Bausteine Forschungsdatenmanagement
Empfehlungen und Erfahrungsberichte für die Praxis von Forschungsdatenmanagerinnen und -managern

The openCARP CDE[i]

Concept for and implementation of a sustainable collaborative development environment for research software

Felix Bach[ii]   Jochen Klar[iii]   Axel Loewe[iv]   Jorge Sánchez[v]   Gunnar Seemann[vi]

Yung-Lin Huang[vii]   Robert Ulrich[viii]

2022
Zitiervorschlag

[i]Correspondence: info@openCARP.org. All authors contributed equally.

[ii]FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany. ORCID: 0000-0002-5035-7978

[iii]Independent researcher, Germany. ORCID: 0000-0002-5883-4273

[iv]Institute of Biomedical Engineering, Karlsruhe Institute of Technology (KIT), Germany. ORCID: 0000-0002-2487-4744

[v]Institute of Biomedical Engineering, Karlsruhe Institute of Technology (KIT), Germany. ORCID: 0000-0002-0824-2691

[vi]Institute for Experimental Cardiovascular Medicine, University Heart Centre Freiburg Bad Krozingen, Germany. ORCID: 0000-0001-7111-7992

[vii]Institute for Experimental Cardiovascular Medicine, University Heart Centre Freiburg Bad Krozingen, Germany. ORCID: 0000-0003-4307-3907

[viii]KIT Library, Karlsruhe Institute of Technology (KIT), Germany. ORCID: 0000-0001-9063-2703

Abstract

This work describes the setup of an advanced technical infrastructure for collaborative software development in large, distributed projects based on GitLab. We present its customization and extension, additional features and processes like code review, continuous automated testing, DevOps practices, and sustainable life-cycle management including long-term preservation and citable publishing of software releases along with relevant metadata. The collaborative development environment (CDE) is currently used for developing the open cardiac simulation software openCARP and an evaluation showcases its capability and utility for collaboration and coordination of sizeable heterogeneous teams. As such, it could be a suitable and sustainable infrastructure solution for a wide range of research software projects.

1. Introduction

Successful research software development is often driven by a vibrant user community. Advanced infrastructure can help to make the software maintainable and sustainable.[1] Such an environment was built to foster test-driven development and convenient exchange between users and developers in the DFG-funded SuLMaSS project (Sustainable Life-cycle Management for Scientific Software) and road-tested by the openCARP cardiac electrophysiology simulator.[2] In this work, we describe an infrastructure, which includes a GitLab-based development platform that enables feature requests, bug tracking and release plans, quality assurance practices such as code review by core developers, continuous automated testing, and continuous automated deployment. It also includes tools for sustainable life-cycle management to facilitate reuse and provides means for automatic citable publishing of software releases along with additional data and descriptive metadata in research data repositories, with persistent identifiers (PID) and archiving for long-term availability. Content for the project website is automatically generated and synchronized based on the code repository. All processes are highly automated to reduce manual maintenance effort. The infrastructure could be reused as a development environment for similar research software projects.

---

[1]Hartwig Anzt et al., "An environment for sustainable research software in Germany and beyond: current state, open challenges, and call for action," *F1000Research* 9 (January 2021): 295, doi:10.12688/f1000research.23224.2.

[2]Gernot Plank et al., "The openCARP Simulation Environment for Cardiac Electrophysiology," 2021, doi:10.1101/2021.03.01.433036; Jorge Sánchez et al., "openCARP: An Open Sustainable Framework for In-Silico Cardiac Electrophysiology Research," in *Computing in Cardiology Conference (CinC)* 47 (January 2020), doi:10.22489/CinC.2020.111.

2. Description of Components

2.1. Software repository and continuous integration

The core of the openCARP CDE is the software repository[3], which is based on the open source tool GitLab[4]. GitLab is a well-established software, which is available both as a hosted service, usable on different free and billed plans, and as self-hosted solution. GitLab offers rich features regarding all stages of the software lifecyle.

GitLab is unique with its feature set as a self-hosted open-source solution for a Git-based software development platform. For our purposes, the features of the free Community Edition are sufficient. The software is installed on a virtual machine (VM) running CentOS7 using the GitLab all-in-one package. The specifications of the machine are: 2 CPUs, 8 GB of RAM and 200 GB of disk space (mainly used to store the docker images). As an alternative to the self-hosted instance of GitLab, it would be possible to use the as-a-service variant `GitLab.com` or competing services like `GitHub` or `Bitbucket`. While the main functions are similar, a hosted service does not give the same freedom of integration with other services (e.g. regarding authentication, see Section 2.4) and, more important, it implies that all data are hosted on external servers. It also requires a different funding approach, since those services usually bill by user and month.

Integrated within GitLab is a powerful continuous integration (CI) framework, which allows performing automatic tests, generating documentation and other tasks, whenever a new commit is pushed to the repository. Through this functionality we are able to connect the different components of the openCARP CDE and synchronize information between the code repositories, the project webpage and the long-term archive.

Furthermore, GitLab ships with its own docker registry[5]. Docker[6] uses container-based virtualization to encapsulate the whole system environment of an application (except the kernel of the operating system). Using Docker, we can create images for openCARP, which include all prerequisites and can therefore be installed with much less friction than would be the case for distribution of the source code alone.

GitLab offers sophisticated possibilities to create and orchestrate its CI jobs. An overview of our pipelines is given in Figure 2. Jobs can be grouped into different stages, which are processed sequentially, and can have dependencies. The main tasks of the CI for openCARP are:

- Build `deb` and `rpm` packages for Debian, CentOS and Ubuntu based Linux distributions, AppImages for Linux-based systems, as well as `pkg` packages for macOS.

---

[3]openCARP GitLab: git.openCARP.org, accessed July 28, 2021.
[4]GitLab: https://about.gitlab.com, accessed July 28, 2021.
[5]In our case docker.openCARP.org, accessed July 28, 2021.
[6]Docker: https://www.docker.com, accessed July 28, 2021.

These binary packages allow for an automatic installation of openCARP and all its system prerequisites.

- Build a Docker image from the openCARP source and upload it to the openCARP docker registry. Docker allows for a frictionless deployment of openCARP on any System which can run Docker (Linux, macOS, Windows).

- Build the Doxygen[7] documentation of openCARP and copy it to a public directory on openCARP.org[8]. Doxygen automatically creates a browsable documentation from comments (so-called docstrings) in the source code.

- Build the user manual as pdf and copy it to a public directory on openCARP.org.

- Build a browsable interface for the different parameters of openCARP and copy it to a public directory on openCARP.org[9]. This and the artifacts mentioned in the two points above are available per Git branch and tags and thus allow users to go back to the documentation of any specific version.

- Build a browseable report interface for the available regression tests and copy the report to the public directory on openCARP.org[10].

- Generate a DataCite record from structured metadata files in the code repository (see Section 2.5) and copy it to a public directory on openCARP.org.

- Create releases for certain versions of the code in GitLab. On a technical level, releases are tags in Git, which have certain assets assigned to them. In our case, these are the different binary distributions of openCARP. GitLab creates a release page where those assets are accessible, as well as a tarball of the repository (at the release commit).

- Create a dataset in the long-term archive RADAR4KIT[11] and upload the release assets (see Section 2.5).

- Create a bagpack archive with the release assets and the DataCite metadata file (see Section 2.5).

- Synchronize the content of the openCARP.org GRAV content management system with structured metadata files from the code repository. Specific pages in the CMS are tagged with `pipeline` and `source` metadata entries. For openCARP, if a page has the pipeline `openCARP`, the CI inserts the content of the file given as

---

[7]Doxygen: http://www.doxygen.nl, accessed July 28, 2021.

[8]openCARP documentation: https://openCARP.org/documentation/opencarp-documentation, accessed July 20, 2021.

[9]openCARP parameters: https://openCARP.org/documentation/opencarp-parameters, accessed July 28, 2021.

[10]openCARP parameters: https://openCARP.org/documentation/test-reports#webpage_testing_report, accessed August 04, 2021.

[11]RADAR4KIT: https://radar.kit.edu, accessed July 28, 2021.

source into the metadata of the page. Its content can then be used as structured data by the CMS, e.g. to use the contributors list from the repository to display on the public web page[12] (see Section 2.2 for more information on the GRAV CMS).
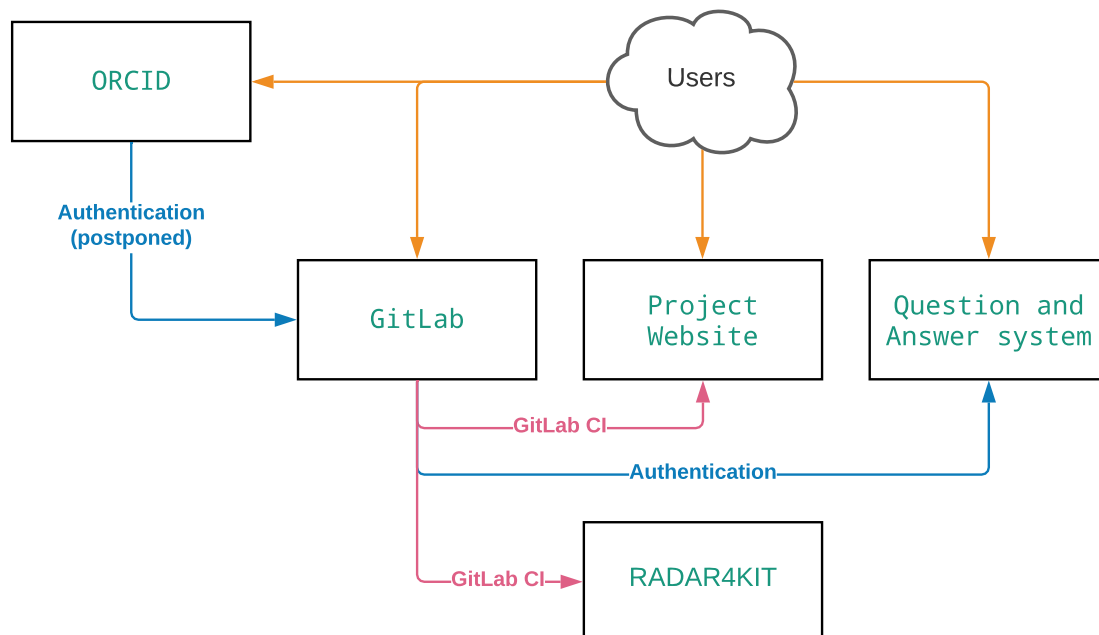
Figure 1: Schema of the main components of the openCARP CDE. The users interact with the GitLab software repository, the project web page and the question and answer (Q&A) system. Authentication for the software repository is not yet possible via ORCID, but can be obtained via GitHub/GitLab.com, while the authorization for the Q&A system is provided by the software repository. RADAR4KIT is used to automatically publish and archive software releases from GitLab

Similar, but less extensive pipelines exist for other repositories, like the `carputils`[13] framework. Since the compilation process of openCARP causes some computational effort, CI jobs are not executed on the same VM that is used for GitLab but on a different, dedicated VM with 4 CPUs and less disk space. The different assets, which are produced by the CI are copied to a web space under openCARP.org from which they can be downloaded or integrated in the public project web page (see Section 2.2). Besides, the binary packages are uploaded to GitLab package registry, which preserves records between assets and repository. The project-specific pipelines are available in the openCARP Git repository. The generic pipelines for creating releases, DataCite records, BagIt/BagPack archive packages, RADAR releases and synchronizing code

---

[12]Contributors to openCARP: https://openCARP.org/community/contributors, accessed July 28, 2021.
[13]carputils: https://git.openCARP.org/openCARP/carputils, accessed July 28, 2021.

repository content with the web page are available in the openCARP-CI repository[14] together with documentation on how to use them.

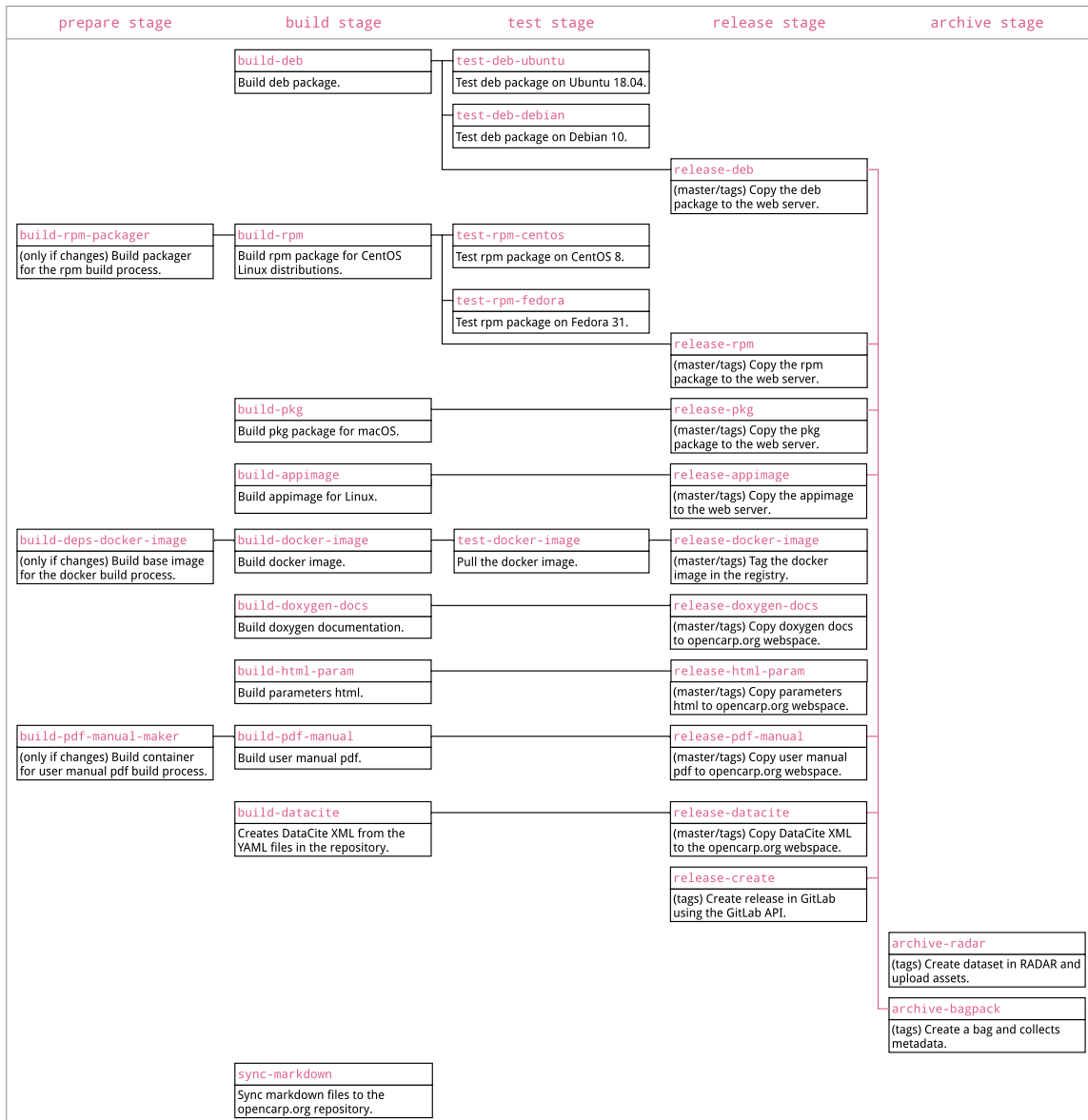| prepare stage	build stage	test stage	release stage	archive stage
	build-deb Build deb package.	**test-deb-ubuntu** Test deb package on Ubuntu 18.04. **test-deb-debian** Test deb package on Debian 10.	**release-deb** (master/tags) Copy the deb package to the web server.	
build-rpm-packager (only if changes) Build packager for the rpm build process.	**build-rpm** Build rpm package for CentOS Linux distributions.	**test-rpm-centos** Test rpm package on CentOS 8. **test-rpm-fedora** Test rpm package on Fedora 31.	**release-rpm** (master/tags) Copy the rpm package to the web server.	
	build-pkg Build pkg package for macOS.		**release-pkg** (master/tags) Copy the pkg package to the web server.	
	build-appimage Build appimage for Linux.		**release-appimage** (master/tags) Copy the appimage to the web server.	
build-deps-docker-image (only if changes) Build base image for the docker build process.	**build-docker-image** Build docker image.	**test-docker-image** Pull the docker image.	**release-docker-image** (master/tags) Tag the docker image in the registry.	
	build-doxygen-docs Build doxygen documentation.		**release-doxygen-docs** (master/tags) Copy doxygen docs to opencarp.org webspace.	
	build-html-param Build parameters html.		**release-html-param** (master/tags) Copy parameters html to opencarp.org webspace.	
build-pdf-manual-maker (only if changes) Build container for user manual pdf build process.	**build-pdf-manual** Build user manual pdf.		**release-pdf-manual** (master/tags) Copy user manual pdf to opencarp.org webspace.	
	build-datacite Creates DataCite XML from the YAML files in the repository.		**release-datacite** (master/tags) Copy DataCite XML to the opencarp.org webspace.	
			release-create (tags) Create release in GitLab using the GitLab API.	
				archive-radar (tags) Create dataset in RADAR and upload assets. **archive-bagpack** (tags) Create a bag and collects metadata.
	sync-markdown Sync markdown files to the opencarp.org repository.			

Figure 2: Schema of the CI jobs for the openCARP repository. Jobs are separated into different stages (prepare, build, test, release and archive), which are processed sequentially. Dependencies between jobs are indicated by black (same task) or pink (different tasks) arrows. Some jobs are only called for the master branch or only for releases (tagged commits in Git)

---

[14]openCARP-CI: https://git.openCARP.org/openCARP/openCARP-CI, accessed July 28, 2021.

2.2. Project website

For the general public, the web page `openCARP.org` acts as the central contact point and hub for information about the software and as starting point for new users and possible collaborators. The page informs about the structure of the consortium, its governance and code of conduct and features news, announcements and recent tweets[15]. Users can subscribe to an email newsletter to keep being informed about openCARP. Code examples and video tutorials make it easy to get started and perform first simulations. The documentation of openCARP and carputils as well as the user manual are embedded into the page.

The page, which is based on GRAV[16], works two-fold: If the CMS content is changed via the GRAV admin interface, the repository in GitLab is automatically updated and if changes are pushed to the master branch of the GitLab repository, the web page is updated accordingly.

In addition to this synchronization between the web page and its Git repository, we use the GitLab CI (described in Section 2.1) to integrate additional structured Markdown files from other software repositories. For example, the list of contributors and publications, and the code of conduct, which are both part of the main openCARP software repository, are synchronized to the web page in this way. The publication list is automatically built from a BibTeX file. In a similar way, the example experiments reside in their own dedicated repository web pages and are created directly from the Python source code files of the experiments via a pandoc[17] pipeline.

Additionally, usage statistics for the web page (number of requests, visiting countries, visited page, transfer volume) can be created from the local log files without the need for external services or cookies.

2.3. Question and answer system

In software engineering, question and answer (Q&A) platforms have become a valuable tool for developers to discuss, learn, and interact. In recent years, they mostly replaced mailing lists and classical internet forums. The most important Q&A site nowadays is certainly Stack Overflow[18]. Q&A systems usually employ the following features:

1. Users are invited to ask a question using a convenient web interface.

2. Other users can answer these question or comment on the question (e.g., asking for clarification).

---

[15]@openCARP Twitter Account: https://twitter.com/openCARP, accessed July 28, 2021.
[16]GRAV: https://getgrav.org, accessed July 28, 2021.
[17]pandoc: https://pandoc.org, accessed July 28, 2021.
[18]https://stackoverflow.com/questions, accessed July 28, 2021.

3. Specific answers can be marked as accepted if they provide sufficient information to solve the initial problem.

4. All other users are invited to up- or down-vote an answer's quality, thus collectively ordering the thread. This self-regulation nature of the system allows for an effortless access to the best answers in the future.

5. Questions can be tagged with specific topics to keep the system clearly arranged even after a large number of user interactions and to notify the respective maintainers about new questions.

As with the GitLab repository, we decided to have our own, on-premise question and answer solution. Ideally, the platform will grow to a high-quality knowledge library around the openCARP software. Stack Overflow is a centrally hosted platform and sets high bars to create new topic-specific instances. Therefore, it was not suitable for this scenario. Several other solutions exist, however:

- Question2Answer is an open source Q&A platform using a PHP/MySQL software stack. According to its web page[19], it is used on over 24,000 sites. Question2Answer is licensed under the GPLv2+ license and can therefore be adopted without problems. Its community on GitHub appears quite active, although the last commit to the code in July 2021 was from November 2020.

- Askbot[20] is another open source Q&A solution and is used, e.g., by the fedora Linux distribution. It uses Python and the Django Framework as underlying technology and is licensed under the GPLv3, which allows for an easy adaptation. Its GitHub community seems more active than that of Question2Answer, with commits in recent days. It is based on the no-longer-maintained OQSA system.

- Shapado was a Ruby on Rails based Q&A system, but it is clearly unmaintained since 2012. The code is still available on GitHub[21].

- The enterprise software company Atlassian also offers a Q&A solution as add-on to its collaboration software Confluence[22], which is closed source software and priced depending on the number of users.

After weighing up the functions, requirements and maintenance status of the different applications, we decided to use Question2Answer for the openCARP CDE. With PHP, it uses the same underlying technology as the GRAV CMS and MySQL is a well-established and easy-to-maintain database system. While we are not able to integrate Question2Answer seamlessly into openCARP.org in a sustainable way with reasonable effort, the official SnowFlat theme can be customized to follow the color scheme of

---

[19]https://www.question2answer.org, accessed July 28, 2021.
[20]https://askbot.com, accessed July 28, 2021.
[21]https://github.com/ricodigo/shapado, accessed July 28, 2021.
[22]https://www.atlassian.com/software/confluence/questions, accessed July 28, 2021.

openCARP.org. Askbot seems to be even better maintained but the different technology stack and smaller installation base tipped the scales.

2.4. Authentication and authorization

Regarding authentication and authorization, several requirements need to be met for the openCARP CDE:

- The system should employ uniform user accounts across all components of the CDE, so that users do not need to maintain separate user accounts.

- The service should be integrated in an external authentication and authorization infrastructure (AAI), removing the need for separate credentials to be created and maintained by the users.

- The used AAI should already have a wide usage in the academic community, ensuring a high probability that potential collaborators already have an account there.

- The used AAI needs to be trustworthy, sustainable and fulfill all legal requirements.

- The system needs to be capable of being integrated into the GitLab instance.

As of 2022, there were two systems worth considering with these requirements in mind: connecting to a Shibboleth based AAI or using the OpenID Connect service of ORCID.

Shibboleth is an internet-based single sign-on solution, which allows users to log in to different services using credentials they obtain from the organization they are affiliated with. These services are called Service Providers (SP). When a user tries to authenticate with the SP, a redirect is initiated to another web service, called Identity Provider (IdP), run by the organization of the user. The IdP then decides if the authentication was successful, usually by querying a local directory service (e.g. LDAP), and communicates the result to the SP. If the request was successful, the SP itself issues a query for additional attributes to the IdP (e.g. full name, email address) depending on the particular web service. Finally, the service provider decides if the authentication request was successful and continues with its own logic.

The benefit of this approach is that the user shares their credentials only with the home organization's IdP and the organization can enforce very fine-grained policies on which data are shared with the SP (which might be run by a completely different entity). There is, however, considerable configuration needed to set up such a system. Especially the connection between the SP and the IdP needs to be cryptographically secured. In order to simplify the process, organizations create federations[23]

---

[23]E.g., the AAI of the German research network (https://www.aai.dfn.de_aai), accessed July 28, 2021.

to simplify the process even further. Another limitation is that potential users might not be affiliated with home organizations serving as IdPs, for example non-academic users.

Introduced in 2012, the main objective of ORCID is to provide a unique code to identify scientific authors. In order to obtain this personal identifier, researchers need to create an account on the ORCID web page. For this purpose, ORCID was recently recommended by several institutions.[24] Based on the ORCID iD, ORCID introduced several tools and integrations. One of these services is the possibility to use ORCID as an OpenID Connect Provider.

OpenID Connect[25] is part of the OAuth2 authorization framework, which became widely popular with the advent of social networks. Smaller websites use login-with-Google or login-with-Facebook buttons to allow users to authorize with their social network accounts and share information or resources (e.g., pictures) from the network with the website. OpenID Connect offers a standardized authentication layer on top of OAuth2.

On a technical level, OpenID Connect works mostly on the client side and involves fewer negotiations between the web service, called Relying Party (RP) and the OpenID Provider (OP). Usually, the OP issues a client ID and a client secret, which needs be configured in the RP. When authenticating with the RP, the user first gets redirected to the OP, much like with Shibboleth. The user logs into the OP, if not already logged in, and acknowledges any authorization of additional information to the RP. Then, the user gets redirected back to the RP with a code added to the HTTP request. Using this code, the RP then issues a request to the OP (authentication itself with the client ID and secret) to obtain the previously authorized user credentials. Based on this information, the RP decides if the authentication was successful and continues with its logic. While large social networks are the most common OPs, ORCID is certainly the most suitable for the academic environment.

Both approaches have their benefits: Shibboleth authentication is common in the academic realm and in particular at universities. Furthermore, its decentralized approach prevents being dependent on a central actor. There is, however a considerable technical overhead in setting up and maintaining a SP and in negotiating the exchange of metadata with different institutions and/or federations. openCARP has potential users all over the world, and therefore a large number of IdPs need to be supported. Moreover, not all institutions (e.g. independent research institutions, commercial parties) are part of a Shibboleth federation and would need to apply for user accounts in academic institutions, which is often a long and bureaucratic process.

---

[24]E.g. Paul Vierkant et al., *Autorenidentifikation anhand der Open Researcher and Contributor ID (ORCID) - Positionspapier* (Berlin: Humboldt-Universität zu Berlin, 2018), doi:10.18452/19528.
[25]OpenID Foundation, "OpenID Connect Core 1.0," 2014, https://openid.net/specs/openidconnect-core-1_0.html, accessed July 28, 2021.

In contrast, ORCID is a centralized platform. From the point of view of the operators of the CDE, this considerably simplifies the process of integrating an external authentication provider. Since OpenID Connect is a well-known industry standard, the process is very streamlined and well-documented. For users, it is relatively uncomplicated to obtain an ORCID iD, even if they are not affiliated with a research institution, and an ORCID iD offers additional advantages as explained above. While using an external site to store identity information raises questions about data privacy, ORCID offers detailed control about which data are stored and shared with the public or with other services.[26] To be able to receive user's email addresses during account creation, the hosting institution needs to be a paying ORCID member or part of an ORCID consortium[27].

In conclusion, ORCID is the designated choice for the openCARP CDE, since it combines a low entry threshold for an international academic and non-academic audience with manageable maintenance efforts. The privacy issues, which come with using a central platform, are sufficiently addressed. Regarding levels of assurance about the user identity, ORCID meets our rather low requirements.

Unfortunately, as of 2022, ORCID can only be used as authentication provider with GitLab if it is integrated using a customized GitLab package. This package needs to be built again for each new version of GitLab and the process involves several manual steps. This additional effort constitutes a serious challenge for the sustainable operation and maintenance of the whole environment. While we still think that ORCID authentication would be beneficial to the environment, we postponed the integration to minimize the overhead for a sustainable operation of the CDE. Until this issue is resolved, users need to register using their email and a password or using a GitHub or GitLab.com account.

2.5. Citation and long term preservation

One of the key features of the openCARP CDE is to enable researchers to properly cite the software in their work and, thus, attributing the effort of the developers and maintainers. In order to be citable, software releases need to be properly archived in a repository and assigned a persistent identifier (PID), preferably a digital object identifier (DOI)[28]. Connected with the creation of a DOI is the collection of descriptive metadata about the dataset (in this case the software release) in form of the DataCite

---

[26]See also Jan Schallaböck and Max von Grafenstein, *ORCID from a data protection perspective [Expert opinion on behalf of the German Research Foundation (DFG) as part of the project ORCID DE for the promotion of the Open Researcher and Contributor ID in Germany]* (Berlin: iRights.Law Rechtsanwälte, 2017), doi:10.2312/lis.18.02.

[27]ORCID membership: https://info.orcid.org/about-membership/, accessed July 28, 2021.

[28]While DOI is the most common, other systems exist, are discussed, and exhibit different advantages and disadvantages (Daniel S. Katz, "Second thoughts on Proper Citation Guidance for Software Developers," 2019, https://danielskatzblog.wordpress.com/2019/12/20/thoughts-on-citation-guidance-for-developers/, accessed July 28, 2021).

metadata schema.[29] These metadata contain information like the title, creators, subjects and publication dates but also information on who contributed to the curation and archiving process.

This practice, initially coming from conventional publications, was recently adopted for research data and is now used for software as well. In this context, the FAIR principles[30] for research data serve as guiding principles and can be adopted to software, although, for specific characteristics of software, they need to be extended.[31] A number of guidelines around software publication and citation supporting this approach have been published.[32]

For the openCARP infrastructure, these practices result in a set of additional requirements:

- the software needs to be long-term archived in a suitable repository,

- a DOI needs to be registered for every release, and

- additional metadata, which are not directly connected to the development process need to be collected and maintained.

In order to be actually able to run the code in the version it was referred to in a publication, it is advisable to not only preserve the source code of the release, but also a number of additional assets:

- the compiled docker image,

- the compiled binary RPM distribution package (for CentOS systems),

- the compiled binary DEB distribution package (for Debian or Ubuntu systems),

- the compiled binary pkg distribution package (for macOS systems),

- the user manual as PDF,

- additional components like the carputils Python framework at the matching revision.

---

[29] DataCite Metadata Working Group, *DataCite Metadata Schema Documentation for the Publication and Citation of Research Data. Version 4.3.*, 2019, doi:10.14454/7xq3-zf69.

[30] Mark D. Wilkinson et al., "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data* 3, no. 1 (March 2016): 160018, ISSN: 2052-4463, doi:10.1038/sdata.2016.18.

[31] Anna-Lena Lamprecht et al., "Towards FAIR principles fo research software," *Data Science* (2019) 1–23, ISSN: 2451-8492, doi:10.3233/DS-190026; Daniel S. Katz, Morane Gruenpeter, and Tom Honeyman, "Taking a fresh look at FAIR for research software," *Patterns* 2, no. 3 (March 2021): 100222, doi:10.1016/j.patter.2021.100222.

[32] See Arfon M. Smith, Daniel S. Katz, and Kyle E. and Niemeyer, "Software citation principles," *PeerJ Computer Science* 2 (September 2016): ISSN: 2376-5992, doi:10.7717/peerj-cs.86; Katharina Sara Scheliga et al., *Dealing with research software: Recommendations for best practices* (Potsdam, 2019), doi:10.2312/os.helmholtz.003; Katz, "Second thoughts on Proper Citation Guidance for Software Developers," 2019, https://citation-file-format.github.io, accessed July 28, 2021.

This release package, which is called submission information package (SIP) in terms of the OAIS reference model[33] needs to be saved in a way that it can be reliably stored and transferred. A common specification for this kind of packages is BagIt, which was developed by the Library of Congress and is defined in RFC 8493.[34] BagIt is a hierarchical file layout convention, which is simple to implement in common file systems. A so-called *bag* directory contains the data directory with the actual *payload*, as well as *tags*, which are files containing technical and organizational metadata, e.g., file checksums or contact email addresses. BagIt can be considered mature and is widely adopted by archives and libraries.[35] In addition, several software libraries supporting the creation and handling exist[36]. Recently, the Research Data Repository Interoperability Working Group of the Research Data Alliance extended BagIt towards BagPack by allowing for content-specific metadata in a dedicated *metadata* directory inside the bag structure.[37] In principle, different extended metadata standards can be added to a BagPack. However, the before-mentioned DataCite metadata schema, which is the minimum information that needs to be provided to get a DOI, is mandatory for BagPack.
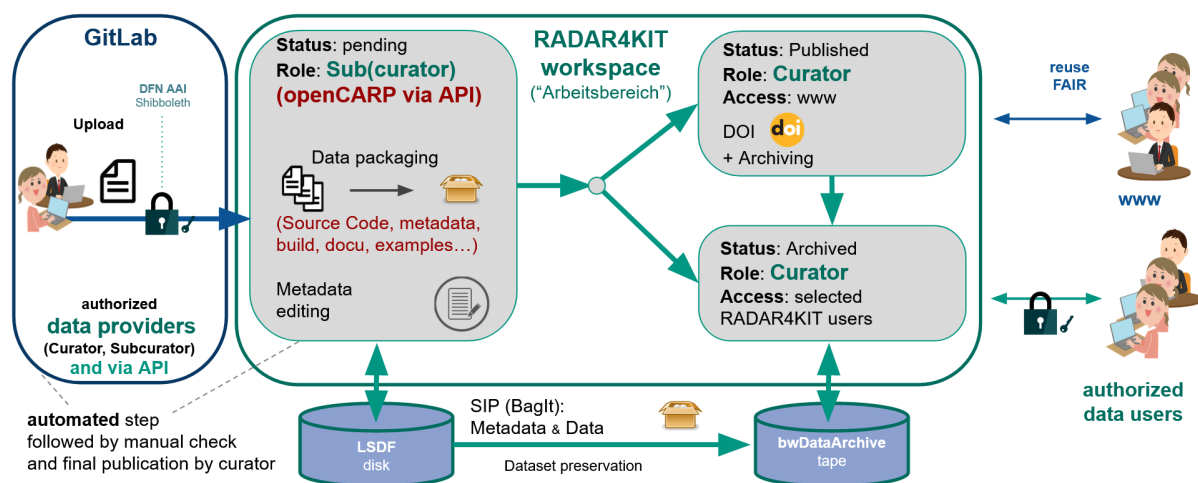

Figure 3: RADAR4KIT publication and archiving workflow of software releases from GitLab repositories

For the archival of the created package, several established options were considered. KIT, which hosts the openCARP infrastructure, operates a long-term preservation ser-

---

[33] CCSDS 650.0-B-1, *Reference Model for an Open Archival Information System (OAIS)*, 2012, https://public.ccsds.org/Pubs/650x0m2.pdf, accessed July 28, 2021.

[34] J. Kunze et al., *The BagIt File Packaging Format (V1.0)*, 2018, https://tools.ietf.org/html/rfc8493, accessed July 28, 2021.

[35] E.g. Steffen Fritz, "Die Anwendung des BagIt-Formats im Deutschen Literaturarchiv Marbach," *B.I.T. Online* 17 (January 2014): 109–114, https://www.b-i-t-online.de/heft/2014-02/fachbeitrag-fritz.pdf, accessed July 28, 2021.

[36] We use `bagit-python` maintained by the Library of Congress, https://github.com/libraryofcongress/bagit-python, accessed July 28, 2021.

[37] Research Data Repository Interoperability WG, *Research Data Repository Interoperability WG Final Recommendations*, 2018, doi:10.15497/RDA00025.

vice called bwDataArchive[38] This service can be used to reliably archive a BagIt or BagPack package. RADAR4KIT[39] is operated by KIT as well and is a cooperation with RADAR service[40]. It provides a user-friendly web-based workflow to create and archive datasets which is shown in figure 3. RADAR uses a metadata schema, which is very similar to DataCite. A possible external service, which is open for all kinds of research data and publications is Zenodo[41]. Zenodo is operated by CERN and was developed under the openAIRE programme. With both RADAR and Zenodo, files are uploaded via HTTP and metadata are added using a web interface or a JSON-based API. In both cases, the creation of the concrete archival package (as stored by the service provider) is not under the control of the user. Since KIT is using RADAR4KIT as a generic institutional repository for research data, we wanted to use it as a depositing solution. This makes it easy to find software contributions by KIT scientists and extract its metadata, e.g. for institutional publication lists.

- As part of the release process of a particular version of openCARP, the assets described above are automatically uploaded to RADAR4KIT and the corresponding metadata are added using the API of the service. After publication, a DOI is created automatically by the RADAR service and can be used to reference the software release in a text publication or from a different dataset.

- Optionally or as an alternative, an additional BagPack can be created and archived in the bwDataArchive infrastructure. The resulting package is not publicly available and is used for long-term preservation only.

While this strategy relies on infrastructure that is specific to the particular institutional circumstances of the openCARP project, it can be easily adapted to use other infrastructure, e.g., Zenodo instead of RADAR4KIT and any long-term bitstream preservation service instead of bwDataArchive.

In both cases, metadata need to be aggregated as part of the release process. The DataCite schema contains a number of mandatory and optional fields to describe the resource. The fields which we selected for openCARP are given in Table 1 of Appendix A. In order to integrate the maintenance of the metadata into the regular software development process, two files were added to the openCARP software repository: the `CONTRIBUTORS.yml` file contains the list of people involved in the development, publication and preservation of openCARP (Creators and Contributors in the DataCite schema) and the `METADATA.yml` file contains the other metadata fields. Both files are machine-readable using the YAML format and use naming conventions used in software engineering as well as the DataCite terminology (see Listing 1 of Appendix A). To

---

[38]bwDataArchive, https://www.rda.kit.edu/english/, accessed July 28, 2021.
[39]RADAR4KIT, https://radar.kit.edu, accessed July 28, 2021.
[40]RADAR, https://www.radar-service.eu, accessed July 28, 2021.
[41]Zenodo https://zenodo.org/, accessed July 28, 2021.

interoperably identify persons and institutions in the contributor lists, we employ the well-established ORCID[42] and ROR[43] identification systems.

In order to automatically perform all steps to create the archive package and to aggregate the metadata, we integrate three jobs into the release CI (see also Section 2.1):

1. Based on the `CONTRIBUTORS.yml` and the `METADATA.yml` YAML files in the openCARP repository and the version tag of the release, the `datacite.xml` file is created and added to the release assets.

2. The release assets are collected and a BagPack archive is created. The archive is copied to a non-public location in the bwDataArchive infrastructure.

3. Using its REST API, a dataset is created in RADAR4KIT and metadata is added based on the YAML files in the openCARP repository. In a second step, the assets are uploaded into the dataset. When the curator publishes the dataset with a single click in the RADAR4KIT interface, RADAR4KIT creates a DOI which can be used to cite the release.[44]

3. Evaluation and Discussion

The described infrastructure was and is intensively used in the development of the openCARP software, which provides an open cardiac electrophysiology simulation solution[45]. In the field of cardiac electrophysiology, computational modeling and simulation are becoming increasingly important. openCARP offers single cell as well as multiscale simulations from ion channel to organ level. Additionally, it includes a wide variety of functions for pre- and post-processing of data as well as visualization. The Python-based carputils framework enables the user to develop and share simulation pipelines, i.e., automating *in silico* experiments including all modeling/simulation steps. openCARP was derived from the existing closed-source CARPentry solution, which had grown organically over more than 15 years. The software architecture was adapted to meet new requirements and provide a user-friendly interface. The development team comprises contributors and maintainers from several countries. As a development platform, the described CDE was introduced in an early stage while still being in development. This allowed early adoptions of user- and developer-requirements and enabled a real-world evaluation of the CDE components and their interplay. The system was intensively used and well accepted by the developer community. The environment proved to serve its purpose with currently more than 250 registered users in the openCARP GitLab instance.

---

[42]ORCID, https://orcid.org/, accessed July 28, 2021.
[43]ROR, https://ror.org/, accessed July 28, 2021.
[44]E.g. openCARP consortium et al., *openCARP (v5.0)*, 2021, doi:10.35097/389.
[45]Plank et al., "openCARP".

An increase of resources - such as disk space of the used virtual machine - would ease its operation and make it more scalable for several hundreds of users and multiple simultaneous jobs. A concept for "on-the-fly scaling" could be applied to keep the costs minimal while providing the needed resources. Regarding the technical hosting and management of the systems, the costs are relatively low with about 90 Euros per month for the two virtual machines, including a full backup. Maintaining the web server and GitLab requires very little maintenance of around 1 person hour per month for a trained system operator with a good knowledge of the technical systems. We successfully use automatic updates and backups for both the system and GitLab. The infrastructure is based on the CentOS Linux 7 distribution, which is supported until mid of 2024. The CDE needs to be adapted or migrated to another distribution then. The used software is based on GitLab, which is open source software, and apart from GitHub, the most popular Git repository management software. As expected, it was received well by the developers and users. We use only features available in the free Community Edition of GitLab.

The modular approach of building on existing components and connecting them allows for reuse in other research data management infrastructures. For example, the described concept for citation and long term preservation will be adopted by MO|RE data[46], a research data repository for sports science.

4. Conclusion

The presented CDE framework consisting of a DevOps platform, an interactive project web page and tailored continuous integration pipelines serves the purpose of providing a comprehensive and smooth yet lightweight infrastructure for research software projects. The openCARP CDE combines several existing open pieces of software to an integrated and comprehensive solution. The successful real-world test by the openCARP user and developer community suggests that this collaborative development environment could be a suitable solution for a wide range of research software projects to support sustainability. The modular architecture enables straightforward exchange or addition of components.

Acknowledgments

---

[46]MO|REdata, http://motor-research-data.org, accessed July 28, 2021.

A. DataCite metadata for openCARP

In order to (automatically) create DataCite metadata for the openCARP releases, we selected a subset of the DataCite 4.3 kernel to describe the software. The selection is given in Table 1.

While some of the metadata, like the version and the issue data can be automatically extracted from the GitLab system, most of the metadata is provided from a `METADATA.yml` file which is part of the openCARP Git repository. Listing 1 shows the content of this file.

Id	Property	Description
1	Identifier	The DOI to be registered. With RADAR, this cannot be determined before creating the archive, it is therefore skipped for the local creation of the BagPack.
2	Creator	The list of researchers and developers involved in the creation of openCARP. For each person, properties 2.1-2.5 are collected.
2.1	creatorName	Full name of the creator.
2.2	givenName	Given name of the creator.
2.3	familyName	Family name of the creator.
2.4	nameIdentifier	ORCID iD of the creator.
2.4.a	nameIdentifierScheme	The scheme of identifier for 2.4: `ORCID`.
2.4.b	schemeURI	The URI of the scheme for 2.4: http://orcid.org.
2.5	affiliation	The affiliation(s) of the creator.
2.5.a	affiliationIdentifier	The ROR of the affiliation.
2.5.b	affiliationIdentifierScheme	The scheme of identifier for 2.5: `ROR`.
2.5.c	SchemeURI	The URI of the scheme for 2.5: http://ror.org.
3	Title	The name or title by which the resource is known: openCARP. As the alternative title: Cardiac Electrophysiology Simulator

Id	Property	Description
3.a	titleType	Only used for the alternative title: `AlternativeTitle`
4	Publisher	The institution responsible for the publication of the resource: `Karlsruhe Institute of Technology (KIT)`.
5	PublicationYear	The year of the particular release.
6	Subject	The list of subjects for openCARP, taken from the Library of Congress Subject Headings.
6.b	schemeURI	The URL for the Library of Congress Subject Headings: http://id.loc.gov/authorities/subjects.
6.c	valueURI	The URL for the particular subject, e.g. http://id.loc.gov/authorities/subjects/sh85082124.
7	Contributor	The list of persons and institutions involved in the publication and archival of open-CARP. For each person/entity, properties 7.1-7.5 are collected.
7.a	contributorType	Type of contributor from the DataCite controlled vocabulary, e.g. `DataCurator`, `HostingInstitution`.
7.1	contributorName	Full name of the contributor.
7.2	givenName	Given name of the contributor.
7.3	familyName	Family name of the contributor.
7.4	nameIdentifier	ORCID iD of the contributor.
7.4.a	nameIdentifierScheme	The scheme of identifier for 7.4: `ORCID`.
7.4.b	schemeURI	The URI of the scheme for 7.4: http://orcid.org.
7.5	affiliation	The affiliation(s) of the contributor.
7.5.a	affiliationIdentifier	The ROR of the affiliation.
7.5.b	affiliationIdentifierScheme	The scheme of identifier for 7.5: `ROR`.
7.5.c	SchemeURI	The URI of the scheme for 7.5: http://ror.org.
8	Date	Important dates for the resource, for open-CARP we use `Created` for the date of the release and `Issued` for the date of the upload in the archive.

Id	Property	Description
8.a	dateType	The type of date from the DataCite vocabulary: `Created` or `Issued`.
9	Language	Language of the resource: `en-US` for American English.
10	ResourceType	The type of resource: `Simulation code`.
10.a	resourceTypeGeneral	The general type of resource from the DataCite vocabulary: `Software`.
11	AlternateIdentifier	The list of altenative identifiers for the resource. We only give the link to the release page in the openCARP GitLab repository: https://git.openCARP.org/openCARP/openCARP/-/releases.
11.a	alternateIdentifierType	The type of identifier for 11: `URL`.
12	RelatedIdentifier	The list of related identifiers for the resource. We use `IsNewVersionOf` to link back to an older release of openCARP and `IsVersionOf` to link to a concept DOI for openCARP.
12.a	relatedIdentifierType	The type of identifier for 12: `DOI`.
12.b	relationType	The type of relation for 12: `IsNewVersionOf` and `IsVersionOf`.
15	Version	The version of the openCARP release, identical to the tag in the software repository.
16	Rights	The licence for openCARP: "ACADEMIC PUBLIC LICENSE (openCARP, v1.0)"
16.a	rightsURI	The URL where the licence can be found: https://openCARP.org/download/license
17	Description	A short description of openCARP.
17.a	descriptionType	The type of the description for 17: Abstract
19	FundingReference	A list containing references about funding.
19.1	funderName	Name of the funder.
19.2	funderIdentifier	ROR of the funder.
19.2.a	funderIdentifierType	The type of identifier for 19.2: `ROR`.
19.2.b	SchemeURI	The URI of the scheme for 19.2: https://ror.org.
19.3	awardNumber	The identifier for the funding grant/award assigned by the funder.
19.3.a	awardURI	URL for more information about the funding grant/award.

Id	Property	Description
19.4	awardTitle	Title for the funding grant/award.

Table 1: Adoption of the DataCite 4.3 schema for openCARP. DataCite properties not suitable for openCARP are omitted from the table

```
1   title: openCARP
2   additional_titles:
3   – additional_title: Cardiac Electrophysiology Simulator
4     additional_title_type: AlternativeTitle
5   keywords:
6   – cardiac electrophysiology
7   – simulation software
8   – in silico medicine
9   – bidomain theory
10  – computational cardiology
11  publisher: Karlsruhe Institute of Technology (KIT)
12  descriptions:
13  – description: openCARP is an open cardiac electrophysiology simulator
        for in silico experiments.
14    description_type: Abstract
15  subjects:
16  – subject: Mathematical models
17    value_uri: http://id.loc.gov/authorities/subjects/sh85082124
18    scheme_uri: http://id.loc.gov/authorities/subjects
19  – subject: Computer simulation
20    value_uri: http://id.loc.gov/authorities/subjects/sh85029533
21    scheme_uri: http://id.loc.gov/authorities/subjects
22  – subject: Biomedical engineering
23    value_uri: http://id.loc.gov/authorities/subjects/sh85014237
24    scheme_uri: http://id.loc.gov/authorities/subjects
25  – subject: Cardiology
26    value_uri: http://id.loc.gov/authorities/subjects/sh85020214
27    scheme_uri: http://id.loc.gov/authorities/subjects
28  – subject: Biophysics
29    value_uri: http://id.loc.gov/authorities/subjects/sh85014253
30    scheme_uri: http://id.loc.gov/authorities/subjects
31  radar_subjects:
32  – LifeScience
33  – ComputerScience
34  – Medicine
35  resource: Simulation code
36  resource_type: Software
37  alternate_identifiers:
38  – alternate_identifier: https://openCARP.org/download/releases
39    alternate_identifier_type: URL
40  related_identifiers:
41  – relation_type: IsVersionOf
42    related_identifier:
43    related_identifier_type: DOI
44  – relation_type: IsNewVersionOf
```

```
45        related_identifier:
46        related_identifier_type: DOI
47    rights: ACADEMIC PUBLIC LICENSE (openCARP, v1.0)
48    rights_url: https://openCARP.org/download/license
49    rights_holder: NumeriCor GmbH
50    funding_references:
51    - name: Deutsche Forschungsgemeinschaft
52      ror: https://ror.org/018mejw64
53      award_number: 391128822
54      award_uri: https://gepris.dfg.de/gepris/projekt/391128822
55      award_title: "Sustainable Lifecycle Management for Scientific Software
              (SuLMaSS) - Software Dissemination and Infrastructure Development
            Driven by a Cardiac Electrophysiology Simulator"
```

Listing 1: METADATA.yml for openCARP