Automated Software Metadata Conversion and Publication Based on CodeMeta

Marie Houillon[1], Jochen Klar[2], Tomas Stary[1], Axel Loewe[1]

[1]Karlsruhe Institute of Technology (KIT);
[2]Independent Software Developer

Reproducible research requires publication of software together with appropriate metadata. Different metadata standards exist for different steps in the research software publication process: the Citation File Format (CFF) became very popular to provide information on how users are supposed to cite the software, DataCite is one of the established standards for research data archiving and CodeMeta is an extension of schema.org specifically tailored to research software. If research software developers must maintain a whole set of metadata files in different formats with largely overlapping content, it poses a risk both to data consistency and to adoption of good software publication practices. Therefore, we developed pipelines that put developers in a position to only maintain a CodeMeta file, from which CFF and DataCite files are automatically generated. These pipelines can easily be integrated in continuous integration and deployment environments. They also provide tools for software publication via tagged releases, creation of BagIt and BagPack files and publication on the research data repository RADAR.

1 Introduction

Research software development is a fundamental aspect in research (Anzt et al. 2021), and it is now acknowledged that the FAIR principles (Findable, Accessible, Interoperable, Reproducible; Wilkinson et al. 2016), historically established for research data, should also be applied to research software (Chue Hong et al. 2021). In particular, reproducible research requires that software and its associated metadata can be found easily by both machines and humans, and that they are retrievable via standardised protocols. In this context, several metadata standards are widely used across the scientific community:

- the Citation File Format (CFF; Druskat et al. 2021)[1] aims to indicate to users how to cite a software package

Publiziert in: Vincent Heuveline, Nina Bisheh und Philipp Kling (Hg.): E-Science-Tage 2023. Empower Your Research – Preserve Your Data. Heidelberg: heiBOOKS, 2023. DOI: `https://doi.org/10.11588/heibooks.1288.c18081` (CC BY 4.0)

[1] CFF: `https://citation-file-format.github.io`.

- DataCite[2] (DataCite Metadata Working Group 2021) is a standard Metadata schema for archiving digital assets
- CodeMeta[3] (Jones et al. 2017) is an extension of `schema.org` created to standardize the exchange of software metadata across repositories and organizations

All of these standards serve specific purposes and several of them are required to cover the whole software lifecycle. However, research software developers should ideally not be burdened with maintaining multiple metadata files in different formats and largely overlapping content. This poses a risk to data consistency and to adoption of good software publication practices.

Therefore, we have developed a framework, named *openCARP-CI*, which allows developers to easily create and maintain the metadata associated to research software, by only maintaining a CodeMeta file from which CFF and DataCite files are automatically generated. The framework also allows publishing software according to the FAIR principles: releases with persistent identifiers can be created, archived and published on the open research data repository RADAR.

2 Description of Components

2.1 The openCARP-CI environment

The openCARP-CI package (Houillon et al. 2023) is part of the openCARP Collaborative Development Environment (Bach et al. 2022), an advanced technical infrastructure for collaborative research software projects based on GitLab[4]. It is composed of a set of Python scripts around the publication and long-term preservation of software repositories (see Figure 1). These tasks can be performed automatically when being integrated in GitLab Continuous Integration and Deployment (CI/CD) pipelines.

The openCARP-CI was created for the openCARP simulation software (Plank et al. 2021) but has its own separated repository and can be adopted for any project including research software hosted on GitLab. It complements efforts by other teams (such as the HERMES Project[5]) that aim to simplify publication workflow of research software together with rich metadata.

In the next section, we describe the different pipelines related to metadata management and software publication available in openCARP-CI.

[2] DataCite: `https://schema.datacite.org`.
[3] CodeMeta: `https://codemeta.github.io`.
[4] GitLab: `https://about.gitlab.com`.
[5] HERMES: `https://project.software-metadata.pub`.

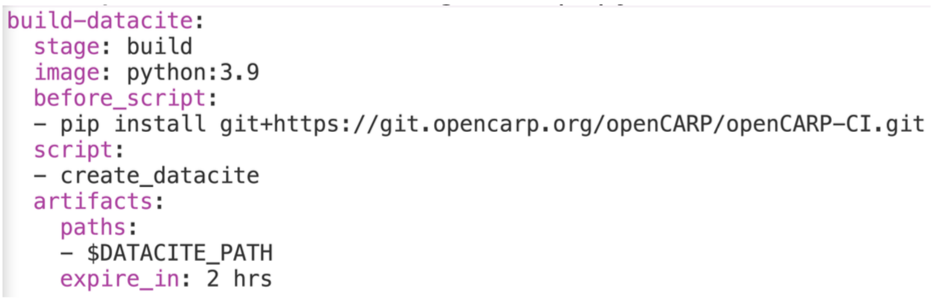
Table 1: Components of openCARP-CI.

Script	Functionality
`create_cff`	generates Citation File Format (CFF) metadata file
`prepare_release`	updates *version* and *dateModified* in metadata
`create_release`	creates release in GitLab
`create_datacite`	generates DataCite metadata file
`create_bag`	creates BagIt package
`create_bagpack`	adds DataCite XML to BagIt
`prepare_radar`	reserves DOI on RADAR
`create_radar`	creates archive and uploads it to RADAR
`run_markdown_pipeline`	updates Grav CMS website
`run_bibtex_pipeline`	treats BibTex file for publications on website
`run_docstring_pipeline`	extracts docstrings from Python scripts

2.2 Automated metadata conversion

In order to ensure the coherence of metadata across different metadata file formats and to remove the burden of copying and maintaining redundant metadata information in several files, openCARP-CI offers scripts that convert metadata expressed in the CodeMeta standard to other metadata formats. As a consequence, developers only need to maintain `codemeta.json` as the unique metadata file for their software.

To generate the initial codemeta.json file, the CodeMeta Generator[6] can be used. Then, the script `create_cff` generates a Citation File Format (CFF) metadata file from the CodeMeta file (Druskat et al. 2021). The script `create_datacite` generates a DataCite XML file from the CodeMeta file.



```
build-datacite:
  stage: build
  image: python:3.9
  before_script:
  - pip install git+https://git.opencarp.org/openCARP/openCARP-CI.git
  script:
  - create_datacite
  artifacts:
    paths:
    - $DATACITE_PATH
    expire_in: 2 hrs
```

Figure 1: Example of a Gitlab CI job for automated creation of the DataCite metadata file.

[6] CodeMeta-Generator: `https://codemeta.github.io/codemeta-generator`.

2.3 Creation of releases

A software release associated with a version number can be created on GitLab using the scripts `prepare_release` and `create_release`. The script `prepare_release` updates the CodeMeta file with a given version number and date. When using the script as part of a CI pipeline, this information is taken from the *tag* of the release and the current date. The script `create_release` actually creates the software release on GitLab using its API.

2.4 Creation of archives

openCARP-CI allows creating software packages destined to persistent long-term storage in research data repositories. These archives are created using the BagIt File Packaging Format[7], which is designed for reliable storage and transfer of arbitrary digital content.

The script `create_bag` creates a BagIt package containing the given assets, using the Python package `bagit-python`[8]. The script `create_bagpack` adds a DataCite XML file to the BagIt package as recommended by the RDA Research Data Repository Interoperability WG (RDA Research Data Repository Interoperability WG 2018).

2.5 Software publication

`prepare_radar` and `create_radar`, can be used to publish the software in the research data repository service RADAR[9]. In the RADAR repositories, datasets are assigned a persistent DOI (Digital Object Identifier) and published in accordance with the FAIR principles.

The script `prepare_radar` assigns a DOI and a RADAR ID to the dataset and adds them to its metadata (`codemeta.json`). The script `create_radar` creates the release in the RADAR service. This is done in a two step process, where first a *dataset* is created in RADAR, which contains the metadata. Then, in a second step, the different assets of the release (e.g. the source code and different compiled binaries) are uploaded.

2.6 Integration with the project website

An additional feature of openCARP-CI is publication of relevant information on a web page managed with the Grav content management system (CMS)[10].
The scripts `run_markdown_pipeline`, `run_bibtex_pipeline` and `run_docstring_pipeline` can be used for this purpose if desired.

[7] BagIt description: `https://www.rfc-editor.org/rfc/rfc8493`.

[8] bagit-python repository: `https://github.com/LibraryOfCongress/bagit-python`.

[9] RADAR: `https://radar.products.fiz-karlsruhe.de/en`.

[10] Grav CMS: `https://getgrav.org`.

3 Pipeline setup in a software repository

3.1 Prerequisites

The pipelines provided in openCARP-CI can be set up directly in any software project which fulfills the following conditions:

- The project's repository is under version control using Git and hosted in a GitLab instance
- A Docker runner is configured for the project's GitLab CI pipelines
- For optional publiction on RADAR, credentials have to be provided

3.2 Integration in GitLab CI pipelines

The CI scripts can be included in any GitLab project using the following process.
For projects hosted on GitHub, adaptations are required[11].

- In the project repository, go to *Settings → Access Tokens*, and create a token with the role *Maintainer* and scopes *api* and *write_repository*. Copy the token value.
- Go to *Settings → CI/CD → Variables* and choose *Add Variable*. Create a masked variable named `PUSH_TOKEN` and as a value, paste the copied token.
- Create a variable with key `PRIVATE_TOKEN` and as a value enter `$PUSH_TOKEN`.
- Copy the GitLab CI configuration files (`.gitlab-ci.yml` and `.gitlab/`) from the openCARP-CI repository to your software repository and adapt them to your needs. You can deactivate the release on RADAR by setting `ENABLE_RADAR` to `false` in `.gitlab-ci.yml`.
- Create a commit with the tag `pre-vX.Y`. The CI jobs will update metadata and create a release commit with the tag `vX.Y`.

4 Conclusions

The package openCARP-CI provides tools for automatic metadata conversion and software publication according to the FAIR principles, which can be automated in CI/CD pipelines on the GitLab development platform. After the initial setup, the user maintains a single metadata file in CodeMeta format. Other metadata formats are automatically generated from this file. The releases and supporting files are archived automatically for every new version of the software.

[11] GitLab CI/CD to GitHub Actions: `https://docs.github.com/en/actions/migrating-to-github-actions/manual-migrations/migrating-from-gitlab-cicd-to-github-actions`.

We believe that the automated metadata conversion based on CodeMeta can be a useful tool for many research software developers and can facilitate the adoption of good software publication practices by reducing the effort for developers.

Acknowledgements

We gratefully acknowledge support by Deutsche Forschungsgemeinschaft (DFG, projects LO2093/1-1 and LO2093/9-1) and Karlsruhe Institute of Technology (KIT). This project has received funding from the European High-Performance Computing Joint Undertaking EuroHPC (JU) under grant agreement No 955495. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Italy, Germany, Austria, Norway, Switzerland.

References

Anzt, H, F Bach, S Druskat, F Löffler, A Loewe, BY Renard, G Seemann, et al. 2021. "An environment for sustainable research software in Germany and beyond: current state, open challenges, and call for action". *F1000Research* 9 (295). DOI: `https://doi.org/10.12688/f1000research.23224.2`.

Bach, Felix, Jochen Klar, Axel Loewe, Jorge Sánchez, Gunnar Seemann, Yung-Lin Huang, and Robert Ulrich. 2022. "The openCARP CDE: Concept for and implementation of a sustainable collaborative development environment for research software". *Bausteine Forschungsdatenmanagement,* number 1: 64–84. DOI: `https://doi.org/10.17192/bfdm.2022.1.8368`.

Chue Hong, Neil P., Daniel S. Katz, Michelle Barker, Anna-Lena Lamprecht, Carlos Martinez, Fotis E. Psomopoulos, Jen Harrow, et al. 2021. "FAIR Principles for Research Software (FAIR4RS Principles)". DOI: `https://doi.org/10.15497/RDA00068`.

DataCite Metadata Working Group. 2021. "DataCite Metadata Schema Documentation for the Publication and Citation of Research Data and Other Research Outputs v4.4". DOI: `https://doi.org/10.14454/3W3Z-SA82`. `https://schema.datacite.org/meta/kernel-4.4/`.

Druskat, Stephan, Jurriaan H. Spaaks, Neil Chue Hong, Robert Haines, James Baker, Spencer Bliven, Egon Willighagen, David Pérez-Suárez, and Olexandr Konovalov. 2021. *Citation File Format.* Version 1.2.0. DOI: `https://doi.org/10.5281/zenodo.5171937`.

Houillon, Marie, Jochen Klar, Axel Loewe, Tomas Stary, and openCARP consortium. 2023. *openCARP-CI.* DOI: `https://doi.org/10.35097/974`.

Jones, Matthew B., Carl Boettjiger, Abby Cabunoc Mayes, Arfon Smith, Peter Slaughter, Kyle Niemeyer, Yolanda Gil Gil, et al. 2017. "CodeMeta: an exchange schema for software metadata. Version 2.0." Edited by KNB Data Repository. DOI: https://doi.org/10.5063/schema/codemeta-2.0.

Plank, Gernot, Axel Loewe, Aurel Neic, Christoph Augustin, Yung-Lin Huang, Matthias A.F. Gsell, Elias Karabelas, et al. 2021. "The openCARP simulation environment for cardiac electrophysiology". *Computer Methods and Programs in Biomedicine* 208:106223. DOI: https://doi.org/10.1016/j.cmpb.2021.106223.

RDA Research Data Repository Interoperability WG. 2018. *Research Data Repository Interoperability WG Final Recommendations.* DOI: https://doi.org/10.15497/RDA00025.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersbergand, Gabrielle Appleton, Myles Axtonand, Arie Baakand, Niklas Blombergand, et al. 2016. "The FAIR Guiding Principles for scientific data management and stewardship". *Scientific data* 3 (1): 1–9. DOI: https://doi.org/10.1038/sdata.2016.18.