MaRDIFlow: A CSE workflow framework for abstracting meta-data from FAIR computational experiments

Pavan L. Veluvali^{*} Jan Heiland^{*} Peter Benner^{*}

* Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg Email: {veluvali, heiland, benner}@mpi-magdeburg.mpg.de; ORCID: 0000-0001-8804-0338, ORCID: 0000-0003-0228-8522, ORCID: 0000-0003-3362-4103

Abstract: Numerical algorithms and computational tools are instrumental in navigating and addressing complex simulation and data processing tasks. The exponential growth of metadata and parameter-driven simulations has led to an increasing demand for automated workflows that can replicate computational experiments across platforms. In general, a computational workflow is defined as a sequential description for accomplishing a scientific objective, often described by tasks and their associated data dependencies. If characterized through input-output relation, workflow components can be structured to allow interchangeable utilization of individual tasks and their accompanying metadata. In the present work, we develop a novel computational framework, namely, MaRDIFlow, that focuses on the automation of abstracting meta-data embedded in an ontology of mathematical objects. This framework also effectively addresses the inherent execution and environmental dependencies by incorporating them into multi-layered descriptions. Additionally, we demonstrate a working prototype with example use cases and methodically integrate them into our workflow tool and data provenance framework. Furthermore, we show how to best apply the FAIR principles to computational workflows, such that abstracted components are Findable, Accessible, Interoperable, and Reusable in nature.

Keywords: Computational Workflows, Research Data Management, FAIR Research, Scientific Computing

Novelty statement: This manuscript presents a novel research data management tool, MaRDIFlow, that focuses on the automation of abstracting meta-data embedded in an ontology of mathematical objects. The working prototype of this software tool is illustrated through FAIR computational experiments.

1 Introduction

The interplay of data-intensive computational studies is a substantial part of scientific endeavors across all disciplines. Computational workflows have been used as a systematic way of describing the methods needed, the data involved, as well as computing resources and infrastructures. With ever more complex simulation models and ever larger primary data volumes, CSE (Computational Sciences and Engineering) workflow descriptions themselves have become an enabler for research beyond the execution of simulations, for example, to extract latent information from various data repositories and compare methodologies across diverse data and computational frameworks [AGMT17].

The FAIR principles [WDA⁺16], describe a set of requirements for data management and stewardship to ensure that the research data are Findable, Accessible, Interoperable, and Reusable. Each guiding principle is proposed to define the degree of 'FAIRness' via describing the distinct considerations for contemporary environments, tools, vocabularies and data infrastructures. While the elements of FAIR Principles are related and separable, they are equally applied to identify, describe, discover, and reuse meta-data assets of scholarly outputs. Overall, FAIR principles act as a guide to assist data stewards in evaluating their implementation choices. More recently, they have been adopted by funding agencies, such as the German Research Foundation [For22] for developing assessment metrics of research metadata across various disciplines [DHM⁺20].

While there is a knowledge base for CSE workflows from a (software) engineering point of view [HW09, BCG⁺19] and while it has been acknowledged that for documentation, model descriptions and code can complement each other [FHHS16], an inclusive abstract description of CSE workflows is not yet anchored. As for combining models, code, and data for the description of CSE simulations in a virtual lab notebook, Jupyter notebooks have gained popularity [KRKP⁺16a]. Also services like Code Ocean [CSFG19] target the combination of code and model descriptions. Still, little effort has been made to use abstraction for CSE workflow components in view of documentation tools that are generally applicable and that scale well with ever more demanding and sophisticated simulations. Lately, with the advancement of data intensive research, there has been a rise in the development of automated and reusable workflows, wherein these workflows aim to seamlessly integrate computer-based and laboratory computations through artificial intelligence [Nat22].

In this work, we analyze general and particular components and provide an abstract multi-layered description of CSE workflows: Each component will be characterized through an input/output description so that model, data, and code can be used interchangeably and, in the best case, redundantly. For that, we describe suitable meta data and a low level language for the descriptions of general CSE workflows [VHB23]. Additionally, we emphasize that the introduction of redundancy in the representation of models, code, and data serves as a positive feature for CSE workflows. This redundancy enhances the robustness of workflows via ensuring compatibility during potential execution issues. With interchangeable and multi-level components, workflows become more adaptable and reproducible, contributing to the overall reliability of a scientific task. Generally, we understand a CSE workflow as a chain of one or more interconnected models used for simulations. From existing literature, a CSE workflow is defined as a precise description of a multi-step procedure to coordinate multiple tasks and their meta-data dependencies [GCBSR+20]. In workflow systems, each task is represented through the execution of a computational process, such as, executing a code, calling a command line tool, accessing a database, submitting a job to a HPC cluster, or executing a data processing script.

In their general treatment, the following constraints need to be taken into account

- 1. In particular in a CSE context, each model might be arbitrarily complex and computationally demanding.
- 2. Often, the particular numerical realization represents a compromise between accuracy and computational costs.
- 3. Within a workflow, models are likely implemented in different frameworks or languages.
- 4. In the case of, say, commercial codes that may well be one part of a workflow, some simulation models might not be fully available but only evaluable through interfaces.
- 5. Possibly, the actual simulation code is not available at all but only descriptions and, in the better case, alternative implementations.

Nevertheless, the goal of any CSE workflow framework is to offer a specialized programming environment that minimizes the efforts required by scientists or researchers to perform a computational experiment [VHB23]. In general, CSE workflow description can be categorized into distinct parts or phases, as listed below, governing its functional operation.

- Composition and abstraction
- Execution

• Meta-data mapping and provenance

Firstly, during composition and abstraction, a CSE workflow is created either from scratch or from modifying a previously designed workflow, whereby the user relies on different workflow components and data catalogs. Some of the well-known methods for editing and composing workflows are either textual or graphical or mechanism-based semantic models [DGST09]. The workflow then abstracts software components written by third parties, and handles heterogeneity via shielding run time incompatibility and complexities. Secondly, during execution, the workflow components are executed either by a computational engine or via a subsystem, wherein a static or an adaptive model is implemented to realize the meta-data. Importantly, repetitive and reproducible pipelines in order to manage the control and flow of a simulation are an important aspect of the second phase. Once the workflow is well defined, all, or portions of the workflow are sent for mapping. Finally, the data and all associated metadata and provenance information are recorded and placed in user-defined registries which are then accessed to design a new workflow description. Through the following stages, CSE workflow description act as modular building blocks with standardized interfaces, and are generally linked and run together by a computational framework.

The present article is organized as follows: in the following section, we discuss the current state of the art in Jupyter notebooks and computational workflows. Afterwards, we present our research data management tool, namely, MaRDIFlow, wherein the framework and its usage as a command-line tool is discussed in detail. Next, we discuss our RDM tool via minimum working examples. Lastly, we put forward the conclusions and future direction from the present work.

Existing Solutions

Reproducibility poses a multifaceted challenge which often demands a comprehensive examination. In this work, we focus on the specific aspect of ensuring reproducibility of computational workflows through integrated software solutions.

Because of the popularity and universality of Jupyter notebooks, we begin with highlighting the capabilities of the Jupyter environment, which significantly boost productivity in computational science and mathematics, while also promoting reproducibility [BTK⁺21]. In general, Jupyter Notebooks [KRKP⁺16b] are accessed through a modern web browser and are typically designed to support interactive exploration and publishing records of a scientific computation. Through text and code blocks, it performs a specific computation and elucidates it in detail. The code within a given Jupyter Notebook is organized into cells, which in turn allows individuals to modify and execute, respectively. Also, the output from each cell appears directly below it and is stored as part of the document. This approach often facilitates a symbiotic display of code, data, and model descriptions and naturally ensures replicability of the experiments; [FHHS16]. In this respect, the side by side appearance of text (for documentation) and code (for the execution) is in line with the redundant or multi-layered representation of workflows that we want to achieve but adds to the complexity of such a notebook realization.

What concerns reproducibility, in the strict sense that an experiment could be reproduced solely by information provided in the notebook, certain design decisions in the Jupyter notebook like undocumented versions of the imported libraries let alone the underlying libraries in the backend may stand in the way. From a more practical perspective, by its linear design of consecutive cells, Jupyter notebooks are not well suited to handle larger projects. And although a call of third party code is certainly possible through direct Julia/python/R interfaces or through the system's shell, the embedding of external tools is not a primary and, thus, not a well-defined feature of Jupyter notebooks. Finally, the reproducibility of a workflow in a Jupyter notebook hinges on the code only whereas the description is commonly seen as an add-on. Thus, there is no built-in mechanism that ensures completeness of the documentation to function as an equivalent or fully valid addition to the code base. In fact, recent studies have found that a Jupyter notebook per se is not a strong guarantee for reproducibility; see Refs. [SM24, PMBF21].

Whereas Jupyter notebooks have become a popular tool for implementing, documenting, and publishing workflows and experiments of moderate complexity, in computational science and engineering, multiple advanced tools have addressed the challenge of managing workflows composed of various simulation codes. Domain specific workflow managers, such as CWL [CAI+22a] and

Galaxy [Com22] are applicable to high-performance computing in general. CWL [CAI⁺22a] is widely known for the description of command-line tools and of the workflows made from these tools. It includes many features, such as, software containers, resource requirements, workflow-level conditional branching. While most of the CWL development began in the field of bioinformatics, since 2016, the CWL standards have been used in other fields, including hydrology, radio astronomy, geo-spatial analysis, and high-energy physics. On the other hand, Galaxy [Com22] serves as an accessible browser-based platform for scientific computing. It facilitates data sharing, analysis, and visualization for scientists with minimal technical barriers. In addition, integration with thirdparty tools like *noWorkflow* [PMBF17] allows for effective tracking of provenance, elucidating the relationship between inputs, code, and generated files.

A container-based approach to modelling workflow components has been followed in the *func*tional mockup interface (FMI) [BOA⁺11] development. Here, arbitrary simulation model are encapsulated in a complete virtual computational environment (a container) and made accessible through interfaces. This facilitates an easy exchange of simulation tools (even without disclosing the source) and can be integrated in workflow designs, as it is specified in the FMI using *xml* syntax.

Most of the aforementioned systems have improved the reproducibility of computational workflows over the last years, and have become the defacto standard for syntactic interoperability of workflow management systems. However, each of the systems discussed come with their own limitations. Namely, CWL often syntax fails to address the user-defined construction and interaction with other command-line tools once its execution is finished [CAI+22b]. Likewise, one of the disadvantages of document based workflow definitions is their static nature, as the exact flow of the workflow must be known before execution. This specific mechanism inherently imposes constraints on programming structures that can be utilized, generally via confining options to either directed acyclic graphs (DAGs) or directed cyclic graphs (DCGs), especially in cases where loops are accommodated by the markup language [UHY⁺21]. Nevertheless, all the aforementioned frameworks store the data produced, but none with a focus on explicitly recording provenance in detail, and, in particular, the workflow components are not expressed in a multi-level framework or abstract objects. On the other hand, customizing the system for specific requirements may present a significant challenge for end users, requiring additional effort and resources.

Container-based implementations like in FMI can mitigate the issues with data provenance or setting up the environment as much as mere replication is concerned. However, in view reproducibility in different environments or reusability and adaptation of workflow components, these containers would need to be equipped with just the same meta-data as any other workflow design tool would require it.

Based on the aforementioned challenges, it becomes evident that the existing research landscape for a comprehensive workflow description tailored to effectively handle mathematical data within a multi-component framework is lacking. On that account, the present study aims to bridge the existing gap.

2 MaRDIFlow

The design principle of MaRDIFlow is to handle the components as abstract objects described by their input to output behavior and metadata. By means of the metadata and by matching the I/O interfaces, the objects can be chained together to form a workflow; see Fig. 1.

Each item then can have different descriptions or realizations that are, in the best case, equivalent and redundant; see Fig. 2 for an example of such a vertical dimension in the workflow. We note that the redundancy is meant from a theoretical perspective. Practically, the different realizations or representations can be used in different scenarios like compiling a mathematical description of the workflow or using lookup tables as a shortcut during a simulation.

Additionally, this multi-level description enhances or even enables reproducibility in scenarios where, for example, certain software components of a workflow are not available but will be replaced by data or a description and vice versa.

• Defined by metadata



- Figure 1: Generic chain of models that describe the workflow in the simulation of transformer noise generation and its realization on different levels of abstraction.
 - Realized as Fig. 1 and Fig. 2

The working prototype of MaRDIFlow is designed as a command line tool that enables researchers and users to execute, document and maintain the provenance for reproduction and replication of computer-based experiments. As shown in Fig. 3 via a screenshot, MaRDIFlow --help option provides a help message with an immediate sense of what MaRDIFlow is. Through the following list we discuss some of the important arguments required to configure and perform the most common tasks with our RDM tool.

- --workflow-title provides the working title for the workflow, as default we have, workflow_title = This is a CSE workflow description under MaRDIFlow.
- --input requires an inputs object file in .json format such that it consists of all the numerical parameters for the workflow. An example inputs object file is shown in Fig. 4, where the key value pairs are a valid string representing the parameter name, and the values found to the right side of the colon are the absolute values for the given input parameter. As shown in Fig. 4, the JSON object represents the simulation and material parameters which are then passed on to the required workflow component.
- --output-directory argument allows the user to specify the desired output directory. If not provided, then the workflow output is collected in the root working directory, namely Output.
- MaRDIFlow can also be configured using a .ini config parser file, which includes both the required default and user-defined sections. It also acts as a list of parameters that governs how the tool is run and configured.
- --config example_config_file.ini will execute MaRDIFlow through terminal. An example configparser file is showcased via a screenshot in Fig. 5.
- --component argument is necessary to execute the desired workflow component in the multilevel framework, as previously discussed. In present version, users can choose between --math-data or --math-solver components for executing a given I/O data or a numerical model, respectively.
- --display represents the descriptive part of our RDM tool. Herein, --display_html=TRUE or --display_pdf=TRUE will convert the .md into desired format. An empty string will parse



Figure 2: An exemplified vertical multi-level dimension of a MaRDIflow component: equivalent and preferably redundant descriptions of a workflow unit.

	veluvali@pc1432:/scrat usage: MaRDIFlow [-h] [ou [di [ou	ch/veluvalt/MaRDI/mardtflow\$./MaRDIFlowhelp [workflow-title] [config] [input] [output-directory] tput-file] [component] [math-solver] [math-data] splay] [display-html] [display-pdf] [inputmarkdown] tputmarkdown] [get-data] [display-data] [get-url-data]
MaRDIFlow: A computational framework for abstraction of multi-layered CSE workflow components		
ontional arguments. T		
	b bolo	chow this holp mossage and exit
	-n,netp	Drowide working title for the CCE workflow description here
	workitow-title	Provide the absolute filenath of config(ini) or
	connitg	config(conf) filo
	input	Dravide the shealute file path for input parameters in
	chpuc	(icon) format
	output directory	Drowide the abcelute math to collect output from workflow
	output-utrectory	components. If not provided, default directory: ./Output
	output-file	Provide the absolute path for the output file, default
		directory: ./Output
	component	Specify workflow component for a multi-level description
	math-solver	Choose of the solver from the listed strings
	math-data	Argument to pass I/O data in excel file (.xlsx) format
	display	This flag initiates the descriptive part of MaRDIFlow tool
	display-html	Converts the markdown file and workflow describes into html-
		format
	display-pdf	Converts the markdown file and workflow describes into pdf-
		format
	inputmarkdown	Provide the input markdown file directory for CSE workflow
		description
	outputmarkdown	Provide the output display file directory for workflow
		description in the desired .html or .pdf format
	get-data	Provide the input filepath for I/O data workflow description
	display-data	Displays an I/O data from a given dataset in excel file
		(.xlsx) format
	get-url-data	Calls an I/O dataset from a given url with DOI

Figure 3: A screenshot illustrating the help message of our RDM tool, MaRDIFlow.

FALSE bool value. The file path for workflow description in .md file format is passed on to the workflow tool via --inputmarkdown flag. For this flag, it is important that the absolute file path for input as well as for output is provided.

• --data configures the second component of the workflow description for a given I/O data. By utilizing --get-data and --get-url-data flags, users can furnish a workflow description for a lookup table or a database, thereby presenting an alternative approach to a numerical model.

Overall, the working prototype of MaRDIFlow ideally provides the user with a specification de-

```
{
         "nx"
                 : 128,
         "ny"
                 : 128,
         "dx"
                 : 2.0e-9
         "dy"
                   2.0e-9,
         "sigma" : 0.2,
         "R"
                 : 8.314.
         "temp"
                 : 673.
         "nsteps": 6000,
         "c0"
                 : 0.5,
         "ac"
                 : 3.0e-14,
         "La"
                 : 13460e-6
}
```

Figure 4: Example inputs object JSON file format with a set of static parameters required for a specific workflow component.



Figure 5: Screenshot of an example configparser .ini file to initialize and run MaRDIFlow through command-line.

scription, that elucidates the meta-data for a given use case. In addition, it also consists of a computational part that acts as a mechanism to call and execute the given meta-data. In order to further understand our RDM tool, the following minimum working examples implemented in MaRDIFlow are discussed below.

Minimum working examples

In the present section, we present some use cases as minimum working examples to illustrate in detail the working prototype of MaRDIFlow.

Methanization Reactor

As a working example within the MaRDIFlow framework, a forward solution that converts CO_2 to CH_4 as a result of methanization [BHBS21] is illustrated. In general, reactor models are crucial for converting renewable electricity into chemical energy carriers, specifically through carbon dioxide methanation [BHBS21]. In this study, a reactor model was examined through a set of nonlinear partial differential equations (PDEs) for mass and energy balances. The schematic representation of the workflow is provided in Fig. 6, and we write the governing PDEs as



Figure 6: A screenshot illustrating the workflow description of a methanization reactor model using the MaRDIFlow tool. As an output, screenshots of the console and trajectory of the changing reactor load are shown here from the workflow. In addition, as a final output, the descriptive part of the tool shows the necessary meta-data to reproduce the use case.

$$\epsilon \rho_{gas} \frac{\partial w_{\alpha}}{\partial t} = -\rho_{gas} \mathbf{v} \,\nabla \cdot w_{\alpha} - \nabla \cdot \mathbf{j}_{\alpha} + (1-\epsilon) \nu_{\alpha} \widetilde{r} \tag{1}$$

$$(\rho c_p)_{eff} \frac{\partial T}{\partial t} = -(\rho c_p)_{gas} \mathbf{v} \cdot \nabla T - \nabla \cdot \dot{\mathbf{q}} + (1 - \epsilon) \Delta_R \tilde{H} \tilde{r}$$
(2)

In the above set of equations, w_{α} is the component of mass fraction for $\alpha \in CO_2$, H₂CH₄, H₂, O, **v** is the superficial gas velocity, ρ is the gas density, \mathbf{j}_{α} are dispersive component fluxes, ν_{α} is stoichiometric coefficients, \tilde{r} is the reaction rate of methanation, c_p is the specific heat, T is the temperature, $\dot{\mathbf{q}}$ and $\Delta_R \tilde{H}$ is the heat reaction. Furthermore, this study revolves around to ensure that the reactor operates at maximum conversion rates, even when subjected to varying loads. To address this specific objective, an optimal control problem (OCP) is defined as follows:

$$\zeta(u) = \frac{1}{t_e} \int_0^{t_e} X_{CO_2}(t) \, dt \tag{3}$$

Here, we express the reactor model as a system of ordinary differential equations (ODEs), obtained through the finite volume method applied to the PDE system. Additionally, on states and control we incorporate inequality constraints. Further initialization and model specific details can be found in Ref. [BHBS21]. Workflow description for the methanization reactor model is defined as given below:

- Initialize the system with the given set of governing equations
- Perform a forward simulation with temperature as the input parameter
- Calculate the conversion rates via calculating the change in CO₂ mass fraction with time via post-processing

Spinodal decomposition in a binary A-B alloy

As a second example, let us consider a two-dimensional simulation of the Cahn-Hilliard equation [CH58] for an A-B alloy. During spinodal decomposition, when a homogeneous binary alloy is rapidly cooled from a given temperature, the resulting domain consists of a fine-grained structure of two phases, and over time, the fine-grained structure coarsens at the expense of smaller particles. The development of a fine-grained structure from a homogeneous state is referred to as spinodal decomposition, while the coarsening mechanism is often defined as Ostwald ripening.



Figure 7: A screenshot illustrating the workflow description of a Cahn-Hilliard Model using the MaRDIFlow tool. Screenshots of the console and simulation images as the outputs from the workflow are shown here. In addition, as a final output, the descriptive part of the tool shows the necessary meta-data to reproduce the use case.

The schematic representation of the workflow is provided in Fig. 7, and the set of governing equations required to simulate the phase-separation behavior between A-B alloy is given below. At first, we define an order parameter c as the concentration of B atom, and the bulk free energy of the system is defined by

$$G = \int_{v} (g_{chem}(c) + g_{grad}(\nabla c)) dV$$
(4)

where g_{chem} and g_{grad} are the chemical free energy and the gradient energy densities, respectively. In this study, the chemical free energy density is formulated as:

$$g_{chem} = RT[c\ln c + 1 - c\ln(1 - c)] + Lc(1 - c)$$
(5)

where L is the atomic interaction parameter, and the gradient energy density is given as:

$$g_{grad} = \frac{a_c}{2} |\nabla c|^2 \tag{6}$$

where a_c is the gradient energy coefficient. Considering the total free energy of the system decreases with time, the temporal evolution of the order parameter c is given by:

$$\frac{\partial c}{\partial t} = \nabla \cdot \left(M_c \nabla \frac{\delta G}{\delta c} \right) = \nabla \cdot \left(M_c \nabla \mu \right) \tag{7}$$

where μ is the diffusion potential of B atom. According to classical thermodynamics, μ is generally expressed as:

$$\mu = \frac{\delta G}{\delta c} = RT[\ln c - \ln(1-c)] + L(1-2c) - a_c \nabla^2 c$$
(8)

From 7, M_c is the diffusion mobility of B atom, given by:

$$M_{c} = \left[\frac{D_{A}}{RT}c + \frac{D_{B}}{RT}(1-c)\right]c(1-c) = \frac{D_{A}}{RT}\left[c + \frac{D_{B}}{D_{A}}(1-c)\right]c(1-c)$$
(9)

In the above equation, the parameters D_A and D_B are the diffusion coefficients of the respective A and B atoms in the system. Lastly, herein, the Cahn-Hilliard equation is discretized by simple finite difference method, 1st-order Euler method is used for time-integration, and for spatial derivatives the 2nd-order central finite difference method is implemented. The workflow for the present use-case is carried out as given below:

- Initialize the bulk free energy and initial local concentration through an inputs object JSON file.
- The initial configuration of the simulation domain as shown in Fig. 7.
- Pass the required simulation parameters to the workflow component.
- Time evolution of local concentration as well as the phase-separation process is captured as an output through simulation images.
- Alongside, concentration for various timesteps is collected as an output as well.

The above workflow can be performed by using MaRDIFlow --config config_CH_2D.ini in the root directory terminal, and the resulting output shall be displayed on the screen, similar to Fig. 7. At the end of the workflow, the phase-separated simulation screenshots along with the corresponding equilibrium concentration are collected in the user-defined output directory.

Summary and Outlook

The practice to perform data and software intensive tasks has been taken hold by computational workflows. Subsequently, the rapid growth in their uptake and application on computer-based experiments presents a crucial opportunity to advance the development of reproducible scientific softwares. As a part of the MaRDI consortium [MaR21] on research data management in mathematical sciences, in this work, we presented a novel computational workflow framework, namely, MaRDIFlow, a prototype that focuses on the automation of abstracting meta-data embedded in an ontology of mathematical objects while negating the underlying execution and environment dependencies into multi-layered vertical descriptions. Additionally, the different components are characterized by their input and output relation such that they can be used interchangeably and in most cases redundantly.

The design specification as well as the working prototype of our RDM tool was presented through different use cases. In the present version, MaRDIFlow acts a command-line tool such that it enables users to handle the workflow components as abstract objects described by input to output behavior. At its core, MaRDIFlow ensures that the output generated is detailed, and a comprehensive description facilitates the reproduction of computational experiments. At first we illustrated the conversion rates of CO2 using a methanization reactor model, and later, we demonstrated the two-dimensional spinodal decomposition of a virtual A-B alloy using the Cahn-Hilliard model. Our RDM tool adheres to FAIR principles, such that the abstracted workflow components are Findable, Accessible, Interoprable and Reusable. Overall, the ongoing development of MaRDIFlow aims at covering heterogeneous use cases and act as a scientific tool in the field of mathematical sciences.

Apart from this, we are also working towards developing an Electronic Lab Notebook (ELN) in order to visualize as well as execute the MaRDIFlow tool. The ELN will provide researchers with a user-friendly interface to interact with the tool efficiently and seamlessly. Lastly, although the present manuscript introduces our RDM tool as a working proof of concept, we plan to publish a detailed manuscript with technical details and use cases in the near future.

Acknowledgments

The authors are supported by *NFDI4Cat* and *MaRDI*, funded by the Deutsche Forschungsgemeinschaft (DFG), project 441926934 "NFDI4Cat – NFDI für Wissenschaften mit Bezug zur Katalyse" and project 460135501 "MaRDI – Mathematische Forschungsdateninitiative".

Data Availability

Results presented in this work are apart of an ongoing investigation, however a working prototype with the second use-case is available and documented at https://doi.org/10.5281/zenodo.10608764

References

- [AGMT17] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor. Scientific workflows: Past, present and future, 2017.
- [BCG⁺19] A. Brinckman, K. Chard, N. Gaffney, M. Hategan, M. B. Jones, K. Kowalik, S. Kulasekaran, B. Ludäscher, B. D. Mecum, J. Nabrzyski, V. Stodden, I. J. Taylor, M. J. Turk, and K. Turner. Computing environments for reproducibility: Capturing the "whole tale". *Future Generation Computer Systems*, 94:854–867, 2019.
- [BHBS21] J. Bremer, J. Heiland, P. Benner, and K. Sundmacher. Non-intrusive time-pod for optimal control of a fixed-bed reactor for co2 methanation. *IFAC-PapersOnLine*, 54(3):122–127, 2021.
- [BOA⁺11] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, et al. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th international Modelica conference*, pages 105–114. Linköping University Press, 2011.
- [BTK⁺21] M. Beg, J. Taka, T. Kluyver, A. Konovalov, M. Ragan-Kelley, NM. Thiéry, and H. Fangohr. Using jupyter for reproducible scientific workflows. *Computing in Sci*ence & Engineering, 23(2):36–46, 2021.
- [CAI+22a] M. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Ménager, S. Soiland-Reyes, B. Gavrilović, C. Goble, et al. Methods included: Standardizing computational reuse and portability with the common workflow language. *Communications of the ACM*, 65(6):54–63, 2022.
- [CAI⁺22b] MR. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Ménager, S. Soiland-Reyes, B. Gavrilović, C. Goble, et al. Methods included: Standardizing computational reuse and portability with the common workflow language. *Communications of the ACM*, 65(6):54–63, 2022.
- [CH58] JW. Cahn and JE. Hilliard. Free energy of a nonuniform system. i. interfacial free energy. *The Journal of chemical physics*, 28(2):258–267, 1958.
- [Com22] The Galaxy Community. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. *Nucleic Acids Research*, 50(W1):W345–W351, 04 2022.
- [CSFG19] A. Clyburne-Sherin, X. Fei, and SA. Green. Computational reproducibility via containers in social psychology. *Meta-Psychology*, 3, 2019.
- [DGST09] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer* Systems, 25(5):528–540, 2009.

- [DHM⁺20] A. Devaraju, R. Huber, M. Mokrane, P. Herterich, L. Cepinskas, J. de Vries, H. L'Hours, J. Davidson, and Angus W. Fairsfair data object assessment metrics, October 2020.
- [FHHS16] J. Fehr, H. Heiland, C. Himpe, and J. Saak. Best practices for replicability, reproducibility and reusability of computer-based experiments exemplified by model reduction software. AIMS Mathematics, 1(3):261–281, 2016.
- [For22] Deutsche Forschungsgemeinschaft. Guidelines for Safeguarding Good Research Practice. Code of Conduct, April 2022. Available in German and in English.
- [GCBSR⁺20] C. Goble, S. Cohen-Boulakia, S. Soiland-Reyes, D. Garijo, Y. Gil, MR. Crusoe, K. Peters, and D. Schober. Fair computational workflows. *Data Intelligence*, 2(1-2):108–121, 2020.
- [HW09] M. A. Heroux and J. M. Willenbring. Barely sufficient software engineering: 10 practices to improve your cse software. In Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, pages 15–21, 2009.
- [KRKP⁺16a] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter notebooks - a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, 2016.
- [KRKP⁺16b] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Jonathan Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Damián Avila, S. Abdalla, and C. Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.
- [MaR21] MaRDI. Mathematic research data initiative, 2021. URL: https://www.mardi4nfdi.de.
- [Nat22] National Academies of Sciences, Engineering, and Medicine. Automated Research Workflows for Accelerated Discovery: Closing the Knowledge Discovery Loop. The National Academies Press, Washington, DC, 2022.
- [PMBF17] JF. Pimentel, L. Murta, V. Braganholo, and J. Freire. noworkflow: a tool for collecting, analyzing, and managing provenance from python scripts. *Proceedings* of the VLDB Endowment, 10(12), 2017.
- [PMBF21] JF. Pimentel, L. Murta, V. Braganholo, and J. Freire. Understanding and improving the quality and reproducibility of Jupyter notebooks. *Empirical Software Engineer*ing, 26(4):65, 2021.
- [SM24] S. Samuel and D. Mietchen. Computational reproducibility of Jupyter notebooks from biomedical publications. *GigaScience*, 13:giad113, 01 2024.
- [UHY⁺21] M. Uhrin, SP. Huber, J. Yu, N. Marzari, and G. Pizzi. Workflows in aiida: Engineering a high-throughput, event-based engine for robust and modular computational workflows. *Computational Materials Science*, 187:110086, 2021.
- [VHB23] PL. Veluvali, J. Heiland, and P. Benner. Mardiflow: A workflow framework for documentation and integration of fair computational experiments. In *Proceedings of* the Conference on Research Data Infrastructure, volume 1, 2023.

[WDA⁺16] MD. Wilkinson, M. Dumontier, IJ. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, JW. Boiten, LB. da Silva Santos, PE. Bourne, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.