FAIR4RS

Atelier RDA France

Collaborative notes - 28 June 2021

14:00 CEST

Useful links

RDA group page, GitHub repository

- These slides https://tinyurl.com/RDA-France-FAIR4RS-slides
- Collaborative notes https://tinyurl.com/RDA-France-FAIR4RS

How to get involved?

- Join the RDA group and be part of the mailing list
- Come to events
- Follow the steering committee meeting minutes
- Say 'Hi' on the gitter channel
- Visit and read the publications on Zenodo
- Review the bibliography collected on Zotero

All this information is detailed on the community engagement channels page

Participants

- ***Add your name, institution, job title, country, member***
  Click Join Group to become a member
  https://www.rd-alliance.org/groups/fair-4-research-software-fair4rs-wg
  This is the preferred way for communication with the community. For more information to get involved look here.

Name	***Institution/affiliation***	***Job title***	***Country***	***FAIRS4RS member***
Louys Mireille	Strasbourg , Icube , CDS	Assistant professor	F	
SABATIÉ Sandrine	INRAE, Bordeaux	IT Software Engineer	France	yes
Romier Geneviève	CNRS CC-IN2P3	Ingénieur de recherche	France	non
Jean-Christophe SOUPLET	OpenEdition (CNRS/AMU/EHESS/UA)	Resp. secteur informatique	FR	non
Clement Jonquet	INRAE / UMontpellier	Researcher	France	no
Marc-Antoine Drouin	IPSL/CNRS	ingenieur	France	non
Pierre Navaro	IRMAR / CNRS	Ingénieur	France	non
Mathieu Servillat	LUTH - Observatoire de Paris	Ingénieur de Recherche	France	oui
Jonathan Mineau	INRAE - CIRM	IE - Data manager	France	Non
Michelle Sibilla	Univ. Toulouse 3, IRIT	Professeur	France	non
Franck Le Petit	Observatoire de Paris	Astronome	France	non
Chrystel Moreau	LAM, Marseille	Ingenieure de Recherche	France	non
Capdeville	CNRS, INSU	DR	Fr	non

Yann	(CS-TS), Université de Nantes, CCIPL, GLiCID			
Rossi Albane	UBFC, Besançon	Chargée de valorisation projet dat@UBFC	Fr	Non
Éric Buchlin	CNRS	Chargé de recherche	France	Non
Da Rocha Martine	INRAE/ Plateforme PlantBIos	IE bioinformaticienne	France	Non
Arènes Cécile	Bibliothèque de Sorbonne Université	Chargée de mission Données de la recherche	France	Non
Fiocca Amélie	URFM, INRAE, Avignon, Future DipSO	IE informatique et connaissances	France	Oui (new)
Doux Gwenaël	Cirad, DGDRS-Dist	Administrateur fonctionnel Dataverse	France	
Debray Bernard	CNRS, Institut UTINAM, Besançon	Ingénieur recherche informatique	France	Non
Delage Emmanuel	OPGC Observatoire de Clermont	IE Calcul	France	Non
Céline Blitz Frayret	CIRAD, UMR Eco&Sols	Ingénieur de recherche dev. Info et calcul scientifique	France	oui
Fabien Blarel	CNRS, LEGOS, SNO CTOH	Ingénieur d'Etude	France	oui
Laure Vidaud	INRAE, Etna, Grenoble	Ingénieur	France	non
Françoise	Observatoire	Chercheur	France	oui

Genova	astronomique de Strasbourg	émérite		
Violaine Louvet	GRICAD / CNRS - Grenoble	Ingénieur de Recherche	France	Non
Maria Grazia Santangelo	Bibliothèque Universitaire Joseph-Fourier, Grenoble	Chargée Projet CollEx-Physique	France	Non
Fournier Alexandre	IPGP	Pr	France	Non
Christelle DANTEC	CRBM	Ingénieure de Recherche	FRANCE	non
Ligia Onetto	ARMINES - IC MINES	Chargée PI / Valorisation	France	Non
Nicolas Moreau	Observatoire de Paris, LERMA	Ingénieur	France	non
Barde Julien	IRD, UMR MARBEC	Ingénieur	France	Non
Sophie BOULARD	ARMINES	Juriste	France	Non
DAGUZE Katie	Sorbonne Université	Archiviste	France	non
GOUAT Isabelle	CNRS/UMontpellier	Ingénieur documentaliste	France	non
Poulleau Gilles	CNRS / Paris Saclay /IDOC	Ingénieur	France	Non
OBERTO Anaïs	CNRS. Observatoire astronomique de Strasbourg	Ingénieur logiciel	France	Non

Q & A : Questions & Réponses

Name or identifier **Nom ou identifiant**	Questions **Quehttps://docs.google.com/docu**	Answers **Réponses**

	ment/d/19NCSJuPJiAVPb0tclfHJoGURYpEV09o7n7aFbdbhxAA/edit#stions	
JC Souplet	Dans la définition “logiciel de/dans recherche” je ne suis pas sûr d’avoir compris où sont les logiciels développés pour permettre la recherche (site de récolte de données, site de diffusion d’articles) ? Par exemple les logiciels développés pour le service HAL entrent-ils dans la définition ? Où arrêtez- vous le travail de recherche (cela va-t-il jusqu’à la diffusion) ?	
Sandrine Sabatié	Est-ce que ces principes ne sont pas liés et à adapter au niveau de maturité des logiciels qui n’a pas été abordé ?	? à aborder dans les guidelines

Activity

Ice-breaker

Name or identifier **Nom ou identifiant**	Are the FAIR principles (Wilkinson et al.) relevant to Software ? **Les principes FAIR (Wilkinson et al.) sont-ils pertinents pour les logiciels ?**
Francoise Genova	Les principes FAIR sont utiles comme guide, à adapter éventuellement au cas des logiciels. Find, Access, Interoperate, Reuse sont des éléments importants pour faciliter la pratique de la recherche. Pas seulement la reproductibilité mais aussi pour la réutilisation des logiciels par d'autres que ceux qui les ont créées
JC Souplet	Les principes guident la "qualité" des données produites par les logiciels. Les Principes ont donc un impact fort sur les logiciels mais ils ne définissent pas directement les logiciels selon moi (dans leur définition de base).
Sandrine Sabatié	Ces principes doivent permettre de favoriser la science ouverte dans le domaine des logiciels, doivent être sûrement un peu adaptés car ils ont été édictés en première intention pour les données.
Amélie Fiocca	Les principes FAIR sont pertinents pour la reproductibilité de la recherche et surtout pour la gestion de la provenance. Les principes FAIR n'attestent pas forcément de la qualité des logiciels mais permettent de la vérifier.
Jérôme Pansanel	Les principes FAIR sont importants pour les codes de recherche, car tout comme l'accès à la donnée FAIR, ils sont nécessaires à la reproductibilité des résultats
E Gondet	
Mathieu Servillat	Oui, mais ils ne sont pas suffisants, notamment la partie interopérable (interfaces logicielles), et réutilisable (environnement d'exécution)
Emmanuel Delage	DOI de logiciels scientifiques
Michelle Sibilla	Interêts pour les trouver, les réutiliser et les étendre au sein d'une communauté (notion de bibliothèque) mais problèmes de leur évolution, de documentation, ...d'interopérabilité
Fabien blarel	Traçabilité du code et des données. Cela va de pair. Je dirais oui, c'est important.

Mireille Louys	Oui pour le partage dans une communauté qui pratique le logiciel libre, qui développe beaucoup, qui cherche des citations
B.Debray	Les logiciels (scientifiques) doivent être faciles à trouver, accessibles, réutilisables (-> métadonnées) - il faudrait voir ce qu'interopérable veut dire pour un logiciel
Violaine Louvet	Une bonne base de départ mais sûrement ne prenant pas en compte toutes les spécificités du logiciel
Céline Blitz Frayret	Les données sont des objets fixes alors que les logiciels sont évolutifs donc les principes FAIR doivent être effectivement adaptés.
Martine Da Rocha	Oui pour simplifier le partage et l'échange des logiciels. Favoriser leur utilisation dans la durée.
Simon Moré	Oui par exemple pour tout ce qui est metadonnées (What, When, Where, Who, Why, How). notamment un Readme FAIR
	Pas clair de savoir ce qu'il faut rendre public / FAIR. Est-ce juste la version du code qui a servi à publier un article ou bien la version générale du code qui peut servir à traiter d'autres problèmes (et pour lesquels le code n'a pas forcément été testé) ?
Clement Jonquet	Oui. Mais évidemment ils nécessitent une petite projection pour cela.

Q1: Defining the scope

(see figure 3 in Appendix A)

Name or identifier **Nom ou identifiant**	What is Research Software? **Quel est le Logiciel de Recherche?**
JC Souplet	Cela a toujours été un grand débat ou “notion incertaine”. Votre diapositive le définissant en trois sous-catégories au début de votre présentation me semble apporter la réponse. Après si on parle des composants, dépendances on arrive à l’épineuse question de qu’est-ce qu’un logiciel ?
Fabien Blarel	Logiciel de recherche est un code prototype pour explorer une idée.
Amélie Fiocca	Le logiciel de recherche peut être toute ligne de code, toute routine simple ou système complexe susceptible de modifier la donnée brute et de l’analyser en vue de produire un résultat. L’accessibilité des logiciels doit être décrite en fonction de leur degré d’ouverture.
Sandrine Sabatié	Logiciel de recherche est un logiciel qui a vocation à évoluer j’imagine en regard de ce qui est produit dans les projets de recherche, donc c’est un outil de valorisation de la recherche. Peut être un logiciel très finalisée, ou seulement un prototype / poc.
Francoise Genova	A définir au cas par cas en réfléchissant au pourquoi (pareil pour les données!).
	Un logiciel commercial peut être un logiciel de recherche…
Céline Blitz Frayret	Logiciel DE recherche : donne les valeurs numériques d’une question scientifique ? Logiciel DANS la recherche : améliore la visu, met en valeur certains résultats numériques, traitement de données ?
	Logiciel de recherche traitement de donnée en l’analysant pour extraire de nouvelles informations
Michelle Sibilla	Logiciel de Rech : Implémentation de solutions conceptuelles à un problème scientifique apportant un traitement automatique de données (une preuve de concept, prototype, ...) et restituant des résultats numériques, visualisables Logiciel dans la recherche : environnement de travail d’exécution
Mathieu Servillat	Une sous-partie serait déjà les logiciels manipulant des données

	scientifiques. On peut aussi considérer les logiciels destinés à être publiés (et réutilisés)... Faut-il appliquer les principes FAIRS4RS à tout logiciel lié à la recherche ?
Maria Grazia Santangelo	Le logiciel de recherche nous permet d'acquérir des données et de les analyser
Simon Moré	Tout ce qui est nécessaire pour pouvoir reproduire la recherche Ce qui peut servir à d'autres
Mireille Louys	C'est un logiciel utilisé pour fournir des données pour une communauté de recherche et élaborer des publications . Ses propriétes utiles seraient : Partagé, Decouvrable , Publié, maintenu si possible
Jérôme Pansanel	Le logiciel de recherche est le code ou un ensemble de codes permettant de traiter, d'analyser et/ou générer des données de recherche, produire des résultats et répondre à des questions scientifiques

Q1.2 Feedback on the Research Software definition:

Commentaires sur la définition du logiciel de recherche:

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 specifically for research should be considered software in research and not Research Software. This differentiation may vary between disciplines. The minimal requirement for achieving computational reproducibility is that all the computational components (Research Software, software used in research, and hardware) used during the research are identified, described, and made accessible to the extent that is possible.

F Genova: Dans ma discipline il y a aussi des codes partagés y compris dans une large sous-communauté - utilisés par les chercheurs pour leur propre recherche pas seulement pour la reproductibilté

M. Servillat : pourquoi intégrer à la définition un “minimal requirement” ? cela serait plutôt un objectif

JC Souplet : Dans la définition “logiciel de/dans recherche” je ne suis pas sûr d’avoir compris où sont les logiciels développés pour permettre la recherche (site de récolte de données, site de diffusion d’articles) ?

Q2: Software granularity and identifiers

Name or identifier **Nom ou identifiant**	Should the FAIR principles for research software care about the levels of granularity identifiers should be assigned? **Les principes FAIR, Devraient-ils inclure la notion de granularité des identifiants du logiciel?**	If so, which are the most useful granularity levels to ensure the findability of software? **Si oui, quels sont les niveaux de granularité les plus pertinents?**
Francoise Genova	La question de la granularité des identifiants se pose aussi pour les données. Pas sure du tout qu’il faut que la question soit tranchée au niveau des principes, plutôt dans les guidelines. Ca peut être du cas par cas et si on tranche dans les principes ça met en danger le fait que le principe doit être général.	
Mireille Louys	Oui	Si le code est distribué sous forme de Package distribuable , on peut identifier une version avec toute ses

		dépendances (libraries , etc) comme la granularité adaptée . Le niveau fichier est trop fin
	Plutôt d'accord avec l'assignation d'un identifiant pour chaque version d'un logiciel. Concernant la granularité, cela dépend du logiciel, de sa complexité et de ses liens avec d'autres logiciels (intégrés ou non).	
	Oui mais est-ce que la granularité à toutes les étapes est importante ? on peut faire l'économie de certains étages	
	Si le soft est associé a un traitement de données, il faudrait un id a chaque phases de données	
	Avoir un identifiant par version est utile mais cela suppose que le logiciel ait des versions bien identifiees et non une succession de comits sur un Github (ou alors les comits doivent avoir des identifiants ce qui peut etre lourd)	
	Il serait intéressant de pouvoir identifier un "logiciel" de manière unique (création) et repérer ses versions. Dépend des objectifs/contextes	
Mathieu Servillat	Oui, les niveaux semblent en effet suffisamment universels	Peut-être pas nécessaire de descendre à un niveau trop bas : on veut en général réutiliser un logiciel qui est déjà fonctionnel, ou des briques logicielles bien interfacées, pas vraiment des bouts de code.
	Si on identifie tous ces objets, que doit-on transmettre à un éditeur qui voudrait juste un DOI du code source utilisé ? sinon je sui d'accord car il faut tout les objets pour que le code soit re utilisable	

JC Souplet	OUI	Mais je ferais une différence entre version de dev (id perenne par gestionnaire de code) et version de “production/diffusion” (labelisation/ tag avec identification dans un référenciel externe)
	Qu’appelle-t-on une version de logiciel dans le cas d’un logiciel qui est composé d’un code numérique + un très large ensemble de données que le code utilise ? L’ensemble ? Faut-il avoir des identifiants pour le code et des données en entrée de façon séparées ?	
	Si on considère que l’objectif ultime du FAIR est le R, il faut considérer les choses et donc la granularité sous cet angle là. Un DOI par version n’a pas trop de sens par rapport à la façon dont vivent les codes.	

Q3: Long term accessibility

Name or identifier **Nom ou identifiant**	Should software preservation be part of the FAIR principle? **Est-ce-que l'archivage du logiciel devrait faire partie des principes FAIR?**
	C'est un peu illusoire à long terme, il ne restera que des sources non interoperables
	Si l'OS (voir aussi le hardware dans certains cas - par exemple pour les logiciels embarqués) le sont aussi pourquoi pas.
	Je dirai oui mais qu'est-ce qu'un logiciel archivé : un logiciel mort ? qui n'évolue plus ?
	Si l'on veut capitaliser et construire un "héritage" scientifique, cela devrait passer par cet archivage mais sur quels critères (lié à des publications, un taux de (re)utilisation, … -> précisions pouvant être renseignées par les méta-données
	Ce serait l'idéal, mais il faudrait être sûr d'avoir les moyens qui permettent d'archiver sur plus de 20-30 ans. C'est la même question pour les données, il faut voir si c'est rentable en fonction de ce que le logiciel apporte à la recherche, mais c'est compliqué à estimer car des fois un code/données reste en attente plusieurs années avant d'avoir un intérêt
	Sans archivage, comment garantir le F&A de FAIR ?
	L'archivage devrait faire partie des principes FAIR, mais cela ne veut pas dire que l'archivage n'a pas de durée de vie. Il faut renseigner les informations sur cette durée de vie et les raisons de la suppression de certains logiciels. Si un logiciel disparaît des archives, un accès à sa description et ce qui peut le remplacer serait intéressant.
	Comme pour l'archivage des données, il faut vraiment se poser des questions du coût à long terme d'un archivage de tout ce qui est produit : coût financier mais aussi coût environnemental. Il faut aussi accepter le principe de fin de vie des données et des logiciels.
	Si on trouve un bug dans un logiciel faut-il modifier l'archive ou re-documenter l'archive ?
	Un logiciel a un cycle de vie, et il peut devenir inutilisable (dépendances et environnement impossibles à reproduire…). Archivage = maintien des métadonnées seulement ?

	Oui , c'est le terme Findable / on doit pouvoir le citer : article , etc et le découvrir pour pouvoir le charger, l'exécuter et tester . Donc il faut aussi connaître l' environnement d' exécution, ressources, plateforme etc .
	Il y a la question de la préservation "patrimoniale" de logiciels (un des buts du projet "Software Heritage" ?) -> fait partie de l'accessibilité à long terme ; intérêt de pouvoir réaccéder à un logiciel "longtemps après" par exemple quand il y a besoin d'accéder à des données anciennes, non utilisées depuis longtemps - un lien aussi avec la dimension "R" (Réutilisable)
	La dernière version est suffisante
	Ca dépend, si la version est buggé, ce n'est peut etre pas une bonne idée de le garder accessible. Version stable + release majeure => oui.
	Les version associées à un ID
	L'accès au logiciel "binaire", "code" mais aussi doc
	Si on veut une trace des codes ayant servi a des publications, on a besoin d'archiver la version effectivement utilisee bugguee ou pas. Mais dans ce cas, le code risque d'etre utilise dans de mauvais contextes.

Q4: Use and Re-Use

Name or identifier **Nom ou identifiant**	Should FAIR software be re-executable? **Le logiciel FAIR doit-il être ré-exécutable ?**
	Non. C'est le source qui contient la connaissance utile et pérenne. Même avec une documentation incompléte, on peut faire du retro engineering.

	A quoi ça sert de partager un logiciel s'il n'est pas ré-exécutable? Est-ce qu'il ne faut pas accepter que tous les logicels ne soient pas FAIR in fine? Si le logiciel n'est pas documenté il ne peut pas être FAIR. La question de l'environnement est plus complexe. Si on parle d'algorithme seulement le logiciel est seulement une donnée…
	Oui car le principe de reproductibilité porte cette valeur de ré-exécution avec notion d'environnement d'exécution (packaging, container).
	Selon le niveau d'intégration du logiciel , il va être difficile de ré-éxécuter simplement un logiciel. La réutilisation peut être dans l'amélioration ou juste l'adaptation à son besoin. Le logiciel peut être transformé dans un langage différent par ex.
	Oui mais en l'imposant sur un cas test (environnement précis, jeux de données)
	La re-exécution n'est pas suffisante si nous n'avons pas la garantie de la validité scientifique des résultats issus de l'exécution
	Pas forcement si les metadonnées sur l'environnement (hardware, OS, etc…) sont completes
	Logiquement oui mais encore faut-il avoir les détails d'exécution. Les méta-données peuvent-elles nous le garantir ? Une obligation ? pas forcément, pourrait-être intéressant pour faire du reverse engineering ?
	Un effort doit etre fait dans ce sens pour pouvoir reproduire et valider des resultats scientifiques. Ensuite, il faut faire du cas par cas car dans certains cas, cela peut etre difficile.
	Il peut être associer à l'environnement adapté via des conteneurs / associé à fichier de définition des images des conteneurs / avec le code source disponible aussi.
	Compte tenu des réflexions sur la granularité, clairement ce ne sera pas toujours possible.
	Un logiciel qui n'est pas exécutable ne sert pas à grand chose...
	Oui, pour reproduire tout le processus
	Quelle licence pour les codes ? Existe-t-il des licences permettant de contraindre ce qu'on peut faire avec ? Par exemple, on accepte de

	rendre public un code pour un certain usage mais pas d'autres ?
	Un algorithme peut il être évolutif ?

Q5: Software dependencies

Name or identifier **Nom ou identifiant**	Should FAIR be recursive? **Le logiciel FAIR doit-il être récursivement FAIR ?**
	Idem Q4,donc non
	Si possible mais on a pas la main sur tout
	A cause des différents cas possibles , on dirait qu'un parametre 'type de re-exececutablility' doit etre prévu : - including dependencies , - Algorithmically re executable, - citable only, - not re executable
	A priori pourrait être envisageable progressivement, serait logiquement appréciable.
	Non, nous n'avons pas forcément les moyens de maitriser toute la chaine
	Besoin sûrement , est-ce que c'est possible … probablement pas
	Ce n'est pas envisageable, on ne peut pas tout maîtriser… ou alors il faut travailler en vase clos et ce n'est pas le principe de la recherche...

	Difficile à envisager de façon pragmatique ! Il y a des débuts de réponses dans les outils comme guix ou nix.
	Si c'est "doit", la réponse est non car qu'est-ce qu'une dépendance (code compilé ?, service externe appelé ?, script appelé ? autre ??) ?
	(venant du chat) Va dépendre de l'utilisateur: un chercheur ou un développeur. Un développeur va pouvoir réutiliser les logiciels car il a les compétences pour le faire.
	Il faudrait diffuser les bonnes pratiques en développement logiciel pour assurer la faisabilité de la récursivité

Q6: Who is responsible for FAIR software?

Name or identifier **Nom ou identifiant**	Who is responsible for FAIR software? **Qui est responsable du logiciel FAIR ?**
	L'employeur ? le développeur ? le chercheur ? les 3 ? cf. droits patrimoniaux...
	Qui est responsable des données FAIR ?

	Qui décide quel logiciel doit etre fair?
	Tous les acteurs de la communauté des instituts. Il faut accompagner la communauté par la formation et le développement des compétences sur ces questions.
	Quelle est la définition d'un data steward ? -> Data Officer ?
	Responsabilité partagée et coordonnée

Q7: FAIR and FOSS

Name or identifier **Nom ou identifiant**	Should FAIR require software to have a free and open source (FOSS) license? Why or why not? **Devrait-on exiger que le logiciel FAIR ait une licence libre et open source (FOSS) ? Pourquoi ou pourquoi pas?**
	Non. Les chercheurs développant leur code doivent aussi pouvoir en bénéficier pendant un “temps” comme pour les données propriétaires pendant un temps. Ils peuvent donc vouloir rendre disponible le logiciel pour certains usages mais pas d’autres.
	Non pour la reconnaissance des développeurs afin d’éviter qu’une équipe récupère un logiciel et le re-publie de son côté en y changeant trois bricoles. Il faut pouvoir controler cela en diffusant du logiciel mais avec des licences souples qui précisent ce qu’on peut faire avec ou non.
	Pas d’exigence, cela se décide au cas par cas et selon les projets de recherche et financeur (en parallèle des jeux de données sont fermés alors qu’ils sont identifiés et ont des méta-données)
	A quoi sert qu’il soit FAIR s’il n’est pas open source?
	On peut vouloir montrer le code source pour donner de la confiance dans ses résultats mais pas autoriser à récupérer le code pour en faire autre chose;
	Certains logiciels peuvent être sensibles donc il est difficile d’exiger l’open source. Cependant, il faut rendre la description du logiciel accessible avec des descriptions basées sur des standards de métadonnées dédiés. L’ouverture peut être conditionnée à certains utilisateurs en fonction du degré d’implication sur le projet.
	Plus généralement FAIR est différent de Open (voir Turning FAIR into Reality, figure 4 p.21)
	Très communauté dépendant comme positionnement. Mais un logiciel

	open source est clairement plus réutilisable à long terme.

Feedback

Thanks for joining us!!!

Let us know your thoughts of this session, we are keen to improve.

- [name] : feedback
-

Appendices

Appendix A - Additional Figures

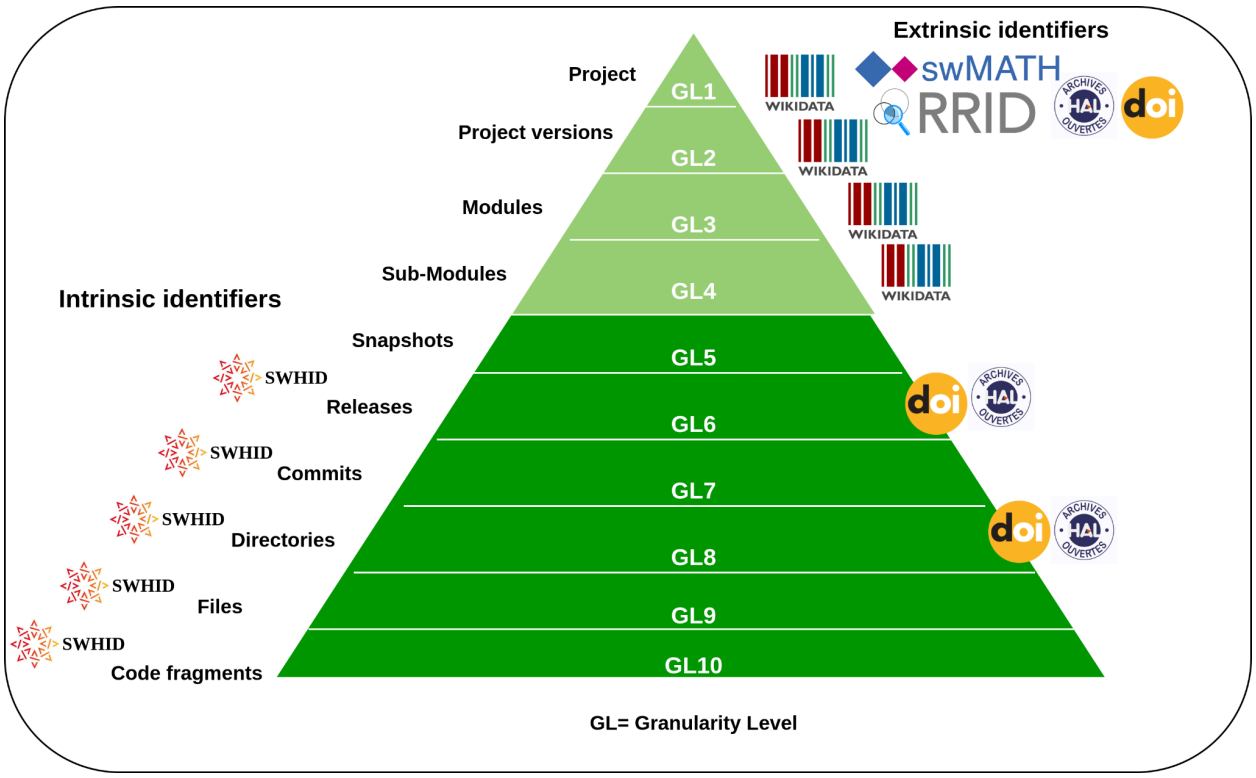


Figure 1: Granularity levels and identifiers currently in use for software [based on granularity levels definition from (RDA/FORCE11 SSCID WG et al., 2020)]

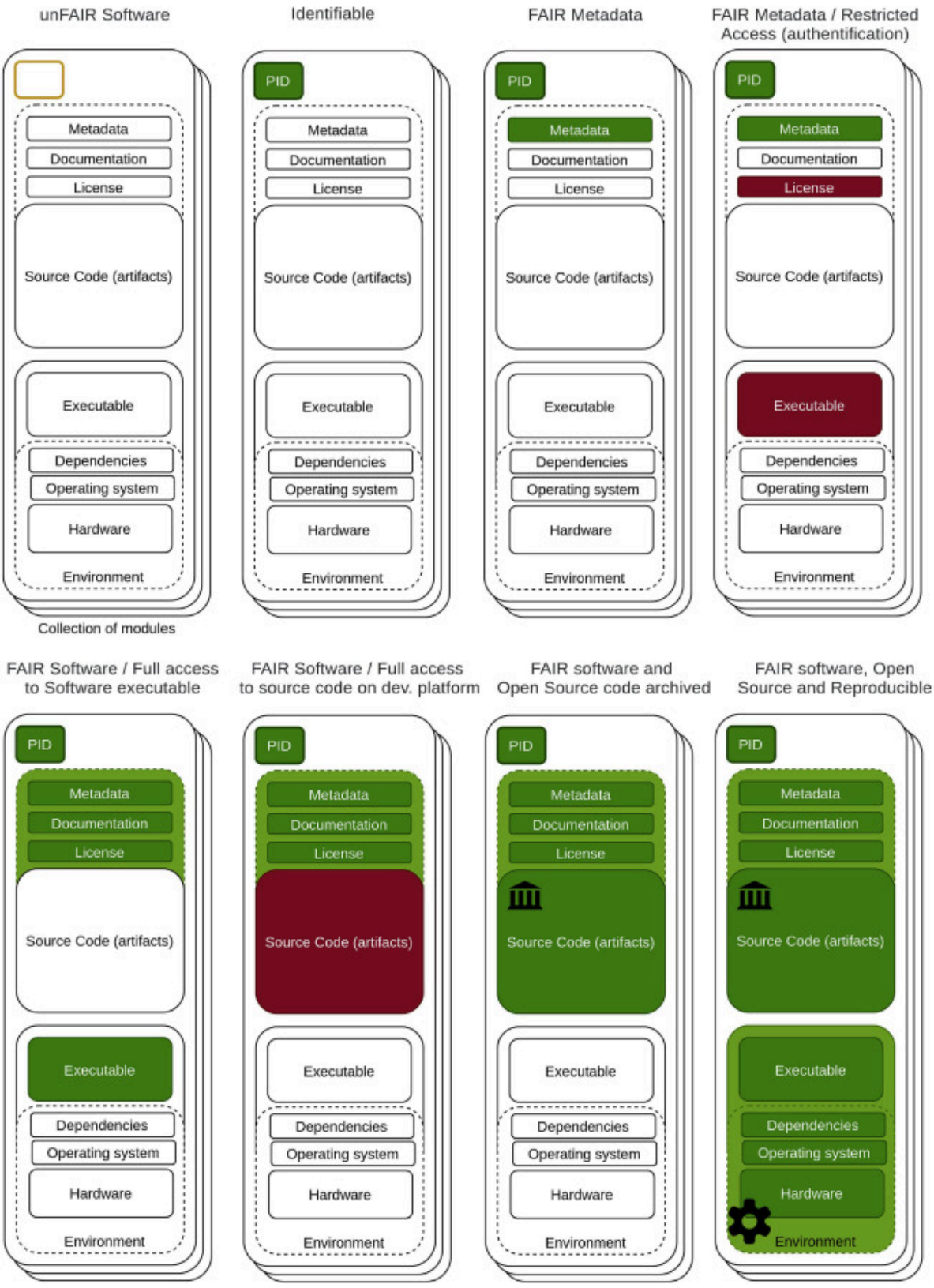
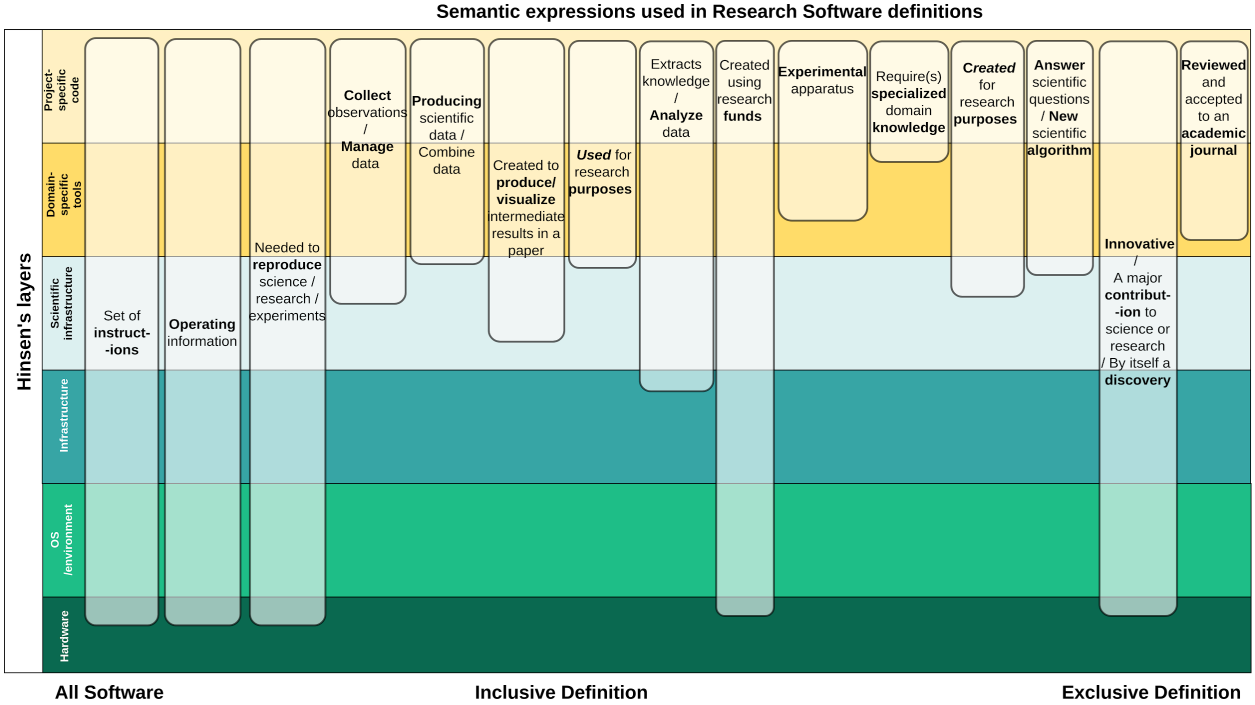

unFAIR Software
Identifiable
FAIR Metadata
FAIR Metadata / Restricted Access (authentification)
PID
Metadata
Documentation
License
Source Code (artifacts)
Executable
Dependencies
Operating system
Hardware
Environment
Collection of modules
FAIR Software / Full access to Software executable
FAIR Software / Full access to source code on dev. platform
FAIR software and Open Source code archived
FAIR software, Open Source and Reproducible

Figure 2: Summarizing software as increasingly FAIR research objects (Katz, Gruenpeter & Honeyman, 2021)


Semantic expressions used in Research Software definitions
Hinsen's layers
Project-specific code
Domain-specific tools
Scientific infrastructure
Infrastructure
OS /environment
Hardware
Set of instruct--ions
Operating information
Needed to reproduce science / research / experiments
Collect observations / Manage data
Producing scientific data / Combine data
Created to produce/ visualize intermediate results in a paper
Used for research purposes
Extracts knowledge / Analyze data
Created using research funds
Experimental apparatus
Require(s) specialized domain knowledge
Created for research purposes
Answer scientific questions / New scientific algorithm
Innovative / A major contribut--ion to science or research / By itself a discovery
Reviewed and accepted to an academic journal
All Software
Inclusive Definition
Exclusive Definition

Appendix B - Comparison of FAIR Principles

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) and the FAIR4RS Principles from the community consultation draft.

FAIR Guiding Principles (2016)	**FAIR4RS Principles (2021)**
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 software, and its associated metadata, should be easy to find for both humans and machines.
F1. (Meta)data are assigned a globally unique and persistent identifier	F1. Software is assigned a globally unique and persistent identifier.
	F1.1. Different components of the software must be assigned distinct identifiers representing different levels of granularity.
	F1.2. Different versions of the same software must be assigned distinct identifiers.
F2. Data are described with rich metadata (defined by R1 below)	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 the identifier of the software they describe.
F4. (Meta)data are registered or indexed in a searchable resource	F4. Metadata are FAIR and is 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 authorisation.	The software, and its metadata, must be retrievable via standardized protocols.
A1. (Meta)data are retrievable by their 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.
A1.2. 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. 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.	The software interoperates with other software through exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs).
I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.	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	
I3. (Meta)data include qualified references to other (meta)data	I2. Software includes qualified references to other objects.
R. Reusable	

The ultimate goal of FAIR is to optimize the reuse of data. To achieve this, metadata and data should be well-described so that they can be replicated and/or combined in different settings.	The software is both usable (it can be executed) and reusable (it can be understood, modified, built upon, or incorporated into other software).
R1. (Meta)data are richly 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 must have a clear and accessible license.
R1.2. (Meta)data are associated with detailed provenance	R1.2. Software is associated with detailed provenance.
R1.3. (Meta)data meet domain-relevant community standards	R3. Software meets domain-relevant community standards.
	R2. Software includes qualified references to other software.