Automated Software Metadata Conversion and Publication Based on CodeMeta

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

[1]Karlsruhe Institute of Technology (KIT), Germany
[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 [1], and it is now acknowledged that the FAIR principles (Findable, Accessible, Interoperable, Reproducible) [2], historically established for research data, should also be applied to research software [3]. 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) aims to indicate to users how to cite a software package
- DataCite is a standard Metadata scheme for archiving digital assets
- CodeMeta [4] 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 a whole set of metadata files in different formats and largely overlapping content. This poses a risk both to data consistency and to adoption of good software publication practices in the first place.

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 [5] is part of the openCARP Collaborative Development Environment [6], an advanced technical infrastructure for collaborative research software projects based on GitLab[1]. It is composed of a set a Python scripts around the publication and long-term preservation of software repositories (see Fig. 1). These tasks can be performed automatically when being integrated in GitLab Continuous Integration and Deployment (CI/CD) pipelines. openCARP-CI was created for the openCARP simulation software [7] but has its own separated repository and can be adopted for any arbitrary project hosted on GitLab.

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

Table 1: Components of openCARP-CI

2.2 Automated metadata conversion

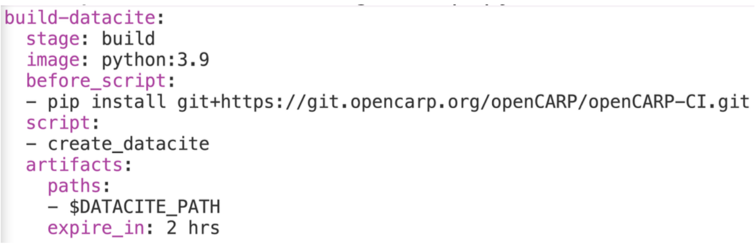
In order to ensure the coherence of metadata across different metadata file formats and to remove the burden of copying and maintaining redundant metadata information in several files,

[1] GitLab: https://about.gitlab.com

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.

The script `create_cff` generates a Citation File Format (CFF) metadata file from the CodeMeta file [8].

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

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`. `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[2], 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`[3]. The script `create_bagpack` adds a DataCite XML file to the BagIt package, as recommended by the RDA Research Data Repository Interoperability WG [9].

2.5 Software publication

With the scripts `prepare_radar` and `create_radar`, developers can publish their software in the research data repository service RADAR[4]. 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] BagIt description: https://www.rfc-editor.org/rfc/rfc8493

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

[4] https://radar.products.fiz-karlsruhe.de/en

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
- Optionally, for using the RADAR pipeline, developers must have credentials for publishing in a RADAR instance

3.2 Integration in Gitlab CI pipelines

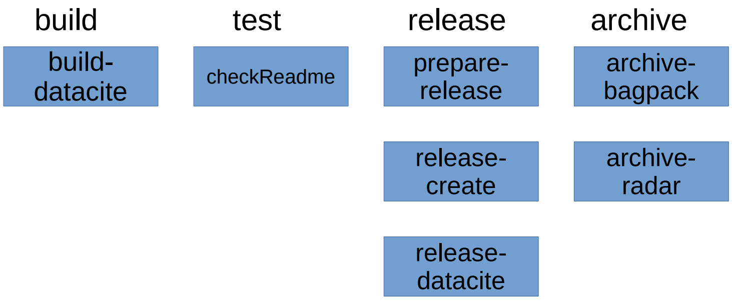


Figure 2: Example of openCARP-CI workflow (each box represents a job in the Gitlab CI pipeline)

Fig. 2 shows an example of workflow integrating the metadata and publication pipelines. This workflow can be included in another GitLab project using this process:

- 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. These files should be adapted to the needs of your project. In particular, 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.

We believe 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

[1] H. Anzt, F. Bach, S. Druskat, *et al.*, "An environment for sustainable research software in Germany and beyond: Current state, open challenges, and call for action," *F1000Research*, vol. 9, no. 295, 2021. DOI: 10.12688/f1000research.23224.2.

[2] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, no. 1, pp. 1–9, 2016.

[3] N. P. Chue Hong, D. S. Katz, M. Barker, *et al.*, "FAIR principles for research software (FAIR4RS principles)," 2021. DOI: 10.15497/RDA00068.

[4] M. B. Jones, C. Boettjiger, A. C. Mayes, *et al.*, "Codemeta: An exchange schema for software metadata. version 2.0.," K. D. Repository, Ed., 2017. DOI: 10.5063/schema/codemeta-2.0.

[5] M. Houillon, J. Klar, A. Loewe, T. Stary, and openCARP consortium, *openCARP-CI*, 2023. DOI: 10.35097/974.

[6] F. Bach, J. Klar, A. Loewe, *et al.*, "The openCARP CDE: Concept for and implementation of a sustainable collaborativedevelopment environment for research software," *Bausteine Forschungsdatenmanagement*, no. 1, pp. 64–84, 2022. DOI: 10.17192/bfdm.2022.1.8368.

[7] G. Plank, A. Loewe, A. Neic, *et al.*, "The openCARP simulation environment for cardiac electrophysiology," *Computer Methods and Programs in Biomedicine*, vol. 208, p. 106 223, 2021. DOI: 10.1016/j.cmpb.2021.106223.

[8] S. Druskat, J. H. Spaaks, N. Chue Hong, *et al.*, *Citation File Format*, version 1.2.0, 2021. DOI: 10.5281/zenodo.5171937.

[9] RDA Research Data Repository Interoperability WG, *Research Data Repository Interoperability WG Final Recommendations*, 2018. DOI: 10.15497/RDA00025.