Research Software Workshop Session A

metadata property analysis

Exercise 1: metadata property analysis

Step A: Choose a group
Step B: Click on the name of the property to fill the dedicated template
Step C: Fill the template asking you: Why? Who? Where? How?
Step D: If your small group encounters complexities with the term, capture the difficulties in the “Discussion” section
Step E: If you agree the term or description should be modified in CodeMeta, open an issue here: https://github.com/codemeta/codemeta/issues

Links

- CodeMeta repository (vocabulary and crosswalk)
- CodeMeta description: https://github.com/codemeta/codemeta/blob/master/properties_description.csv
- Issue-tracker
- CodeMeta generator hosted version
  - Contributions are welcome on the code repository

Funded by
the European Union

Table of Contents

Funded by
the European Union

Instructions for exercise 1

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
(Description from CodeMeta, highlight added information)	*(Use case, purpose to use term)*	*(Intrinsic, extrinsic or both, which platform? Information source, provenance)*	*(Provider of metadata)*	*(availability of the information, metadata creation process - manual or automatic)*	*(Which **metric** can be used to verify this metadata and it's quality)*

Examples of values: *(add 1-3 examples)*	**Challenges**	**Priority:** MUST / SHOULD / MAY *(highlight match)*
	(add 1-3 challenges)	**Bonus: Reference** to existing guidelines: *(provide link or quote from the guidelines collection)*
How to improve metadata quality? *(add 1-3 ideas)*	**Difficulty level** to complete quality metadata term: easy | medium | hard *(highlight match)*	Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*

Funded by the European Union

#RS_workshop_RDAP20

author | Type: Organization or Person,

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
The author of this content or rating. Please note that author is special in that HTML 5 provides a special mechanism for indicating authorship via the rel tag. That is equivalent to this and may be used interchangeably.	You want to know the creator of the software/code, and have them credited for their work. You may want to contact the person	(ORCID, ROR, or the curator of the software	Curator, repository of software/code	Manual entry, harvesting from other repositories (ORCID, CROSSREF, ROR, github, DataCite), semantic enrichment	PIDs, structure, authority file

Examples of values:	**Challenges**	**Priority:** MUST
Research Data Alliance https://orcid.org/0000-0001-5605-9122	Disambiguation Actual names and not github or login type names ORCID profiles are nonexistent or out of date Need roles Authors may change over time with large projects	**Bonus: Reference** to existing guidelines: Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? Use PIDS for each author Structure for roles Stable contact information Ability to periodically send automated request for contact updates (using automatic harvesting)	**Difficulty level** to complete quality metadata term: easy | medium | hard Medium to Hard	

Funded by the European Union

Discussion

Actual names:

- There are pros and cons between strict disambiguation between authorName and an author identifier. Are there any use cases for requiring a "legal name" apart from copyright?
- This one is tricky. Schema.org indicates it should be a Person or an Organization. So IN THEORY, it should be a URL/resolvable identifier

Disambiguation

- ORCID is a good start as its the established identifiers for researchers and contributors

Author roles

- CREDIT taxonomy could be an option
- the author is already a role, no?
- Author isn't a role, it's a status that is linked to the question who gets credit. A role is what did you do for the software and there are many options, writing code isn't the only role.
- Here is the CodeMeta issue to introduce roles: https://github.com/codemeta/codemeta/issues/240
- Will be available in CodeMeta 2.0

Funded by the European Union

buildInstructions | Type: URL

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Provide a URL that describes step-by-step how to get the development environment set and running. “link to installation instructions/documentation”	A user needs to install or run the software locally or in its own hardware (such as a HPC, or cloud)	In the code, can be in README file	(Provider of metadata)		Existing or not Link to dependencies and other requirements (OS, hardware, etc.)

Examples of values:	**Challenges**	**Priority:** MUST / SHOULD / MAY
	Build instructions for which OS and what hardware? This type of documentation is necessary for reuse but is quite hard to compile.	**Bonus: Reference** to existing guidelines: *(provide link or quote from the guidelines collection)* Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality?	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Funded by the European Union

#RS_workshop_RDAP20

Discussion

There is an ongoing discussion in the CodeMeta community about the necessity of different types of properties to documentation. See these two issues examples:
https://github.com/codemeta/codemeta/issues/247
https://github.com/codemeta/codemeta/issues/245

Funded by the European Union

dateCreated | Type: Date or DateTime

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
A link related to this object, e.g. related web pages	Identifying when the project first started to keep a timeline of software evolution.	Extrinsic and intrinsic with version control systems	Software creator	*Automatic with version control systems like git*	- VCS for the software - dateCreated available

Examples of values:	**Challenges**	**Priority:** MUST / SHOULD / MAY
		Bonus: Reference to existing guidelines: *(provide link or quote from the guidelines collection)* Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? *(add 1-3 ideas)*	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Funded by the European Union

#RS_workshop_RDAP20

datePublished | Type: Date

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Date of first broadcast/publication.	Publishing of a given version(?) of software/code on a given platform. Can be different from creation date (embargo, publishing on a different platform,...).	Extrinsic	Repository, code development platform (researcher/ RSE sets the date if e.g. setting embargo)		

Examples of values: *2023-03-24*	**Challenges**	**Priority:** MUST / SHOULD / MAY
	- extrinsic vs. intrinsic values, may conflict - Not all repositories comply with FAIR4RS F1.2, different PIDs for different versions. Should the properties only apply to repositories that apply it? If not, does it apply to the latest release? (prone to cause confusion)	**Bonus: Reference** to existing guidelines:
How to improve metadata quality? Is the allowed format defined to avoid trouble with different date conventions? (couldn't find it) Applies to all timepoint-related values.	**Difficulty level** to complete quality metadata term: [✓ easy] | medium | hard	Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*

Funded by
the European Union

description | Type: Text

What?	Why?	Where?	Who?	How?	Bonus: Metrics
A description of the item (44): *what is the software for, what platform, language.*	Help to discover the software As a software developer I want to share information about my software with others	Intrinsic Development platform (GitHub, GitLab, etc.)	Authors - 21	Manual process when or after creating a software	Check, if possible, the presence of “objective”, “platform”, “language” ?

Examples of values: *(add 1-3 examples)* *no time	**Challenges** Should not be a “burden”, if it repeats the readme file content. How to address this : description should be less detailed, more concise (3-4 lines) / Generate the readme.file from the Codemeta.json content. So many things in the git lists it will always be a challenge to select carefully. Description is a fairly non-specific introduction whereas the readme is the action document There are different types of descriptions: e.g., 1 line summary (almost like a title) vs a 5 line description vs an abstract (what you would include in a paper)	**Priority:** MUST / **SHOULD** / MAY **Bonus: Reference** to existing guidelines: We do not know, but are interested in having some guidelines to write description - HAL - Create deposit guide: description isn’t mandatory but suggested Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match from FAIR4RS)*
How to improve metadata quality? With guidelines!	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Funded by the European Union

Discussion

The difference between readme and description

- https://github.com/codemeta/codemeta/issues/247
- `readme` is a special case of `description`,
    - Maybe worth mentioning that usually package metadata files (such as pyproject.toml / setup.py or package.json) have a description field that is exactly this... 1-4 sentences describing the project. But a similar case is for keywords, license, and developmentStatus (and possibly others). Keeping all this in sync is usually a burden...
- CodeMeta strives to keep the the number of codemeta properties small, `documentation property is preferable over` readme because it can include it

Metrics:

- Checking presence of certain keywords isn't convincing
- I would check first whether it exists or not as a sign of quality.
- what would be the sign for "quality"? How do you assess quality? For different disciplines this could be assessed very differently.

Funded by the European Union

developmentStatus | Type: Text

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Description of development status, e.g. Active, inactive, suspended. See repostatus.org	Inform the public if a software is live or outdated. Important information to decide, if I - as a researcher - want to use or build on this software for my research	Intrinsic, via README or codemeta file	Developers	Should be an explicit decision made and maintained by the developers. Could be - to a certain degree - automatically inferred by the activity logs of git repo	Check according to last commit, if information is valid

Examples of values:	**Challenges**	**Priority:** MUST / **SHOULD** / MAY
● Concept ● WIP ● Suspended ● Abandoned ● Active ● Inactive ● Unsupported. ● Moved.	If software is suddenly abandoned, information will not be updated Maturity of a research software is somehow integrated, but not really.	**Bonus: Reference** to existing guidelines: Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? Including maturity information into the repostatus list Inform developers to update this value during the software development	**Difficulty level** to complete quality metadata term: easy | **medium** | hard	

Funded by the European Union

embargoDate | Type: Date

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Date when the embargo is over	Some software might be restricted for a period of time. The users need to know when this period of restriction has ended.	Extrinsic via the repository ensuring this embargo	Authors when publishing software via any sort of repository?		

Examples of values: *Input for date when the embargo ended: 01/01/2000*	**Challenges**	**Priority:** MUST / SHOULD / **MAY (pref. MAY NOT)**
	Identifying use cases for embargo dates for software	**Bonus: Reference** to existing guidelines: *(provide link or quote from the guidelines collection)* Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | **Access** | Interoperate | **Reuse** *(highlight match)*
How to improve metadata quality?	**Difficulty level** to complete quality metadata term: **easy** | medium | hard	

Funded by the European Union

#RS_workshop_RDAP20

Discussion

We (workshop participants) would be interested in hearing about user stories related to software embargo dates, we can imagine it happening for publication or valorisation purposes, but never encountered such a situation.

Funded by
the European Union

funder | Type: Organization or Person

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
A person or organization that supports (sponsors) something through some kind of financial contribution.	To acknowledge the funder who supported the software development, to be transparent about the genesis of the software		Crossref, OpenAire, VIAF, Library of Congress AF	Manually entered into record by curator, harvest from Crossref or other source	

Examples of values: Sloan Foundation	**Challenges** Many different funders may be involved in one software, and the information about them is scattered across different places	**Priority:** MUST / SHOULD / MAY **Bonus: Reference** to existing guidelines: Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? Using authoritative PIDs Use registries and semantic artifacts Have a “funder” role	**Difficulty level** to complete quality metadata term: medium	

Funded by the European Union

identifier | Type: PropertyValue or URL

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
The identifier property represents any kind of identifier for any kind of Thing, such as ISBNs, GTIN codes, UUIDs etc. Schema.org provides dedicated properties for representing many of these, either as textual strings or as URL (URI) links. See background notes for more details.	To identify the software LJ: But in codemeta it is not clear what the "Thing" is, what the object/concept/entity described is.	Extrinsic It can become intrinsic if you add it to the code/binary files. It depends on when it is added but most likely it is created after the software starts existing Intrinsic when calculated based on the content, e.g., hash, checksum (as those in SWH).	The Who is dependant on the What, i.e. each identifier is provided by the relevant body PID provider gives the ID, probably through a publisher or repository (extrinsic) System, e.g., SWH, calculating the checksum or so It can be used by anyone, e.g., authors, to identify the software (or a version of it)	Extrinsic identifiers are provided by external actors: RRID, identifiers.org, W3ID (?), DOI, etc. Intrinsic identifiers are computed from the software itself: SWHID	The characteristics of the PID can be measured: ● Existence ● Persistence ● Unicity ● Machine actionable

Examples of values: *DOI* *SWHID* *RRID*	**Challenges** Choice, knowing which to get Granularity, e.g., version, release, modules The identifier is not necessarily in the metadata file as a property, as metadata there is always a type and an identifier so maybe no need to have it in the property	**Priority: MUST** / SHOULD / MAY
How to improve metadata quality? Make them unique, persistent, global and machine readable/actionable, complete (e.g., full DOI rather than just the last bit)	**Difficulty level** to complete quality metadata term: Easy to medium, easy to get but requires time to get it (e.g., filling in a form)	**Bonus: Reference** to existing guidelines: Archive | **Reference** | Describe | **Cite** *(highlight match from SIRS)* **Find** | **Access** | Interoperate | Reuse *(highlight match)*

Funded by the European Union

#RS_workshop_RDAP20

Discussion

- What does How mean? How is the identifier accessed? How is the identifier used?
  - How can we get this property? What's the method to have an identifier for a software artifact?
- Borrowing the Dublin Core wording as a starting point, what about "an unambiguous reference to the resource within a given context *described by the codemeta metadata*". Noting that the contexts can be many and varied (e.g., reference/citation vs retrieval). I'd say the string ideally conforms to a formal identification scheme, rather than list specific instances, because that's ambiguous as to future schemes, and schemes that are not explicitly mentioned. The scheme may or may not provide a resolver, but it should provide an unambiguous namespace.
- Are intrinsic identifiers considered intrinsic metadata even if it is stored externally in a repository?
- The key thing is SWH as an external provider of the PID rather than how it calculates it (based on a counter unrelated to the code itself or a hash that is based on the code)
  - SWHID can be calculated also locally on any machine, no need for SWH for the SWHID at the content, directory or revision granularity levels
- The key thing is the intended context of use: the purpose for which the identification system exists.
- Wouldn't manual entry be a common method?
  - even with manual entry, you need an external infrastructure to have a PID

Funded by
the European Union

keywords | Type: Text

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Keywords or tags used to describe this content. Multiple entries in a keywords list are typically delimited by commas. Topics of the resource represented by terms within literal value (string, symbol, acronym) that Well known terms of the research field identifies the ressource's features or functions, preferably in a controlled vocabulary. Should exclude items that are already mentioned elsewhere in the codemeta (like programmingLanguage, systemRequirement, OS etc.)	Help in findability of the resource, but also in the classification of it. Could be used in Associated Controlled Vocabularies/Ontologies and referred to URI	From well known terms of the Research Field Also targeting a given research community Keywords should be derived from domain specific semantic resources (thesauri, ontologies..)	The developer of the resource	The process could be a mix of automatic and manual editing. A first automatic process could be done, and then manual editing can be performed to sharpen information	The keywords have a thesaurus or and ontology connected

Examples of values: (Coming from Photon&Neutron SCientific Research field): X-Ray Spectroscopy, BioCristallography, etc (and the related scientific discipline): Quantum Physics, Chemistry: etc (and the mathematical technique used): Monte Carlo method, etc **Examples**: functions of the software, techniques, field of research.	**Challenges** Identify pertinent thesaurus Choose the right term to target the right community (be it Experts in the field, Academic researchers, Students,...)	**Priority:** MUST / SHOULD / MAY **Bonus: Reference** to existing guidelines: - HAL - Create deposit guide: keywords are recommended but not mandator Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? Try to give hints to the provider from well known ontologies in the field	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Funded by the European Union

license | Type: CreativeWork or URL

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
A license document that applies to this content, typically indicated by URL.	To make a clear statement, what are the conditions and terms that apply to use the product/software	Intrinsic via a LICENSE file(s)	Authors of the software	Usually, has to be decided at any time before publishing the software Manually added If LICENSE file(s) is existing, can be automatically extracted to extrinsic metadata	Checking, if license is in the SPDX license list Checking if license is OSI approved Checking, if license is approved by funders or institution

Examples of values: *SPDX License identifier* AGPL-3.0-or-later BSD-3-Clause	**Challenges** Little interest in PI minds about the choice of the license Poor understanding of legal implications and license compatibility Handling non-standardized licenses	**Priority:** MUST **Bonus: Reference** to existing guidelines: *Ex:* https://webcampus.pasteur.fr/upload/docs/application/pdf/2021-04/fiche_diffusion_logiciel_en_v1.pdf - HAL - Create deposit guide: license is mandatory Archive | Reference | **Describe** | Cite *(highlight match from SIRS)* Find | **Access** | **Interoperate** | **Reuse** *(highlight match)*
How to improve metadata quality? Encouraging people to use standardized licenses Using SPDX-IDs for identifying licenses	**Difficulty level** to complete quality metadata term: **easy** | medium | hard	

Funded by the European Union

name | Type: Text

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Schema : Thing The name of the item (software, Organization) (46)	To describe and identify software (31?)	Intrinsic (in codmeta) and extrinsic (repositories, etc.)	Authors/Creators	Manual	Existence or absence of the metadata Check consistency with readme and description

Examples of values:	**Challenges**	**Priority:** MUST / SHOULD / MAY
Ocaml ([OCaml · GitHub])	Finding appropriate metadata in codemeta (you need not agree!) Do we need to know, in the name, if the item is a library, a component, a whole software ? Should be : unique, not too short, not too long, not too complex, utf8	**Bonus: Reference** to existing guidelines: - HAL - [Create deposit guide]: name is mandatory Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? With guidelines !	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Discussion

- How specific should the name be to characterize the item ?

Funded by the European Union

operatingSystem | Type: Text

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Operating systems supported (Windows 7, OSX 10.6, Android 1.6).	Very important for re-use.		Maintainer of the resource	Can be provided automatically by the build system of the resource	

Examples of values: Windows 7, OSX 10.6, Android 1.6	**Challenges**	**Priority:** MUST / SHOULD / MAY
	In the new era of cloud computing, it is becoming more and more important to provide the build system (Docker, Kubernetes…) than the Operating system by itself	**Bonus: Reference** to existing guidelines: - HAL - Create deposit guide: operating system is a recommendation Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? Me automatic from a recipe for example Allow for more than one operating system and version to be listed Encourage open OS usage	**Difficulty level** to complete quality metadata term: easy | medium | hard	

programmingLanguage | Type: ComputerLanguage or Text

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
A text field that describes the programming languages and versions.	To ease the search for language specific tools To enable interoperability with other tools written in the same language To allow reuse by users who can use that language Mentioning version helps in reaching reproducibility of computational work in an embedded environment (compilation or interpretation)	Intrinsic But also, if there is some library dependency, might be extrinsic	The author or maintainer	This should be a pre populated field to choose one or many	At least one text field must be filled

Examples of values: ● Python 3.04 ● R 2.1 ● Go	**Challenges** ● Could it be more than one text? ● How to have more than one language listed? ● Do we also need to add how much percentage of the code is written in one language? Github allows for this, but many registries will only allow one value ● Some softwares are now developed as micro-services and can mix different languages (one per service)	**Priority:** MUST / SHOULD / MAY **Bonus: Reference** to existing guidelines: *(provide link or quote from the guidelines collection)* Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality? Add more detail to the text, specially if more language is sed	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Funded by the European Union

#RS_workshop_RDAP20

readme | Type: URL

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
A URL that links to the Readme file of the source code, might be linked directly to the root folder of the code repository - 68	A Readme file helps the user (person) with instructions on how to use the software. Provides a description of minimum requirements (as in other tools or libraries, test dataset) It might also link to a citation file or reference etc). It might describe the structure of your project for people to navigate it better	Extrinsic (but can appear in the metadata file as intrinsic information) Development platform (GitHub, GitLab, etc.)	The Author, or maintainer	A Readme file is usually manually created. But there can be templates to replicate ● https://www.makeareadme.com/ ● https://www.readme-templates.com/	*(Which **metric** can be used to verify this metadata and it's quality)*

Examples of values:	**Challenges**	**Priority:** MUST / [✓ SHOULD] / MAY
● *https://www.freecodecamp.org/news/how-to-write-a-good-readme-file/*	*(add 1-3 challenges)* ● A Readme file is not standard across projects. ○ not standard to have? or not standard in terms of format (plain, md, rst, ...), contents (sections to have there), location (root, subdirs) or filename? ○ This file is a bit free form and challenging for machine actionability ● *Different development environments display them in different ways depending on the format used*	**Bonus: Reference** to existing guidelines: *(provide link or quote from the guidelines collection)* Archive | Reference | [✓ Describe] | [✓ Cite] *(highlight match from SIRS)* Find | Access | [✓ Interoperate] | [✓ Reuse] *(highlight match)*
How to improve metadata quality? ● Provide a Readme template for reuse ● Automatically generate the readme file with some of Codemeta.json content	**Difficulty level** to complete quality metadata term: [✓ easy] | medium | hard	

relatedLink | Type: URL

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
A link related to this object, e.g. related web pages	To provide additional context and information located elsewhere (not directly in the project)	Extrinsic probably	Author	Manual	Valid URL check Resolvable

Examples of values:	**Challenges**	**Priority:** MUST / SHOULD / MAY
	- Decide what is still really relevant for the described project - Maintain links that work (URL does not resolve) - Linked content changes (and it is not relevant anymore) - Other than HTTP links (e.g. FTP) or links accessible only internally or after authentication - The links will be probably duplicated in README with additional context	**Bonus: Reference** to existing guidelines: Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality?	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Funded by the European Union

runtimePlatform | Type: Text

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Runtime platform or script interpreter dependencies (Example - Java v1, Python2.3, .Net Framework 3.0). Supersedes runtime.	- Use of software, reproducibility	Intrinsic, property related to the code / SW itself.	Author (programmers or maintainers), users of SW ("I got it running in virtual box…")	Depends on programming environment: many environments have a builtin mean to express that	- Try to run (or execute tests) the code in the given env. - PID resolves to env.

Examples of values:	**Challenges**	**Priority:** MUST / **SHOULD** / MAY
- Java 17 - Python >=3.8, <4, =3.4.5 - Docker - PID of env published in a dedicated repo	- Multiple values (and testing it works for all, test matrix) - No curated vocabulary (or technology-agnostic format) of all possible values - Future version of the runtime platform - Relation to `processorRequirements` and `operatingSystem` - Runtime environments do not provide a PID => must be referenceable - RT environment is a complex thing to describe, not a simple scalar property - In some cases the platform is very general and difficult to describe	**Bonus: Reference** to existing guidelines: *Comment: ("should" for now, but maybe "must" in future?)* SHOULD be part of the citation as well Archive | Reference | **Describe** | Cite *(highlight match from SIRS)* Find | Access | **Interoperate** | **Reuse** *(highlight match)*

Funded by the European Union

How to improve metadata quality? - Unambiguously referencing a runtime platform and its dependencies via a PID	**Difficulty level** to complete quality metadata term: easy | medium | **hard** - Hard	

Funded by
the European Union

#RS_workshop_RDAP20

supportingData | Type: DataFeed

https://schema.org/supportingData

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics

Examples of values:	**Challenges**	**Priority:** MUST / SHOULD / MAY
		Bonus: Reference to existing guidelines: Archive | Reference | Describe | Cite *(highlight match from SIRS)* Find | Access | Interoperate | Reuse *(highlight match)*
How to improve metadata quality?	**Difficulty level** to complete quality metadata term: easy | medium | hard	

Funded by
the European Union

#RS_workshop_RDAP20

version | Type: Number or Text (see also SoftwareVersion)

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
The version of the CreativeWork embodied by a specified resource. SoftwareVersion - Version of the software instance. (Type-Text)	- Is this version current one - Encoded info in version string about major/minor release… - Citing the exact version that was used for research - Reproducibility of experiments (bugs…) - Mark a milestone	Intristric, in code repository (tag) or constant, package metadata (e.g. pom file) In the release artifact	- Auto-generated - Author or maintainer of repo	- Auto-generated (e.g. git tag)	- Using consistently versioning schema (regex match?) - Consistency across repository / codebase (matches git tag?)

Examples of values:	**Challenges**	**Priority:** MUST / SHOULD / MAY
- v3.10.1-rc.2 - 2022.7.1 - 1.0	- Different versioning schemas (CalVer, SemVer, etc.) - Should there be “v” prefix before semver in metadata or not? - Consistency with tags and version constants in the software (e.g. “tool --version”) - Keeping the information up to date (automation?) - Version string in filenames, codemeta.json… - Version is volatile metadata - Auto-generated or reviewed/added by humans - Name changes - Find current version - Branches - Forks - Follow Best practices vs. institutional policies - Not marking different code with the same version	**Bonus: Reference** to existing guidelines: Archive | **Reference** | Describe | Cite *(highlight match from SIRS)* **Find** | **Access** | **Interoperate** | **Reuse** *(highlight match)*

Funded by the European Union

#RS_workshop_RDAP20

How to improve metadata quality? *(add 1-3 ideas)* *Include provenance to previous published versions* Possibility to test the code… to see if it is working?	**Difficulty level** to complete quality metadata term: easy | **medium** | hard If a consistent approach is selected.. Agree - a best practice that works on a scale of decreasing consistency: ● If the git/ repository provides a version number automatically, then always use this. ● If not, then …	

Funded by the European Union

softwareRequirements | Type: SoftwareSourceCode

What?	**Why?**	**Where?**	**Who?**	**How?**	**Bonus:** Metrics
Required software dependencies	To ensure all the pre-requisites for the software are available before re-use To inform about dependencies of the Resource	Must be taken from the build system of the software	The developer of the software	Manual information added	

Examples of values:	**Challenges**	**Priority:** MUST / SHOULD / MAY
	There quit a lot of kinds of dependencies: Algorithmics (which numerical method,...ect) Software development frameworks Libraries (with a variety of maturity levels) Versions of Languages/platforms	**Bonus: Reference** to existing guidelines: **Archive** | Reference | **Describe** | **Cite** *(highlight match from SIRS)* Find | **Access** | **Interoperate** | **Reuse** *(highlight match)*
How to improve metadata quality?	**Difficulty level** to complete quality metadata term: easy | **medium** | hard	

Funded by the European Union

#RS_workshop_RDAP20