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

Approaching automation of multiple instance orchestration of the menoci web portal

Luca Freckmann[i] Christian Henke[ii] Robert Kossen[iii]
Linus Weber[iv] Ulrich Sax[v] Sara Y. Nussbeck[vi]
Harald Kusch[vii]

2023

Zitiervorschlag

Freckmann, Luca, Henke, Christian, Kossen, Robert, Weber, Linus, Sax, Ulrich, Nussbeck, Sara Y., Kusch, Harald. 2023. Approaching automation of multiple instance orchestration of the menoci web portal. *Bausteine Forschungsdatenmanagement. Empfehlungen und Erfahrungsberichte für die Praxis von Forschungsdatenmanagerinnen und -managern* Nr. 5/2023: S.3-11 DOI: 10.17192/bfdm.2023.5.8608.

[i]Department of Medical Informatics, University Medical Center Göttingen (UMG), Göttingen, Germany. ORCID: 0000-0002-8285-2586
[ii]Department of Medical Informatics, University Medical Center Göttingen (UMG), Göttingen, Germany. ORCID: 0000-0002-4541-4018
[iii]Department of Medical Informatics, University Medical Center Göttingen (UMG), Göttingen, Germany. ORCID: 0000-0003-1236-0815
[iv]Department of Medical Informatics, University Medical Center Göttingen (UMG), Göttingen, Germany. ORCID: 0000-0001-7973-7491
[v]Department of Medical Informatics, University Medical Center Göttingen (UMG), Göttingen, Germany; Campus-Institute Data Science (CIDAS), Göttingen. ORCID: 0000-0002-8188-3495
[vi]Department of Medical Informatics, University Medical Center Göttingen (UMG), Göttingen, Germany; Central Biobank UMG, UMG, Göttingen, Germany. ORCID: 0000-0003-1223-6494
[vii]Department of Medical Informatics, University Medical Center Göttingen (UMG), Göttingen, Germany; Campus-Institute Data Science (CIDAS). ORCID: 0000-0002-9895-2469

Abstract

Introduction: The *menoci* web portal addresses the needs of FAIR representation of biomedical basic research data and has been successfully implemented for several large consortia at Göttingen Campus. The operation of multiple menoci instances requires efficient measures to reduce administrative resource efforts. This manuscript describes our approach to automatize server operation and software updating procedures. ***Methods:*** The *menoci* instances are hosted on virtual machines (VM) using IT infrastructure of the local academic IT-service provider. Source code and process documentation is hosted in the Göttingen Campus *GitLab* service. Continuous Integration/Continuous Delivery (CI/CD) pipelines were developed to routinely build updated *Docker* images from latest source code revisions and the upstream *Drupal Docker* image. *GitLab* functionality for code reviews is employed, using protected branches and the "approval" feature for merge requests. ***Results:*** At the beginning, *menoci* development was mainly driven by the implementation of additional modules, features and optimization of user experience to fulfill the researchers' requirements. Since the roll out of *menoci* to an increasing list of research consortia, we additionally focused on improving performance, software quality and enhanced automation processes. Our developed automation pipelines include updates for web server and database components, as well as the *Drupal* content management system and other components that together form the *menoci* platform. Furthermore, all *menoci* code enhancements are automatically distributed to all instances. Success or failing of update processes is monitored systematically to facilitate error handling. All processes are extensively documented to easily integrate new team members into administrative tasks. ***Discussion:*** Our experience indicated that automation processes ar key to reduce resource efforts for technical administrative tasks. However, a high degree of automation and dependencies invoke the potential of small errors possibly leading to large effects. Therefore, tight quality control by testing and monitoring processes is necessary.

Keywords. Research data platform, data management, automated processes, Continuous Integration/Continuous Delivery (CI/CD)

1 Introduction

In the life sciences, FAIRification tools that are designed to represent the research output of large scientific consortia are still limited and are actively developed[1].

Since 2012 our *menoci* software development approach addresses the needs of FAIR representation of biomedical basic research data and has been successfully imple-

[1]Online: FAIRification Process - GO FAIR. 2019. https://www.go-fair.org/fair-principles/fairification-process/, accessed 2023-07-18.

mented for several large consortia at Göttingen Campus[2] [3]. *menoci* offers an open-source data management web portal that can be customized and rapidly deployed. In the beginning of the project, *menoci* development was mainly driven by the implementation of additional modules, features and optimization of user experience to fulfill the researchers' requirements. Several life sciences associated research consortia at Göttingen Campus, i.e., CRC 1002, CRC 1190, CRC 1286, CRC 1565, TRR 274, RU 2705, RU 2848, and the cluster of excellence MBExC, included the *menoci* web portal[4] as information hub for their research data management strategies. At the organizational and technical level, this requires the operation of multiple instances of *menoci* on thirteen servers because each consortium had a variety of overlapping requirements, but also some important distinct feature needs.

Although web resource orchestration processes and techniques have been available for a long time in different application fields, our use case cannot be covered by simply reusing available templates as customizable out-of-the-box technology and configuration templates are not readily available[5]. Frequently, the orchestration process involves "Continuous Integration / Continuous Delivery" (CI/CD) techniques like CI/CD pipelines[6] [7] [8] [9]. In addition, professional digital infrastructure operation – especially complex web resource orchestration – within academic research institutes is challenging due to limited personnel resources for the management of such infrastructures as well as the required expertise. Several hazards reaching from system downtime to manual update deployment can be diminished by increasing automation[10].

[2] Suhr M, Lehmann C, Bauer CR, Bender T, Knopp C, Freckmann L, u. a. Menoci: lightweight extensible web portal enhancing data management for biomedical research projects. *BMC Bioinformatics*. 2020;21(1): 582. https://doi.org/10.1186/s12859-020-03928-1.

[3] Kusch H, Kossen R, Suhr M, Freckmann L, Weber L, Henke C, Lehmann C, u. a. Management of Metadata Types in Basic Cardiological Research. *Stud Health Technol Inform*. 2021: 59–68. https://doi.org/10.3233/SHTI21054.

[4] Online: menoci. https://menoci.io, accessed 2023-07-18.

[5] Weerasiri D, Barukh MC, Benatallah B, Sheng QZ, Ranjan R. A Taxonomy and Survey of Cloud Resource Orchestration Techniques. *ACM Computing Surveys*. 2017;50(2): 26:1-26:41. https://doi.org/10.1145/3054177.

[6] Duvall PM, Matyas S, Glover A. *Continuous Integration: Improving Software Quality and Reducing Risk*. 1. Aufl. Upper Saddle River, NJ: Addison-Wesley Professional, 2007. ISBN: 978-0-321-33638-5582.

[7] Vassallo C, Proksch S, Jancso A, Gall HC, Di Penta M. Configuration smells in continuous delivery pipelines: a linter and a six-month study on GitLab. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*; 2020; New York, NY, USA: Association for Computing Machinery. p. 327–37. https://doi.org/10.1145/3368089.3409709

[8] Hilton M, Tunnell T, Huang K, Marinov D, Dig D. Usage, costs, and benefits of continuous integration in open-source projects. In: *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*; 2016; New York, NY, USA: Association for Computing Machinery. p. 426–37. https://doi.org/10.1145/2970276.2970358

[9] Vasilescu B, Yu Y, Wang H, Devanbu P, Filkov V. Quality and productivity outcomes relating to continuous integration in GitHub. In: *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*; 2015; New York, NY, USA: Association for Computing Machinery. p. 805–16. https://doi.org/10.1145/2786805.2786850

[10] Díaz J, López-Fernández D, Pérez J, González-Prieto A. Why Are Many Businesses Instilling a DevOps

To address these challenges in the context of multiple instances of the *menoci* platform, we focused on improving performance, software quality, enhanced automation processes, and detailed documentation.

Here, we describe our approach towards sustainable data management, software development, and service operation.

2 Methods

Multiple instance operation requires efficient measures to avoid errors, e.g., in update processes, and to reduce administrative resource efforts. Therefore, a multitude of conceptual requirements need to be determined regarding several aspects of the automation routine.

2.1 Setup fundamentals

The *menoci* instances are hosted on virtual machines (VMs) using IT infrastructure of the local academic IT-service provider Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)[11]. To avoid the need to customize the set of requirements for all processes included to run a *menoci* instance, the services are encapsulated in containers. These containers are maintained individually by open-source communities and are provided by external partners in the form of downloadable images. Images represent a set of pre-configured, essential requirements for a software to be able to run in an isolated area. An instantiated image becomes a container that runs on the VM without creating interference with the operating system (OS) or other containers. Containers are embedded in a virtual environment on the VM which has its own network to enable communication among the containers. Communication between the inside and the outside of this environment is strictly regulated[12]. An orchestration of the initial setup and configuration processes of the containers and their communication is achieved by defining templates for containers.

2.2 menoci source code

The *menoci* source code as well as pertinent documentation is hosted and maintained in the Göttingen Campus *GitLab* service[13]. *menoci* source code comprises *Drupal* and

Culture into Their Organization? *Empirical Software Engineering.* 2021;26(2): 25. https://doi.org/10.1007/s10664-020-09919-3.

[11] Online: https://www.gwdg.de/, accessed 2023-07-18.

[12] Zhou N, Zhou H, Hoppe D. Containerisation for High Performance Computing Systems: Survey and Prospects. *IEEE Transactions on Software Engineering*. 2022: 1-20. https://doi.org/10.1109/TSE.2022.3229221.

[13] Online: *menoci* source code. https://gitlab.gwdg.de/medinfpub/menoci, accessed 2023-09-18.

several *Drupal* modules developed by both the *menoci* development team and third parties contributing to the *Drupal* community. The self-developed *menoci* modules reside in the local *GitLab* and the external modules as well as *Drupal* itself are accessible via internet. *menoci* modules are maintained as separate git repositories. A specific subset of *menoci* modules represents the core functionalities and needs to be included in every *menoci* instance.

GitLab functionality for code reviews is employed, using protected branches and the approval feature for merge requests. These features provide the development process and the maintenance of the code base with state-of-the-art technologies, ensure better code quality and facilitate agile development. Furthermore, pipelines can aggregate main branches from *menoci* repositories to the latest feature composition of *menoci* source code with access rights granted by deploy tokens specified for this purpose.

2.3 Approaching automation

The first conceptual approach to automation aims to tackle two areas where avoiding manual effort seems most profitable. The first one is the process of updating *menoci* instances to the latest source code revisions and updating all external resources. The second one is the deployment of all updates to the running *menoci* instances with minimal downtime. This represents the common division in CI/CD[14].

A CI/CD pipeline based on *GitLab CI/CD* is used to routinely build updated images from latest source code revisions of *menoci* modules and the upstream *Drupal* image. The pipeline also comprises specific customizations and configurations for the images of different instances. After the build process is complete, the images are uploaded to the *GitLab container registry*. Images residing in a container registry can be accessed by a tag and deployed to VMs. A second CI/CD pipeline has access to the VMs of *menoci* productive instances and has permissions to pull and deploy the latest images to the VM. Upon failure at any given point in the routines of the pipelines, there is a fallback solution to the images from the last successful build or deploy routine.

Our developed automation pipelines include update routines for the web servers and database components as well as the *Drupal* content management system which are depicted in Figure 1. Besides these core elements of our software stack the automation pipeline comprises update routines for several other components such as reverse proxies and backup client software. Updates related to the VM's operating system and configurations are handled by routines initialized once. Success or failing of update processes is monitored systematically to facilitate error handling. The CI/CD pipelines run sequentially to allow for an abortion of the delivery process if unwanted behavior is

[14] Arachchi S, Perera I. Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management. IEEE Xplore. 2018. https://doi.org/10.1109/MERCon.2018.8421965.

monitored in earlier steps. All processes are extensively documented to change or add configurations and easily integrate new team members into administrative tasks.

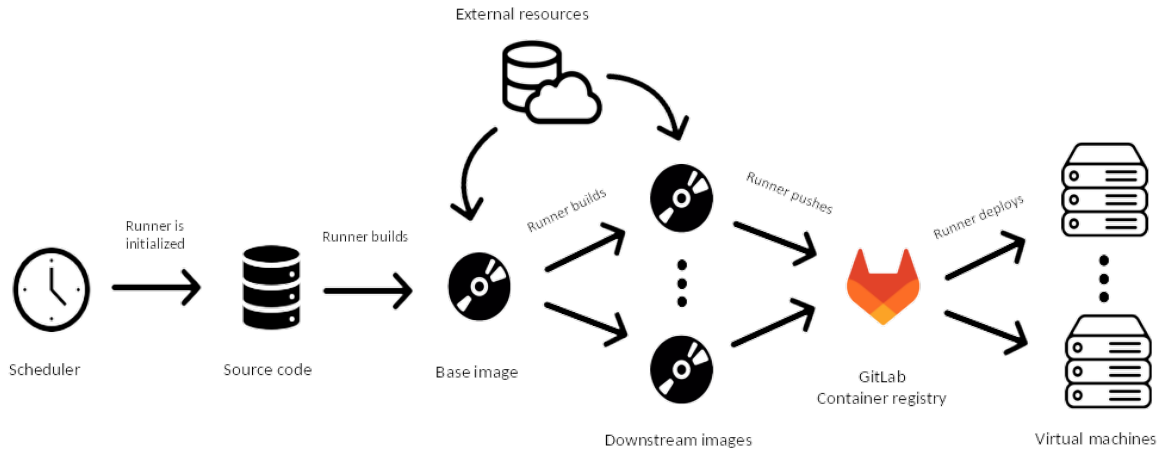

Figure 1: Automation pipelines sequence and coverage for multi-instance operation of *menoci*. Icon sources: *Flaticon* and *GitLab*.

3 Results

Our aim to employ sustainable software development strategies and coordinate the operation of multiple server applications in the context of the *menoci* platform, proved to be a complex task. The fast and frequent roll-out of features or fixes is important for the *menoci* instances in order to maintain efficiency and relevance in their respective research contexts. Here we describe the current setup that reduces maintenance effort and increases the degree of standardization of separate instances of similar software stacks.

To utilize the whole CI/CD routine and to facilitate handling the *menoci* code base, migration to a containerized solution was necessary. *Docker*[15] and *Docker Compose*[16] formed the basis for the container structure. *GitLab's* inherent CI/CD features as well as the support of *Docker* images in *GitLab's* container registries complemented the structure to fully support the automation with CI/CD pipelines.

3.1 Docker integration of menoci code

Drupal forms the basis of the *menoci* platform and it is available as a maintained Docker image. Therefore, the *menoci* code base was integrated into a *Docker* image by

[15] Online: Docker. https://www.docker.com/, accessed 2023-07-18.
[16] Online: Docker Compose. https://docs.docker.com/compose/, accessed 2023-07-18.

extending the *Drupal Docker* image with *menoci* code. The integration was achieved by creating a *Docker* image that was based on the *Drupal Docker* image and depositing the latest revisions of *menoci* repositories and configuration files in the image. Furthermore, dependencies for *menoci* are installed on the image to ensure compatibility.

This structure does not only assure work with an up-to-date code base, but also harmonizes various processes and structural differences of instances. For example, the installation paths are now identical for all downstream images and the instances have the same base, so the variations of versions and dependencies are much smaller.

3.2 Continuous integration features utilized for automation

GitLab's CI/CD features support the use of *Docker* images and comprise the entire CI/CD routine ranging from schedulers to container registries. For our purposes, *GitLab runner* was used to perform all executions of pipelines. The runner itself is available in an encapsulated *Docker* image. The runner is hosted on a dedicated VM and it is instantiated as a *Docker* container by the *GitLab* Scheduler. Furthermore, the runner is registered as a project runner in a specific CI/CD *GitLab* repository. A project runner offers the most privacy and therefore security when using credentials for several VMs. This *GitLab* repository also includes a CI/CD configuration file in *yet another markup language* (YAML)-format, which defines the tasks performed in a pipeline. Additionally, *GitLab CI/CD* offers to store and hide VM credentials that can be accessed from the YAML file. After registering in the repository's container registry, the runner is able to push and store the images to it.

3.3 Pipeline staging and multiple pipelines

Different tasks of the CI/CD routine needed to be automated. We achieved subdivision of tasks by using different stages of pipelines. For example, the first stage builds the base image and the second stage uses the base image to build the downstream images (Figure 1). This allowed for a more fine-grained pipeline structure and a sequential order of tasks. However, decoupling tasks in time requires multiple pipelines and schedulers, as one pipeline executes all stages if it is not interrupted. Multiple pipelines in one repository are not natively supported by *GitLab's CI/CD* but can be achieved by using the repository's branching feature.

Figure 2 summarizes how components of a pipeline interact with each other and where they are deployed. Especially, features from *GitLab* and the interactions are depicted. All connections among components are secured by either an encryption or an encapsulation in an isolated subnetwork. Additionally, the entire CI/CD routine is located in an isolated network dedicated for science at Göttingen Campus. Components like the

runner can access the internet from within the network, but only authorized users can access components.

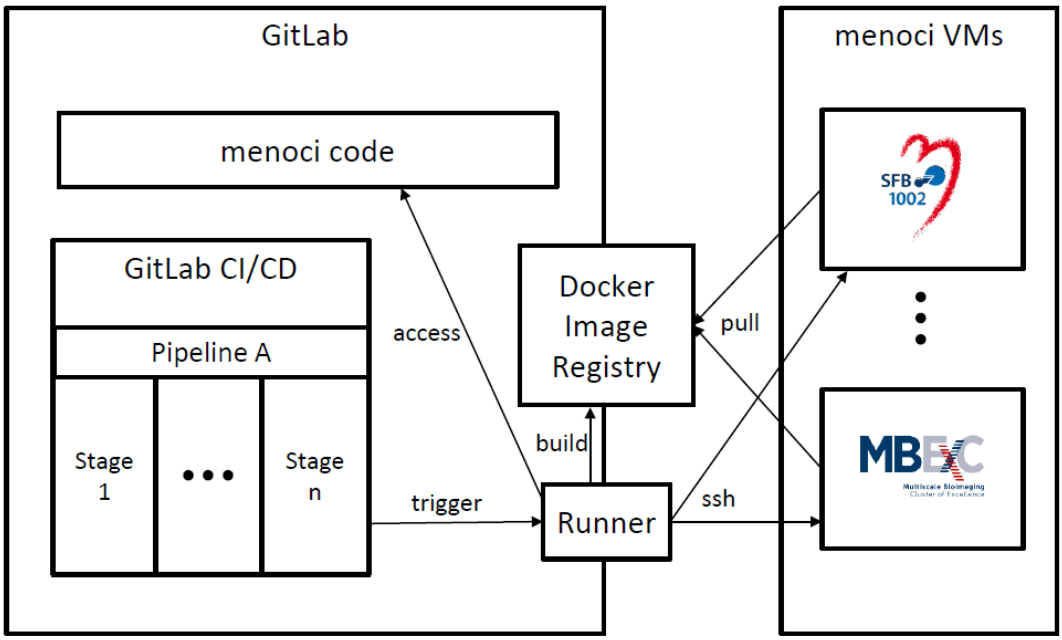

Figure 2: Schematic representation of the automation pipeline applied to multi-instance operation of *menoci*.

As sustainable server operation is technically highly complex and requires a broad range of IT resources, tools and expertise, we develop our server orchestration setup in close cooperation with the experts from our local consulting partners (eResearchAlliance (eRA)[17], Medical data integration center (MeDIC)[18]) and IT centers at Göttingen Campus[19] [20]. This includes technologies for storage, backups, virtualization, server certificates, applications and monitoring solutions.

4 Discussion

For several large research consortia at the Göttingen Campus, FAIRification of biomedical research data is approached by the application of the *menoci* web portal software stack. Individual in combination with overlapping functional requirements directed us to the initialization of multiple *menoci* instances to represent the research outputs of

[17] Online: eRA. https://www.eresearch.uni-goettingen.de/de/, accessed 2023-07-18.
[18] Online: MeDIC. https://medizininformatik.umg.eu/en/about-us/scientific-research-groups/medic/, accessed 2023-07-18
[19] Online: UMG – Informationstechnologie. https://www.umg.eu/ueber-uns/vorstand/ressort-wirtschaftsfuehrung-administration/informationstechnologie-g3-7/, accessed 2023-09-18.
[20] Online: GWDG. https://gwdg.de/, accessed 2023-09-18.

the consortia in an appropriate manner. A frequent request of the research consortia is to have a dedicated and separated web resource for their data handling. For this reason, we decided to maintain several VMs and menoci instances in parallel.

Here, we describe important measures to address challenges for multiple web service orchestration that have improved automation at its core.

As a consequence of wide-ranging automatic processes, the complexity of dependencies of technological components is increasing. This can lead to the increasing risk of small errors in the pipeline causing large effects for many or all jointly orchestrated instances. To minimize this risk potential, several validation and monitoring strategies are available such as automatic code testing, *Gitlab linters*[21] or server monitoring dashboard systems (e.g., OpenITCockpit[22] in our case).

The team responsible for development and operation of the instances needs efficient communication tools to act synergistically. An important basis is the comprehensive documentation of tasks and processes. In this context we started to integrate concepts of a variety of initiatives that are currently forming and developing to increase the standardization of software project documentation with the perspective of sustainable software development (e.g., FAIR Principles for Research Software (FAIR4RS Principles)[23]; Research Software Alliance[24] [25]; FAIR for Research Software (FAIR4RS) Working Group[26]; CodeMeta[27]; HERMES[28]).

We report a strong dependency on services and expertise provided by the local consulting partners and IT centers. A mutually beneficial interaction is that experiences are accumulated in coordination with the eRA. Therefore, our development approaches can serve as templates for other research institutions and comparable workflows in other application areas.

In close collaboration with the eRA we develop processes to cross-link and exchange our highly curated metadata sets with the recently implemented Campus wide services “Göttingen Research Online” (GRO)[29]: publication records for the represented

[21] Online: https://docs.gitlab.com/ee/ci/lint.html, accessed 2023-07-18.
[22] Online: https://openitcockpit.io/, accessed 2023-07-18.
[23] Chue Hong, NP, Katz DS, Barker M, Lamprecht AL, Martinez C, Psomopoulos FE, Harrow J, u. a. FAIR Principles for Research Software (FAIR4RS Principles). RDA recommendation. 2022. https://doi.org/10.15497/RDA00068.
[24] Online: https://www.researchsoft.org/, accessed 2023-07-18.
[25] Online: https://de-rse.org/, accessed 2023-07-18.
[26] Online: https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg, accessed 2023-07-18.
[27] Online: https://codemeta.github.io/, accessed 2023-07-18.
[28] Druskat S, Bertuch O, Juckeland G, Knodel O, Schlauch T. Software publications with rich metadata: state of the art, automated workflows and HERMES concept. arXiv. 2022. https://doi.org/10.48550/arXiv.2201.09015.
[29] Online: https://www.eresearch.uni-goettingen.de/services-and-software/goettingen-research-online/, accessed 2023-07-18.

research consortia are now also represented in GRO.publications, supplemental datasets and other data can be published and cited in GRO.data, and machines, software or services are represented and made bookable via GRO.instruments. Although development is at an early stage, these data driven service interactions are a key to transfer curated metadata from tools funded by restricted funding time periods to those intended to be permanently available.

For our use case scenarios, the development of enhanced automation processes has proven to be key to reduce resource efforts for technical administrative tasks. However, a variety of challenges could not yet be solved sufficiently. For example, IT personnel resources dedicated to server operation for research are largely limited in academia so that optimization of basic server managing tasks is time consuming and benefits are often merely visible for the target user groups. Here, a contract and close collaboration with an IT-Service provider to take over software provision, which is already in an operational state, could be supportive in the long run. Although the different *menoci* instances share a wide range of their components, each use case requires an adapted setup of modules as well as interfaces with additional digital infrastructures. This remains a challenge for the development and implementation of automation pipelines. Last but not least, version leaps of major IT components such as *Drupal* require large additional reprogramming workloads that are hardly compatible with available resources but are necessary to keep a software working over decades.

Data availability: All additional data and materials are available in the research data repository of Göttingen Campus[30] and via the project website[31].

Conflict of interest/contributions of authors and funding: All authors declare that no conflict of interest exists. LF, CH, LW, MS, RK, HK, US, SYN elaborated the concepts and wrote the manuscript. LF, CH, LW, MS, RK, HK, elaborated the concepts and developed the software. The human resources for conceptualization, development, and continuous improvement of the *menoci* platform were mainly funded by the German Research Foundation (DFG) through project funding of the CRC 1002 infrastructure (INF) project, as well as the Z projects of CRC 1190 and CRC 1565 and Germany's Excellence Strategy - EXC 2067/1- 390729940.

Acknowledgements: We sincerely thank Markus Suhr for his valuable preliminary work during his time with our research team. His contributions were instrumental in shaping the direction of this work.

[30] Online: Freckmann, Luca, 2023, *menoci* Software Source Code Distribution, Release 1.1. https://doi.org/10.25625/VA9UYR, accessed 2023-07-18.
[31] Online: menoci. https://menoci.io, accessed 2023-07-18.