Guidelines for Managing and Sharing Research Artefacts in SE

In this document, we report a list of practices for managing and sharing research artifacts extracted from guidelines recommended by major computer science publishers, venues, and organizations. The set of guidelines analyzed is composed by the following documents:

1. The ACM Artifact Review and Badging (ACM, 2020)
2. The EMSE OpenScience initiative (EMSE, 2021a; Méndez Fernández et al., 2019)
3. The Journal of Open Science Software (JOSS) (Katz et al., 2018)
4. The Journal of Open Research Software (JORS) (JORS, 2021)
5. The Guidelines by Wilson et al. (Wilson et al., 2017)
6. The NASA Open Source Software Projects (NASA, 2021)
7. The TACAS artifact evaluation guidelines (TACAS, 2019)
8. The CAV artifact evaluation guidelines (CAV, 2019)

# The ACM Artifact Review and Badging

Below we report a list of practices and properties extracted from (ACM, 2020)

* Includes a software system developed to the experiment (e.g., tool)
* Includes scripts to automate the execution of experiments
* Includes sample input/output data
* Includes raw data collected
* Includes scripts to automate the analysis of results
* Provides a description about the artifact
* Provides a description of the measurement procedures (e.g., tools, sampling rate)
* Provides a description of the measuring system (e.g., units, metrics)
* Describes the operating environment (e.g., CPU, RAM, GPU)
* Describes how to obtain proprietary/third-party artifacts
* Provides documentation to guide reuse/repurposing
* Indicates norms and standards used
* Permanently available for retrieval
* Archived in a public repository
* The package has DOI
* The package has a URL
* Includes a description of the relevance to the study
* Includes the data from which the figures were drawn
* Includes implementations of the algorithms in the article
* Includes the complete set of software under study
* Includes a description of the experiment protocol
* Includes a description of how the artifacts can be exercised
* The artifacts are well-formed, documented, consistent, complete

# The EMSE Open Science initiative

Below we report a list of practices and properties extracted from (Méndez Fernández et al., 2019)

* Includes raw data collected
* Includes the material necessary for the analysis and interpretation of raw data
* Includes the study protocols
* Includes scripts to automate the analysis of results
* Includes an open data set

Below we report a list of practices and properties extracted from (EMSE, 2021a)

* Includes an open data set
* Includes source code
* Archived in a public repository
* Includes an explicit/short statement about why some artifact was not disclosured
* Permanently available for retrieval
* Uses an appropriate license
* Linked to the papers (e.g., DOI, URL) what used the artifacts
* Can be updated any time
* Provides reasons for confidentiality of data or artefacts
* Includes UML diagrams/models
* Includes requirements text
* Includes design documents
* Provided in machine-readable and open formats (e.g., JSON, XMI)

Below we report a list of practices and properties extracted from (EMSE, 2021b)

* Includes source code and data
* Lies behind a single URL (e.g., centralized repository)
* Hosted on a persistent, archived repository
* Provides an inventory of artifacts (e.g., files, folders)
* Documents file formats used
* Indicates naming conventions used
* Includes everything required to understand the paper
* Includes everything required to re-compute data, numbers and figures in the paper
* Includes instructions to compile the code
* Includes instructions to execute the code
* Dependencies are publicly available modules/libraries
* Uses appropriate license

Below we report a list of practices and properties extracted from (Monperrus, 2019).

* Includes associated data, code, or both required to study and reproduce the results of a published paper
* Provide means to fully understand the contribution
* Provide means to verify the claims
* Provide means to build upon the results
* Referred to as Replication package, Laboratory package, supplementary material, Online appendix
* Artifact is downloadable: data and code lies behind a single URL (e.g., DOI)
* Artifact is findable: Well-indexed repository (e.g., findable by search engines based on the paper title)
* Artifact is documented: inventory of artifacts (files and folders),
* Documents used file formats
* Documents naming conventions
* Artifact is complete: numbers (i.e., data) and figures can be re-computed
* Artifact is exercisable: code must be executable, does not depend on non-publicly available modules and libraries
* Artifact is durable: Repository URL must be stable in the long term (e.g., 10-100 years? More? – See GitHub Arctic Code Vault project <https://archiveprogram.github.com/>)
  + Zenodo by CERN
  + FigShare
  + HAL by CNRS
  + archive.org by the Internet Archive foundation
  + OSF.io by the Center for Open Science
  + GitHub as working repo + Zenodo (backup/archiving) = Stable+DOI
* Use version control system
* Provide means to replicate the study
* Provides information about communication channels (e.g., email, issue tracker)
* Completeness more important than Size
* Workflow is broken down into small procedures
  + Raw data to Intermediate data to Final data to Table, Graphics, and numbers
  + Sequence of steps to facilitate the use by future researchers
  + Intermediate steps may help to understand and fix problems
* Uses appropriate licenses (e.g., that maximize impact, back-contributions and control)
* Other points:
  + Main file is easily identifiable, e.g., Readme file (Prana et al., 2019)
  + File formats are documented (e.g., meaning of rows and columns)
  + Restoration procedures are documented (e.g., for database dumps)
  + Reports used versions of artifacts
  + Compilation and execution processes documented
  + Training material should be made available and documented, especially for studies involving humans
  + Provide info on how to cite the paper(s)
* Provides a container, e.g., VM images, docker, for freezing dependencies and running environment quickly
* Relies on dependency management systems (e.g., maven, pip) to specify dependencies

Provides electronic notebooks, e.g., Jupyter Notebooks to document and typeset research code and results

* Promotes Open-science for outreaching to society
* Provides a beautiful repository with nice illustration
* Provides illustration of scientific data and code for citizens/general public
* Appealing to wider audience than our scientific peers

# The Journal of Open Science Software (JOSS)

Below we report a list of practices and properties extracted from (Katz et al., 2018).

* Includes a description focused on the software itself
* Uses appropriate license
* Includes documentation
* Built on GitHub (i.e., issue tracker, rapid interaction)
* Available in open access
* Available in a public repository
* Includes a list of authors
* Provides a description of the purpose of the software
* Provides references related to the artifact
* Repository is indexed
* Repository issues a DOI to the artifact
* Includes source code available
* Uses appropriate license
* Artifact versions used are indicated in the repository
* Provides a statement of need for the software
* Includes installation instructions
* Includes examples of usage (e.g., input data, commands)
* Includes a documentation of functionality
* Includes tests (e.g., unit test)
* Indicates community guidelines that have been followed
* Permanently available for retrieval

# The Journal of Open Research Software (JORS)

Below we report a list of practices and properties extracted from (JORS, 2021).

* Provides information to download the software
* Executable artifacts behave as described
* Provides a list of contributors
* Uses appropriate license
* Reports limitations of the project (e.g., performance, supported formats)
* Provides a description of the software's functionality
* Provides a description of where the software can be used
* Provides keywords to enable to search for the software
* Provides an indication of the context of the software development
* Provides an indication of the context of the software use
* Provides a description of how the software is designed
* Provides a description of constraints that may be placed on the software’s use
* Provides adequate explanation of how results can be trusted
* Provides suggestions for other potential applications
* Indicates ways of extending or modifying the software
* Provides guidelines for integration with other software
* Includes figures and diagrams
* Artifact is hosted in a suitable repository
* There is a persistent identifier (i.e., DOI)
* Hosted on a persistent, archived repository
* Includes sample input/output data
* Source code adequately documented (e.g., in-code citation (Alsudais, 2021))
* Provides description of how to build/deploy/install/run the software
* Provides means to identify if the software is operating as expected (e.g. smoke test)
* Describes the support mechanisms for the software
* Defines unfamiliar concepts
* Recognizes efforts in the area to frame the discussion
* Provides a summary of the experiences of the authors (e.g., authors' depth of understanding)
* Reports authors’ ORCID
* Provides all data as openly available
* Follows FAIR principles
* There is a Data Accessibility Statement (e.g., where to find, how can it be used)
* Gives credit to data obtained from other sources
* Repository suitable for the subject
* Repository has a sustainability model
* Deposited under explicitly indicated open license
* Data is available in an open, non-proprietary format
* Data is labeled (e.g., column headers, description in readme file)
* Data anonymization is indicated
* Funding sources are indicated
* Provides details about requirements for ethical research

# The Guidelines by Wilson et al.

Below we report a list of practices and properties extracted from (Wilson et al., 2017).

1. Data management:
   1. Includes the raw data
      1. Provides a record of the exact procedure used to obtain the raw data
      2. Provides a description of the raw data
   2. Raw data is backed up in more than one location (e.g., External HDs, university-owned data storage, incremental/special backup)
   3. Create the data you wish to see in the world.
      1. Data is provided in machine-readable format
      2. Textual content has reasonable human readability score (e.g., Kincaid, ARI)
      3. Uses open/non-proprietary file formats
      4. Variable names and data codes are self-explaining
      5. There are useful meta-data as part of filenames (e.g., for pattern matching)
      6. Indicates research community standards that have been followed
      7. Includes metadata for the dataset as a whole (e.g., authors, version)
      8. Includes metadata for the dataset’s content (e.g., headers, descriptions)
   4. Data follows an analysis-friendly format (e.g., column is variable, row is observation)
   5. There are scripts for all the steps used to process data
      1. There are scripts for every stage of data processing (e.g., ease redoing data preparation, collection)
      2. Scripts allow to produce intermediate data files (e.g., increasing levels of cleanliness and task specificity)
      3. Workflow is broken-down into small procedures (e.g., non-monolithic, explicit creation/retention of intermediate products)
   6. Uses unique identifiers to identify tables/records.
   7. Repository issues a DOI to the artifact (e.g., Figshare, Dryad, Zenodo)
2. Software
   1. Includes an explanatory comment at the start of every program (e.g., how to use, values for parameters, help menu)
   2. Software programs are decomposed into functions (e.g., ~60 LOC, <5-6 parameters)
   3. Source code has minimal/no duplication (e.g., avoid copy-paste, use data structures)
   4. Software artifact relies on well-maintained libraries that do what you need.
   5. Includes test cases to the reused libraries
   6. Functions and variables have meaningful names and follow language conventions (e.g., CamelCase, snake\_case)
   7. Explicitly indicates dependencies/requirements (e.g., requirements.txt, README.md)
   8. Program behavior set by if/else structures and parameters
   9. Provides examples/test data (e.g., build-and-smoke test) to ensure proper installation
3. Collaboration
   1. Provides an overview of the project
      1. Provides a summary of the purpose of the project
      2. Provides a project title
      3. Provides a description of the project
      4. Provides contact info
      5. Provides a running example
      6. Provides a description of how to setup local a workspace
      7. Provides a list of tasks to contribute (CONTRIBUTING file)
      8. Indicates guidelines/checklists used to manage the project
   2. Provides a shared to-do list (e.g., notes.txt, todo.txt) reporting issues/limitations
   3. Provides info about communication channels (e.g., mailing lists, chat, documentation, meeting notes)
   4. Make the license explicit. (e.g., LICENSE file)
   5. Provide info on how to cite the project (e.g., CITATION file)
4. Project organization
   1. Put each project in its own directory, which is named after the project
      1. Organizes project by the types of files to effective access and usage (i.e., consistency and predictability)
   2. Separates text documents associated with the project (e.g., doc directory)
      1. Manuscripts
      2. documentation for source code
      3. electronic lab notebooks
      4. subdirectories for different classes
   3. Separates raw data and metadata (e.g., in a data directory)
   4. Separates files generated during cleanup and analysis (e.g., in a results directory)
      1. Separates data and metadata in subdirectories according to time, method of collection or other metadata
      2. Separates cleaned data, statistical tables, publication-ready figures or tables
   5. Separates project source code (e.g., in the src directory)
      1. Distinguishes files by clear filenames or directory (e.g., core analysis, controller/driver scripts, analysis steps, particular parameters, data IO commands)
   6. Separates compiled programs (e.g., in the bin directory)
   7. Files are named to reflect their content or function
5. Tracking changes
   1. Changes are kept small and frequent (e.g., edits you could want to undo)
   2. Provides a list for tracking changes to the project
   3. Provides explanation for changes (e.g., CHANGELOG.txt, commit messages)
   4. Data is also stored remotely (e.g., remote version control)
   5. Provides a file called CHANGELOG.txt to the project's docs subfolder
   6. Copy the entire project whenever a significant change has been made.
   7. Provides means to reference/retrieve specific version aid to reproducibility
6. Supplementary materials
   1. Uses branches
   2. Uses build tools
   3. Implements unit testing
   4. Uses test coverage
   5. Uses continuous integration
   6. Provides means for profiling and performance tuning
   7. Uses semantics web concepts
   8. Provides documentation
   9. Uses bibliography management tools (e.g., Zotero, ORCID)

# The NASA Open Source Software Projects

Below we report a list of practices and properties extracted from (NASA, 2021).

* Provides project meta-data
* Provides tags for the code project
* Includes a description of the project
* Provides a list of individuals involved in its creation
* Provides a development timeline
* Provides documentation
* Includes a list of related topics
* Provides details about external dependencies (e.g., packages or sources)
* Provides information about terms and conditions of usage (e.g., license)
* Compliance with ICT accessibility standards (e.g., 508)
* Adopts software engineering practices

# The TACAS artifact evaluation guidelines

Below we report a list of practices and properties extracted from (TACAS, 2019).

* Substantiates the claims made in the paper
* Ideally makes the claims fully replicable
* Provides a tool in binary or source code format
* Provides documentation
* Provides input files (e.g., models analysed or programs verified)
* Provides configuration file or parameters used in the experiment
* Includes easy-to-use scripts to run experiment
* Includes easy-to-use scripts to draw tables/graphs
* Provides an abstract that summarizes the artifacts and its relation to the paper
* Provides a LICENSE file
* Provides README file that contains detailed instructions on how to replicate the study
* Provides means for verifying file integrity (e.g., SHA-256)
* Provides a virtual machine
* Provides a description on how-to use the artifacts
* Includes additional software or libraries required to run the software
* Includes instructions for artifact installation and setup
* Artifact should work without a network connection
* Provides information on how to replicate most, or ideally all, the experimental results
* Documentation assumes minimal expertise of users
* Provides a way to replicate a subset of the results with reasonably modest resources
* Describes resources/environment in which the artifact was successfully tested

# The CAV artifact evaluation guidelines

Below we report a list of practices and properties extracted from (CAV, 2019).

* Includes the final PDF
* Provides instructions on how to run the tool
* Provides virtual machine
* Provides means for verifying file integrity (e.g, SHA1)
* Provides a description of the OS and parameters of the image and host platform
* Provides an information on how to proceed after booting the VM image
* Provides instructions for locating the full documentation for evaluating the artifact
* Includes the accepted paper
* Provides a detailed README of how to run the tool
* Includes benchmarks with tool and/or proof scripts
* Artifact shall be easy to evaluate and yields expected results
* Describes how to reproduce most of the experimental results
* Provides a simple process to run the tool
* Documentation assumes minimal expertise of users
* Artifact should work without network connection
* Provides a way of replicating the requite with reasonably modest resources in a reasonable amount of time to complete
* Includes a simpler benchmark set
* Describes where to find the most relevant and interesting parts of the source tree code
* Includes log files produced by their tools
* Includes relevant log files

# References

ACM, 2020. Artifact Review and Badging - Current [WWW Document]. URL https://www.acm.org/publications/policies/artifact-review-and-badging-current (accessed 3.18.21).

Alsudais, A., 2021. In-code citation practices in open research software libraries. Journal of Informetrics 15, 101139. https://doi.org/10.1016/j.joi.2021.101139

CAV, 2019. Artifacts | CAV 2019. URL http://i-cav.org/2019/artifacts/ (accessed 3.18.21).

EMSE, 2021a. EMSE Open Science Initiative [WWW Document]. URL https://github.com/emsejournal/openscience/blob/master/README.md (accessed 3.18.21).

EMSE, 2021b. EMSE Open science - Evaluation Criteria [WWW Document]. GitHub. URL https://github.com/emsejournal/openscience/blob/master/review-criteria.md (accessed 3.18.21).

JORS, 2021. The Journal of Open Research Software - Editorial Policies [WWW Document]. URL http://openresearchsoftware.metajnl.com/about/editorialpolicies/ (accessed 3.18.21).

Katz, D.S., Niemeyer, K.E., Smith, A.M., 2018. Publish your software: Introducing the Journal of Open Source Software (JOSS). Computing in Science Engineering 20, 84–88. https://doi.org/10.1109/MCSE.2018.03221930

Méndez Fernández, D., Monperrus, M., Feldt, R., Zimmermann, T., 2019. The open science initiative of the Empirical Software Engineering journal. Empir Software Eng 24, 1057–1060. https://doi.org/10.1007/s10664-019-09712-x

Monperrus, M., 2019. How to make a good open-science repository? [WWW Document]. Research Data at Springer Nature. URL https://researchdata.springernature.com/posts/57389-how-to-make-a-good-open-science-repository (accessed 3.19.21).

NASA, 2021. NASA Open Source Software [WWW Document]. URL https://code.nasa.gov/ (accessed 3.18.21).

Prana, G.A.A., Treude, C., Thung, F., Atapattu, T., Lo, D., 2019. Categorizing the Content of GitHub README Files. Empir Software Eng 24, 1296–1327. https://doi.org/10.1007/s10664-018-9660-3

TACAS, 2019. TACAS 2019 - ETAPS 2019 [WWW Document]. URL https://conf.researchr.org/track/etaps-2019/tacas-2019-papers#Artifact-Evaluation (accessed 3.18.21).

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., Teal, T.K., 2017. Good enough practices in scientific computing. PLOS Computational Biology 13, e1005510. https://doi.org/10.1371/journal.pcbi.1005510