment	*Requirements*	Software application
	Method	Call the API
	Essential	The API meets SML3 of the CESSDA Development Documentation guidelines: there is external documentation that describes all API functionality, which is sufficient to be used by any developer.
	Important	The software's REST APIs comply with the OpenAPI standard.
	Useful	The software's REST APIs are described in the published CESSDA API definitions[41].
Comments	Expectations around the API definition and documentation are set out in the section on CMA1.3 Development Documentation of the CESSDA Technical Guidelines.[42] The section on CMA7 Demonstrate Usability notes that at SML5 (excellent standard) compliance with open or internationally recognised standards for the software and software development process, is evident and documented, and verified through testing of all components. At present, this is not being included in the assessment criteria as it is hard to automatically test, but could be independently verified through regular testing and certification from an independent group. An extensible service enables additional services to be built on or around it, including adapting to changing functional requirements over time. This is done by making the integration point the API. New and/or existing services can be combined as required via their APIs to meet changing functional requirements. Versioning the APIs and	

[40] https://docs.tech.cessda.eu/sml/ca7-standards-compliance.html
[41] https://api.tech.cessda.eu/
[42] https://docs.tech.cessda.eu/sml/ca1-documentation.html#cma13-development-documentation

	supporting two versions simultaneously allows services to evolve, without breaking the contract they provide to their consumers.[43]

Field	Description	
Metric Identifier	FRSM-12-CESSDA	
Metric Name	Does the software provide references to other objects that support its use?	
Assessment	*Requirements*	
	Method	Not applicable for CESSDA
	Essential	N/A
	Important	N/A
	Useful	N/A
Comments	CESSDA uses the “docs-as-code” approach for end user and content editor demonstration. Therefore, for this metric, it is hard for CESSDA tools and services to demonstrate compliance. Therefore, this metric is not useful to assess at present. At present, CESSDA does not require publications describing the software - if this changed, a suitable assessment for this metric would be to test the identifier for the publication to be included in the software metadata.	

Field	Description	
Metric Identifier	FRSM-13-CESSDA	
Metric Name	Does the software describe what is required to use it?	
Assessment	*Requirements*	☐ Software
	Method	Check the README file.
	Essential	Dependency information and build instructions are included in the README file. Linting and other relevant checks are present in the automated build and test process (e.g. via the Jenkinsfile).
	Important	The README file includes a badge that links to the automated build tool (Jenkins). Deployment to development and staging environments is automated (conditional on test results).
	Useful	The build badge indicates the status of the latest build (passing or failing)
Comments	See Software Maturity Levels (SML)[44] for: CMA1 - Documentation, CMA3 - Extensibility, CM4 - Modularity, CMA5 - Packaging, CMA6 - Portability, and CMA7 - Standards Compliance. Source code documentation should use the de facto standard for chosen language, e.g: JavaDoc for Java.[45] Although no language-specific coding conventions are mandated, the ‘Coding conventions for languages’ section of the Wikipedia Coding	

[43] https://docs.tech.cessda.eu/software/interoperability.html#extensible
[44] https://docs.tech.cessda.eu/sml/index.html
[45] https://docs.tech.cessda.eu/software/documentation-guidelines/development-documentation.html#technical-manual

	conventions page is a useful reference source for language-specific guidelines, if required.[46]

Field	Description	
Metric Identifier	FRSM-14-CESSDA	
Metric Name	Does the software come with test cases to demonstrate it is working?	
Assessment	*Requirements*	☐ Software source code
	Method	Check the README file.
	Essential	The README file includes badges that link to a comprehensive code quality assessment tool (SonarQube) and automated build tool (Jenkins).
	Important	*CMA9-SML5 - Demonstrable usability:* A production system has been tested and validated through successful use of the application. *CMA7-SML5 - Demonstrable usability:* Compliance with open or internationally recognised standards for the software and software development process, is evident and documented, and verified through testing of all components. Ideally independent verification is documented through regular testing and certification from an independent group.
	Useful	The README file badges indicate the status of the tests and other code quality metrics. The repository contains a subdirectory containing code for the test cases that are run automatically.
Comments	See Software Maturity Levels (SML) for: CMA9 - Verification and Testing[47] and CMA7 Standards Compliance. CESSDA periodically runs the SQAaaS tool[48] against its publicly accessible repositories and displays the results via a badge in the README file.	

Field	Description	
Metric Identifier	FRSM-15-CESSDA	
Metric Name	Does the software source code include licensing information for the software and any bundled external software?	
Assessment	*Requirements*	☐ Software source code ☐ Software
	Method	Check that the LICENSE file exists. Check that the source code headers include a licensing statement.
	Essential	Include a LICENSE.txt file in the root of the repository.

[46] https://docs.tech.cessda.eu/software/documentation-guidelines/index.html#software-code-structure
[47] https://docs.tech.cessda.eu/sml/ca9-verification-and-testing.html
[48] https://sqaaas.eosc-synergy.eu/#/auth/full-assessment

	Important	Include licensing information in the source code header.
	Useful	The build script (Maven POM, where used) checks that the standard header is present in all source code files.
Comments	CESSDA guidance on licence information is part of the guidelines on Standard Git Repository Contents,[49] Further guidance is provided as part of the guidance on CMA2 - Intellectual Property.[50]	

Field	**Description**	
Metric Identifier	FRSM-16-CESSDA	
Metric Name	Does the software metadata record include licensing information?	
Assessment	*Requirements*	☐ Software identifier
	Method	Check for the presence of licence information in the Zenodo repository and source code deposited in the repository
	Essential	Licensing information is included in the Zenodo record and in a LICENSE.txt file included in the root directory of the source code deposited in Zenodo.
	Important	N/A
	Useful	N/A
Comments	CESSDA guidance on licence information is part of the guidelines on Standard Git Repository Contents.	

Field	**Description**	
Metric Identifier	FRSM-17-CESSDA	
Metric Name	Does the software include provenance information?	
Assessment	*Requirements*	☐ Software source code repository
	Method	Check the commit history of the code repository
	Essential	Code repository contains commit messages
	Important	Code that addresses an issue is developed in a branch prefixed with the issue number.
	Useful	Links to Pull Requests are included in issue tracker tickets.
Comments	Git repositories include a commit history as a matter of course. CESSDA uses git repos on GitHub, and uses a branching model where each branch is prefixed with the issue tracker ticket number that it addresses.	

[49] https://docs.tech.cessda.eu/technical-infrastructure/gcp-repository-standard-contents.html#overview
[50] https://docs.tech.cessda.eu/sml/ca2-intellectual-property.html

References

Amdouni, Emna, Bouazzouni, Syphax, & Jonquet, Clement. (2022). O'FAIRe makes you an offer: metadata-based automatic FAIRness assessment for ontologies and semantic resources. In International Journal of Metadata, Semantics and Ontologies (Vol. 16, Issue 1, pp. 16–46). Inderscience Publishers. https://doi.org/10.1504/ijmso.2022.131133

Asmi, Ari, Cordewener, Bas, Goble, Carole, Castelli, Donatella, Kühn, Eileen, Pasian, Fabio, Niccolucci, Franco, Glaves, Helen, Jeffery, Keith, Assante, Massimiliano, Dovey, Matthew, Manola, Natalia, Juty, Nick, Blomberg, Niklas, Jimenez, Rafael, Beckmann, Volker. (2017). D6.3: 1st Report on Data Interoperability: Findability and Interoperability. (1.1). EOSCpilot. Available from: https://www.eoscpilot.eu/sites/default/files/eoscpilot-d6.3.pdf

Barker, Michelle, Chue Hong, Neil P., Katz, Daniel S. et al. Introducing the FAIR Principles for research software. Sci Data 9, 622 (2022). https://doi.org/10.1038/s41597-022-01710-x

Bourque, Pierre and Fairley, Richard E. (eds.) (2014). Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society. Available from: https://www.swebok.org.

CESSDA ERIC. (2020). CESSDA Publication Policy & Procedures (1.0). Zenodo. https://doi.org/10.5281/zenodo.3904264

Chue Hong, Neil P., Katz, Daniel. S., Barker, Michelle, Lamprecht, Anna-Lena, Martinez, Carlos, Psomopoulos, Fotis E., Harrow, Jen, Castro, Leyla Jael, Gruenpeter, Morane, Martinez, Paula Andrea, Honeyman, Tom, et al. (2021). FAIR Principles for Research Software (FAIR4RS Principles). Research Data Alliance. https://doi.org/10.15497/RDA00065

Chue Hong, Neil P., Katz, Daniel S., Barker, Michelle, Lamprecht, Anna-Lena, Martinez, Carlos, Psomopoulos, Fotis E., Harrow, Jen, Castro, Leyla Jael, Gruenpeter, Morane, Martinez, Paula Andrea, Honeyman, Tom, Struck, Alexander, Lee, Allen, Loewe, Axel, van Werkhoven, Ben, Jones, Catherine, Garijo, Daniel, Plomp, Esther, Genova, Francoise, … RDA FAIR4RS WG. (2022). FAIR Principles for Research Software (FAIR4RS Principles) (1.0). https://doi.org/10.15497/RDA00068

Chue Hong, Neil P (ed), Gruenpeter, Morane, Antonioletti, Mario, and Priddy, Mike. (2023) Community Workshop on Metrics for FAIR Software. Zenodo. https://doi.org/10.5281/zenodo.8393889

Devaraju, Anusuriya, & Huber, Robert. (2021). An automated solution for measuring the progress toward FAIR research data. *Patterns*, *2*(11), 100370. https://doi.org/10.1016/j.patter.2021.100370

Devaraju, Anusuriya, Huber, Robert, Mokrane, Mustapha, Herterich, Patricia, Cepinskas, Linas, de Vries, Jerry, L'Hours, Herve, Davidson, Joy, & White, Angus. (2022). FAIRsFAIR Data Object Assessment Metrics (0.5). Zenodo. https://doi.org/10.5281/zenodo.6461229

European Commission. Directorate General for Research and Innovation. (2020). Scholarly infrastructures for research software: report from the EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS. Publications Office. https://doi.org/10.2777/28598

FAIR Data Maturity Model Working Group. (2020). FAIR Data Maturity Model. Specification and Guidelines (1.0). Zenodo. https://doi.org/10.15497/rda00050

Gaignard, Alban, Rosnet, Thomas, De Lamotte, Frédéric, Lefort, Vincent, & Devignes, Marie-Dominique. (2023). FAIR-Checker: Supporting digital resource findability and reuse with Knowledge Graphs and Semantic Web standards. *Journal of Biomedical Semantics*, *14*(1). https://doi.org/10.1186/s13326-023-00289-5

GO FAIR. (2018). FAIR Principles. GO FAIR Initiative. https://www.go-fair.org/fair-principles/ Retrieved from: https://web.archive.org/web/20180212143802/https://www.go-fair.org/fair-principles/

Gruenpeter, Morane, Di Cosmo, Roberto, Koers, Hylke, Herterich, Patricia, Hooft, Rob, Parland-von Essen, Jessica, Tana, Jonas, Aalto, Tero, & Jones, Sarah. (2020). M2.15 Assessment report on 'FAIRness of software' (1.1). Zenodo. https://doi.org/10.5281/zenodo.4095092

Gruenpeter, Morane, Katz, Daniel S., Lamprecht, Anna-Lena, Honeyman, Tom, Garijo, Daniel, Struck, Alexander, Niehues, Anna, Martinez, Paula Andrea, Castro, Leyla Jael, Rabemanantsoa, Tovo, Chue Hong, Neil P., Martinez-Ortiz, Carlos, Sesink, Laurents, Liffers, Matthias, Fouilloux, Anne Claire, Erdmann, Chris, Peroni, Silvio, Martinez Lavanchy, Paula, Todorov, Ilian, & Sinha, Manodeep. (2021). Defining Research Software: a controversial discussion (Version 1). Zenodo. https://doi.org/10.5281/zenodo.5504016

Gruenpeter, Morane, Granger, Sabrina, Monteil, Alain, Chue Hong, Neil, Breitmoser, Elena, Antonioletti, Mario, Garijo, Daniel, González Guardia, Esteban, Gonzalez Beltran, Alejandra, Goble, Carole, Soiland-Reyes, Stian, Juty, Nick, & Mejias, Gabriela. (2023). D4.4 - Guidelines for recommended metadata standard for research software within EOSC (V1.0 DRAFT NOT YET APPROVED BY EUROPEAN COMMISSION). Zenodo. https://doi.org/10.5281/zenodo.8199104

ISO. (2011). ISO/IEC 25010:2011: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. ISO. Available from: https://www.iso.org/standard/35733.html

Lamprecht, Anna-Lena, Garcia, Leyla, Kuzak, Mateusz, Martinez, Carlos, Arcila, Ricardo, Martin Del Pico, Eva, Dominguez Del Angel, Victoria, van de Sandt, Stephanie, Ison, Jon, Martinez, Paula Andrea, McQuilton, Peter, Valencia, Alfonso, Harrow, Jennifer, Psomopoulos, Fotis, Gelpi, Josep LL., Chue Hong, Neil, Goble, Carole, & Capella-Gutierrez, Salvador. (2020). Towards FAIR principles for research software [JB]. Data Science, 3(1), 37–59. https://doi.org/10.3233/DS-190026

Kan, Stephen H. (2002). Metrics and Models in Software Quality Engineering, Second Edition. Addison Wesley. ISBN: 9780201729153.

Katz, Daniel S., Gruenpeter, Morane, & Honeyman, Tom (2021). Taking a fresh look at FAIR for research software. Patterns, 2(3), 100222. https://doi.org/10.1016/j.patter.2021.100222

Martinez, Paula Andrea, Erdmann, Christopher, Simons, Natasha, Otsuji, Reid, Labou, Stephaine, Johnson, Ryan, Castelao, Guilherme, Boas, Bia Villas, Lamprecht, Anna-Lena, Ortiz, Carlos Martinez, Garcia, Leyla, Kuzak, Mateusz, Stokes, Liz, Honeyman, Tom, Wise, Sharyn, Quan, Josh, Peterson, Scott, Neeser, Amy, Karvovskaya, Lena, … Fankhauser, Eliane. (2019, February 1). *Top 10 FAIR data & software things*. Zenodo. https://zenodo.org/record/3409968

Orviz, Pablo, López García, Álvaro, Duma, Doina C., Donvito, Giacinto, David, Mario, Gomes, Jorge, Campos, Isabel, Moltó, Germán, & Tykhonov, Vyacheslav. (2022). A set of common software quality assurance baseline criteria for research projects. DIGITAL.CSIC. http://hdl.handle.net/10261/160086

Wilkinson, Mark D., Dumontier, Michel, Aalbersberg, IJsbrand. J., Appleton, Gabrielle, Axton, Myles, Baak, Arie, Blomberg, Niklas, Boiten, Jan-Willem, da Silva Santos, Luiz Bonino, Bourne, Philip E., Bouwman, Jildau, Brookes, Anthony J., Clark, Tim, Crosas, Merce, Dillo, Ingrid, Dumon, Olivier, Edmunds, Scott, Evelo, Chris T., Finkers, Richard, … Mons, Barend. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*(1). https://doi.org/10.1038/sdata.2016.18

Wilkinson, Mark D., Sansone, Susanna-Assunta, Schultes, Erik, Doorn, Peter, Bonino da Silva Santos, Luiz O., & Dumontier, Michel. (2018). A design framework and exemplar metrics for FAIRness. In Scientific Data (Vol. 5, Issue 1). Springer. https://doi.org/10.1038/sdata.2018.118

Zuser, Wolfgang, Heil, Stefan, & Grechenig, Thomas. (2005). Software quality development and assurance in RUP, MSF and XP. *Proceedings of the Third Workshop on Software Quality - 3-WoSQ*. http://dx.doi.org/10.1145/1083292.1083300

Appendices

Appendix A - Evolution of FAIR principles from data to software

As background information, this section details how the development of the FAIR4RS Principles has evolved, by comparison of The FAIR Guiding Principles for scientific data management and stewardship (Wilkinson et al., 2016, with foundational principle text taken from GO FAIR, 2018) with the Towards FAIR Principles for research software (Lamprecht et al., 2020) and Taking a fresh look at FAIR for research software report (Katz, Gruenpeter & Honeyman, 2021), the previous draft for community review (Chue Hong et al., 2021) and the FAIR4RS Principles described in this document.

FAIR Guiding Principles (2016)	**Towards FAIR Principles for research software (2020)**	**Taking a fresh look at FAIR for research software (2021)**	**FAIR4RS Principles Draft for RDA Community Review (2021)**	**FAIR4RS Principles (2022)**
F. Findable				
The first step in (re)using data is to find them. Metadata and data should be easy to find for both humans and computers. Machine-readable metadata are essential for automatic discovery of datasets and services, so this is an essential component of the FAIRification process.	The main concern of findability for research software is to ensure software can be identified unambiguously when looking for it using common search strategies.	The first step in (re)using software is to find it. Metadata and software should be easy to find for both humans and computers. Machine-readable metadata are essential for automatic discovery of software, so this is an essential component of the FAIRification process.	The software, and its associated metadata, should be easy to find for both humans and machines.	Software, and its associated metadata, is easy for both humans and machines to find.
F1. (Meta)data are assigned a globally unique and	F1. Software and its associated metadata have a	F1. Software is assigned a globally unique and	F1. Software is assigned a globally unique and	F1. Software is assigned a globally unique and

persistent identifier	global, unique and persistent identifier for each released version.	persistent identifier	persistent identifier.	persistent identifier.
			F1.1. Different components of the software must be assigned distinct identifiers representing different levels of granularity.	F1.1. Components of the software representing levels of granularity are assigned distinct identifiers.
			F1.2. Different versions of the same software must be assigned distinct identifiers.	F1.2. Different versions of the software are assigned distinct identifiers.
F2. Data are described with rich metadata (defined by R1 below)	F2. Software is described with rich metadata.	F2. Software is described with rich metadata (defined first by R1 below, and then by the original FAIR principles for metadata)	F2. Software is described with rich metadata.	F2. Software is described with rich metadata.
F3. Metadata clearly and explicitly include the identifier of the data they describe	F3. Metadata clearly and explicitly include identifiers for all the versions of the software it describes.	F3. Metadata clearly and explicitly include the identifier of the software they describe	F3. Metadata clearly and explicitly include the identifier of the software they describe.	F3. Metadata clearly and explicitly include the identifier of the software they describe.
F4. (Meta)data are registered or indexed in a searchable resource	F4. Software and its associated metadata are included in a searchable software registry.	F4. Software is registered or indexed in a searchable resource	F4. Metadata are FAIR and is searchable and indexable.	F4. Metadata are FAIR, searchable and indexable.
A. Accessible				
Once the user finds the required data, she/he needs to know how can they be accessed, possibly including authentication and	Accessibility translates into retrievability [...] however, we found mere retrievability not enough. In order for anyone to use any research	Once the user finds the required software, they need to know how it can be accessed, possibly including authentication and	The software, and its metadata, must be retrievable via standardized protocols.	Software, and its metadata, is retrievable via standardized protocols.

authorisation.	software, a working version of the software needs to be available.	authorization.		
A1. (Meta)data are retrievable by their identifier using a standardized communications protocol	A1. Software and its associated metadata are accessible by their identifier using a standardized communications protocol.	A1. Software is retrievable by its identifier using a standardized communications protocol	A1. Software is retrievable by its identifier using a standardized communications protocol.	A1. Software is retrievable by its identifier using a standardized communications protocol.
A1.1. The protocol is open, free, and universally implementable	A1.1. The protocol is open, free, and universally implementable.	A1.1. The protocol is open, free, and universally implementable	A1.1. The protocol is open, free, and universally implementable.	A1.1. The protocol is open, free, and universally implementable.
A1.2. The protocol allows for an authentication and authorization procedure, where necessary	A1.2. The protocol allows for an authentication and authorization procedure, where necessary.	A1.2. The protocol allows for an authentication and authorization procedure, where necessary	A1.2. The protocol allows for an authentication and authorization procedure, where necessary.	A1.2. The protocol allows for an authentication and authorization procedure, where necessary.
A2. Metadata are accessible, even when the data are no longer available	A2. Software metadata are accessible, even when the software is no longer available.	A2. Metadata are accessible, even when the software is no longer available	A2. Metadata are accessible, even when the software is no longer available.	A2. Metadata are accessible, even when the software is no longer available.
I. Interoperable				
The data usually needs to be integrated with other data. In addition, the data need to interoperate with applications or workflows for analysis, storage, and processing.	Interoperability for research software can be understood in two dimensions: as part of workflows (horizontal dimension) and as stack of digital objects that need to work together at compilation and execution times (vertical dimension)	The software usually needs to communicate with other software via exchanged data (or possibly its metadata). Software tools can interoperate via common support for the data they exchange.	The software interoperates with other software through exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs).	Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.

I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.	I1. Software and its associated metadata use a formal, accessible, shared and broadly applicable language to facilitate machine readability and data exchange.	I1. Software should read, write or exchange data in a way that meets domain-relevant community standards	I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.	I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.
I2. (Meta)data use vocabularies that follow FAIR principles	I2.1. Software and its associated metadata are formally described using controlled vocabularies that follow the FAIR principles.		*Now split between F4 and I1.*	*Now split between F4 and I1.*
	I2.2. Software use and produce data in types and formats that are formally described using controlled vocabularies that follow the FAIR principles.			
I3. (Meta)data include qualified references to other (meta)data		I2. Software includes qualified references to other objects.	I2. Software includes qualified references to other objects.	I2. Software includes qualified references to other objects.
	I4S. Software dependencies are documented and mechanisms to access them exist.			
R. Reusable				
The ultimate goal of FAIR is to optimize the reuse of data. To achieve this, metadata and data should	Reusability in the context of software has many dimensions. At its core, reusability aims for someone	The ultimate goal of FAIR is to enable and encourage the use and reuse of software. To achieve this, software	The software is both usable (it can be executed) and reusable (it can be understood, modified, built	Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or

be well-described so that they can be replicated and/or combined in different settings.	to be able to reuse software reproducibly.	should be well-described (by metadata) and appropriately structured so that it can be replicated, combined, reinterpreted, reimplemented, and/or used in different settings.	upon, or incorporated into other software).	incorporated into other software).
R1. (Meta)data are richly described with a plurality of accurate and relevant attributes	R1. Software and its associated metadata are richly described with a plurality of accurate and relevant attributes.	R1. Software is richly described with a plurality of accurate and relevant attributes	R1. Software is described with a plurality of accurate and relevant attributes.	R1. Software is described with a plurality of accurate and relevant attributes.
R1.1. (Meta)data are released with a clear and accessible data usage license	R1.1. Software and its associated metadata have independent, clear and accessible usage licenses compatible with the software dependencies.	R1.1. Software is made available with a clear and accessible software usage license	R1.1. Software must have a clear and accessible license.	R1.1. Software is given a clear and accessible license.
R1.2. (Meta)data are associated with detailed provenance	R1.2. Software metadata include detailed provenance, detail level should be community agreed.	R1.2. Software is associated with detailed provenance	R1.2. Software is associated with detailed provenance.	R1.2. Software is associated with detailed provenance.
R1.3. (Meta)data meet domain-relevant community standards	R1.3. Software metadata and documentation meet domain-relevant community standards.	R1.3. Software meets domain-relevant community standards	R3. Software meets domain-relevant community standards.	R3. Software meets domain-relevant community standards.
		R2. Software includes qualified references to other software	R2. Software includes qualified references to other software.	R2. Software includes qualified references to other software.