Model coupling through reproducible adapter workflows based on shared transformation functions

Patrick Kuckertz[1,4,*], Benjamin Fuchs[2], Julian Schönau[1], Hedda Gardian[2], Kevin Knosala[1], Eugenio Arellano Ruiz[2], Jan Göpfert[1,3], Hans Christian Gils[2], Jann M. Weinand[1], Patrick Jochem[2], Jochen Linßen[1], and Detlef Stolten[1,3]

[1]Forschungszentrum Jülich GmbH, Institute of Energy and Climate Research – Jülich Systems Analysis, 52425 Jülich, Germany
[2]German Aerospace Center (DLR), Institute of Networked Energy Systems, 70563 Stuttgart, Germany
[3]RWTH Aachen University, Chair for Fuel Cells, Faculty of Mechanical Engineering, 52062 Aachen, Germany
[4]Lead contact
[*]Correspondence: p.kuckertz@fz-juelich.de

Summary

In computational science, coupling research software and data into workflows is essential for addressing complex research questions and ensuring reproducibility. However, current metadata schemas rarely provide sufficient information about data models in software interfaces and datasets, hindering their effective integration into workflows. Additionally, most workflow management tools are not designed to handle metadata for process reproduction. This article refines the DataDesc metadata schema to annotate data models not only in software interfaces but also in datasets, and presents an extension to the DataDesc framework for automatically comparing data models and identifying transformation requirements. The ioProc workflow manager is introduced to bundle shared transformation functions into adapter workflows and ensure transparent process documentation. Two use cases from energy systems analysis demonstrate the workflow design and implementation approaches. Overall, this article bridges the gap between abstract guidelines, such as the FAIR principles, and researchers' daily data and software driven analyses, promoting reusability, reproducibility and transparency in science.

Keywords

Software Coupling, Data Integration, Scientific Workflows, Data Models, Transformation Functions, Metadata, Reusability, Interoperability, DataDesc Schema, ioProc Workflow Manager

Introduction

Motivation

In computational sciences and especially in energy systems analysis, the coupling of research software into comprehensive software workflows is a common approach for incorporating multiple perspectives into a scientific analysis and to investigate complex research questions.[1] In a workflow context, the sustainable handling of resources is particularly important in two aspects. On the one hand, these workflows represent in themselves flexible research tools. Reuse or adaption of these workflows to changes in research questions in a fast, reliable and low-effort approach can avoid considerable redundant development, implementation, and maintenance

efforts but requires careful upfront design and additional effort.[2] On the other hand, these workflows play a central role in scientific exchange, as they contribute to the transparency and reliability of scientific conclusions by enabling others to trace, interpret and reproduce the results of these workflows.[3,4] Workflow design therefore has the potential to positively increase the overall efficiency of a research community.

Usually, however, scientists are confronted with major challenges when creating and applying workflows that should be adaptable, reuseable and reproducible by others. Especially missing or difficult-to-process information about software and data resources represents a major obstacle. Particularly in a research field such as energy systems analysis, in which no uniform interface and data standards have been established and which is characterized by heterogeneous formats originating from different research disciplines, the provisioning of metadata which includes information about individually used data models is imperative.[5] Another hindrance is the lack of computer-aided workflow design and building tools that cover and support both research software and data. In an environment full of initially incompatible data models, there is a strong need for reliable and robust data transformation and processing capabilities, in order to combine input data and software models into a structured process. As long as there are no adequate tools supporting scientists in workflow integration or offering automation capabilities, data models must be manually researched and compared, which is a tedious and time-consuming task. In addition, transformation requirements must be identified manually and suitable transformation functions researched in a time-consuming process. A third challenge lies in the lack of traceability and reproducibility of workflows and workflow results. Even as there is no lack of workflow tools with which researchers can implement their model chains (see for example Kieser et al.[6] or Mölder et al.[7]), the metadata accompanying these processes, which is relevant for their understanding and interpretation, must be recorded, linked and stored separately, usually in a manual process. This additional effort, which has to be made for each individual model coupling, data processing or transformation, is often not done consistently. This situation contributes to the current reproducibility crisis in science.[8–10] An illustrative example is a study of Jupyter Notebooks, a common form of documenting scientific workflows and analysis, for their reproducibility. The study of Pimentel et al.[11] showed issues with dependency documentation and execution of the notebooks and thus a lack of reproducibility, transparency and adaptability.

Confronted with this situation, a cultural change is slowly taking place in energy systems analysis and other science communities, which places increasing importance on overcoming the associated challenges. In particular, the objectives formulated in the FAIR principles, findability, accessibility, interoperability and reusability, are effectively promoting the adoption of metadata standards as this directly contributes to all four principles.[12] At the same time, the joint development and use of domain ontologies for the uniform description of software, data, and other research data artifacts, such as the Open Energy Ontology[13], is also gaining traction. As the progress of adopting the FAIR principles continues, numerous problems and uncertainties in the practical implementation of these guidelines are experienced in everyday science. To solve these, novel approaches and solutions are required with renewed urgency. As described by Leipzig et al.[14] there exists an explicit need for action in the area of metadata description of data that is to be used as input by software models. The authors call for a common metadata standard that enables the description of file formats and data structures at workflow level and their automated comparison in order to link individual elements of analytical processes. They claim that metadata tools are as important for practical and computational research as the software and data itself. Leipzig et al.[14] further show that despite the existence of numerous workflow tools, collecting and analyzing provenance, i.e. recording all activities that lead to the creation of a data object, is still a key challenge for workflow design.

The approaches presented in this article provide an answer to the question how metadata aware workflows, which cover software and data, can be implemented and documented to make

a significant step towards the FAIR principles. In the *Related Work* section, current metadata schemas are first compared and examined with regard to their suitability for describing data models. Subsequently, an introduction to data comparison and workflow tools is given, focusing in particular on their capabilities for metadata processing in a scientific context. The *Results* section shows the extent to which the metadata elements of the DataDesc schema, which is designed to describe data models in software interfaces, are also suitable for describing data models in data files, and which structural refinements have been applied to facilitate this application case. Furthermore, an extension of the DataDesc framework for the automated comparison of metadata is presented, in which data models are compared and any discrepancies are identified as transformation needs, whose machine-actionable description can be used in the future for the automated identification and reuse of data transformation methods in modular workflow designs. In a second step, the ioProc workflow manager[15] is presented, with which data transformation steps can be combined into so-called adapters as needed and sustainably reused in workflows. The individual process steps are also automatically documented as metadata. Finally, the *Results* section presents two use cases from the context of energy systems analysis to demonstrate how DataDesc and ioProc can be used to design and implement transparent, reproducible and well-documented scientific workflows. The *Discussion* section, which emphasizes the main features of the approaches presented as well as their limitations, provides an insight into future work and closes this article.

This work hence presents an adaptable and extensible approach for an improved design and reusability of scientific workflows incorporating software and data, rooted in and exploiting the benefits of metadata. The presented approach contributes to the FAIR principles and supports the transparency and reproducibility of workflows in a scientific community by adding automatic metadata generation.

Related Work

This section examines current metadata schemas in regards to their suitability for describing data models. Furthermore, different approaches for comparing data are contrasted. Finally, the role of workflow tools for computational sciences is established.

Metadata Schemas

A metadata schema standardizes the description of artifacts within its scope by defining a set of metadata elements to be used.[16] A metadata standard may further standardize the encoding format, allowed values, their representation, and so forth.[16] Application profiles adapt subsets of one or multiple metadata schemas to tailor these to specific applications. They may further define custom elements, rules, best practices, etc. Many schemas, standards and application profiles exist with different scopes. The scope refers to specific use cases that are supported, but may be limited to, for example, a particular artifact type (such as images, software, datasets, etc.), scientific domain, or country.

Domain-agnostic standards that propose sets of elements to describe datasets include Dublin Core[17], DataCite[18] that provides a mapping to Dublin Core, DCAT[19] that incorporates terms from Dublin Core as well as other controlled vocabularies, and schema.org's Dataset class[20] that in turn is based on DCAT. None of these describe data models in detail. In addition, the general metadata of a dataset such as *authors*, *titles*, *descriptions*, the content and how to access it is only defined in an abstract way using terms such as *encoding*, *file format*, *has part*, *about*, *variable measured*, *temporal or spatial resolution and coverage*. Dublin Core and DCAT allow to

reference a schema or standard using the property *conforms to*. However, there is no schema describing the data model provided.

RO-Crate[21] and Data Package[22] are approaches to bundling data and their metadata. For example, Data Package allows the specification of table schemas for individual files. OEMetadata[23] is a domain-specific standard that builds on Data Package and encourages the referencing of concepts from controlled vocabularies and ontologies.

Representing n-dimensional datasets in the Resource Description Framework (RDF), the RDF Data Cube Vocabulary[24] is not limited to tabular data. It represents the actual data itself, but uses the rich semantics of RDF to describe information that would normally be part of metadata. D-REPR[25] and the Software Description Ontology[26] reuse the RDF Data Cube Vocabulary to define the structure of a dataset. Other self-describing data formats include HDF5[27] and NetCDF[28].

More recently, Croissant[29] and DataDesc[30] proposed approaches that are independent of the dataset format, allowing for the referencing of terms from controlled vocabularies, and the description of multidimensional as well as nested data structures. To increase the reusability of datasets within the machine learning community, Croissant supports the specification of data types and foreign key references. In addition, usage information, e.g. where the training, validation and test split is located, which columns are to be extracted and how the values are to be parsed, facilitates the training of machine learning models. DataDesc is a domain-agnostic standard for describing software interfaces and their data models with metadata to facilitate model coupling, workflow composition, and software discovery as well as integration in general. It builds on the OpenAPI specification, a widely used standard to document APIs, and provides a mapping to schema.org. DataDesc offers the possibility to describe multi-dimensional and nested function parameter structures and value ranges in runtime representations. Semantic concepts of variables as well as their units or quantity types can be described in a machine-actionable way by referring to terms from controlled vocabularies.

In its area of application for interface documentation, the DataDesc schema offers the highest level of detail and the greatest flexibility in the description of data models, so that it can map variables and function parameters of any complexity. In addition, it integrates and references current metadata standards and can be used across domains and applications. However, there is currently no metadata schema that has been designed to map data models of datasets at a comparable level of detail.

Data Comparison Tools

To what extent and in what respect data is compared depends on the use case. We focus on the comparison of data models between datasets and software or service interfaces with the aim of integrating them into comprehensive software workflows. This requires consideration of both semantics and technical representations, including data structures, data formats, and value range specifications.

To compare different versions of files, utilities such as GNU Diffutils[31] or Git's diff[32] highlight differences based on the line-based edit distances between them. With binary files, a comparison of lines or other text chunks is generally not meaningful; they are either classified as different or not.[31] Specifying data models in separate metadata files circumvents this limitation.

To compare the contents of files beyond the purpose of comparing versions, the similarity of individual strings, such as the keys and values in a metadata file, can be calculated based on their token- or character-based edit distance (e.g., using the Levenshtein distance[33]). Often strings are normalized beforehand (e.g., by lowering and lemmatizing of words). To also consider semantics, the vector similarity between strings can be calculated based on word embeddings (such as Word2Vec[34] and GloVe[35] embeddings). Alternatively, strings can be mapped

to concepts in ontologies or lexical databases (such as WordNet[36]) and compared based on the semantic relations they define, such as a synonyms or hyponyms. The meaning of a word or phrase depends on its context. Hence, consideration of the context is important. A-Match[37] is an example of an application that uses both string metrics and ontologies to compare APIs. Following a human-in-the-loop approach, the system's suggestions are visualized in a GUI and can be accepted or rejected.

When comparing more complex entities than strings, more information can be considered. For example, in addition to the parameter name, the allowed or actual value range can be considered (are the values within the expected range?) as well as the units (can they be converted into each other?). Consideration of the data structures is particularly important when comparing dataset and software interface descriptions. For RDF graphs, their structure, value ranges and other constraints can be defined using the Shapes Constraint Language (SHACL)[38]. Corresponding validators can be used to validate RDF graphs based on these definitions. The MINT (Model INTegration) framework[39] also requires the mapping of datasets to RDF. Variables are associated with a Scientific Variables Ontology (SVO) concept. Based on the unified RDF representations, both content and structure are considered when matching a set of user-selected target variables to models that can output them.

When comparing data models between datasets and software or services, both semantics and data structures must be taken into account, ideally independent of the file format to enable broad applicability. The functional extension of the DataDesc framework presented in this work encourages mapping to controlled vocabularies, is not restricted to a particular file format, and does not require the conversion of heterogeneous datasets into a unified representation.

Workflow Tools

Workflow tools play an important role in computational scientific work.[7] The use of workflow tools ranges from high performance computing environments to individual data analysis pipelines of scientists. Software solutions are correspondingly diverse, with software such as SLURM[40] in HPC contexts not only managing workflows but integrating resource management down to Jupyter notebooks[41] in individual data analysis pipelines with loose but easily adaptable workflows. The side-benefit for science of such workflow tools, especially when they provide a static workflow definition, is the creation of artefacts that are themselves suitable for documentation, and the further creation of traceability and reproducibility for these workflows. A widely used workflow tool is Snakemake[42], which binds programs to flexible, statically declared workflows and provides a reproducible execution environment. An example of one of the more prominent applications of such tools, at least in energy systems analysis, is the coupling of models into complex workflows for multi-perspective scientific analysis.[43]

Most of the workflow tools used in the scientific community, Snakemake being a rare exception, were not originally developed specifically for scientific activities. As a result, few workflow tools incorporate scientific metadata processing and handling by design. The importance of scientific metadata is now being increasingly recognised.[44,45] There, the tracking of data-related metadata and the computer-aided generation of metadata for generated datasets is central to the reusability and traceability of scientific results. The increasing popularity of the FAIR principles both among scientists[44] and funding agencies[46–48] only adds to the importance. The support of metadata information during workflows and of dedicated interfaces to scientific infrastructures such as metadata hubs is usually not natively covered by traditional generic workflow tools. Finally, workflow tools introduce their own complexity and require additional skills to be acquired by researchers. As workflow tool skills are a secondary concern in the scientific work environment, workflow managers with easy-to-learn principles and easy-to-use approaches are usually favored.

In summary, workflow tools often lack native support for scientific metadata and integration with scientific infrastructures, which makes data traceability and interoperability more difficult. Furthermore, their complexity and steep learning curve make them less attractive in scientific environments where ease of use is a priority.

In this work we demonstrate with our implementation of a scientific workflow in ioProc, how the issue of lock-in effects into a specific workflow manager and that workflow tools can be designed, such, that scientists can apply them with their preexisting knowledge in scripting and notebook environments.

Results

This section first describes a modular approach to software and data integration and shows how model workflows can be practically designed from reusable software components using the refinements and extensions of the DataDesc framework presented here. In a next step, ioProc is introduced as a workflow management tool that is designed to meet the special needs of scientific research and with which model workflows can be implemented transparently and reproducibly. Finally, DataDesc and ioProc are used to demonstrate the design and implementation of two application cases from the domain of energy systems analysis.

Designing Reusable Workflows with DataDesc

This work takes up the modular workflow concept presented by Kuckertz et al.[49], which propagates a flexible approach to data and software integration and thus offers a lightweight alternative to the laborious introduction and adoption of static interface and data standards in energy systems research (cf. Figure 1). Along the workflow concept, information transfers between software modules via non-persistent in-memory data formats, such as integers, arrays or classes, and is exchanged between software and data files based on persistent data formats, such as Excel, XML or NetCDF4. This work addresses the integration of persistent data formats into software interfaces. To this end, the concept first requires the metadata description of software and data artifacts, whereby in particular their inherent data models are annotated in detail. Once these are available, the data models mapped by software interfaces can be compared with those used in data files in a second step. This provides information about the compatibility of data and interfaces and identifies transformation requirements that can be covered by data processing. Along this process, the concept aims at the reuse of software and data artifacts and their flexible integration into complex application-specific model workflows. In addition to the metadata descriptions, transformation functions form a central component, as they act as a link between the individual software interfaces and heterogeneously structured datasets within the domain of energy research. The concept intends for the transformation functions to be designed in a modular way, described transparently and made available as open-source so that they can be jointly used and sustainably developed within the research community. Overall, the concept is intended to improve the reuse of research software and also the reproducibility and validation of study results, thereby promoting efficiency and academic exchange in the field of energy systems analysis.

In order to implement this concept in practice, metadata schemas are required that can be used to describe data models in detail both in interfaces and in datasets. These must be compatible with each other insofar as the annotations created on their basis allow them to be compared exactly with each other. In addition, the schemas should support semantic references to ontologies, such as the Open Energy Ontology[50] developed for energy systems research, and machine-actionable exchange formats so that the comparisons can be carried out automat-

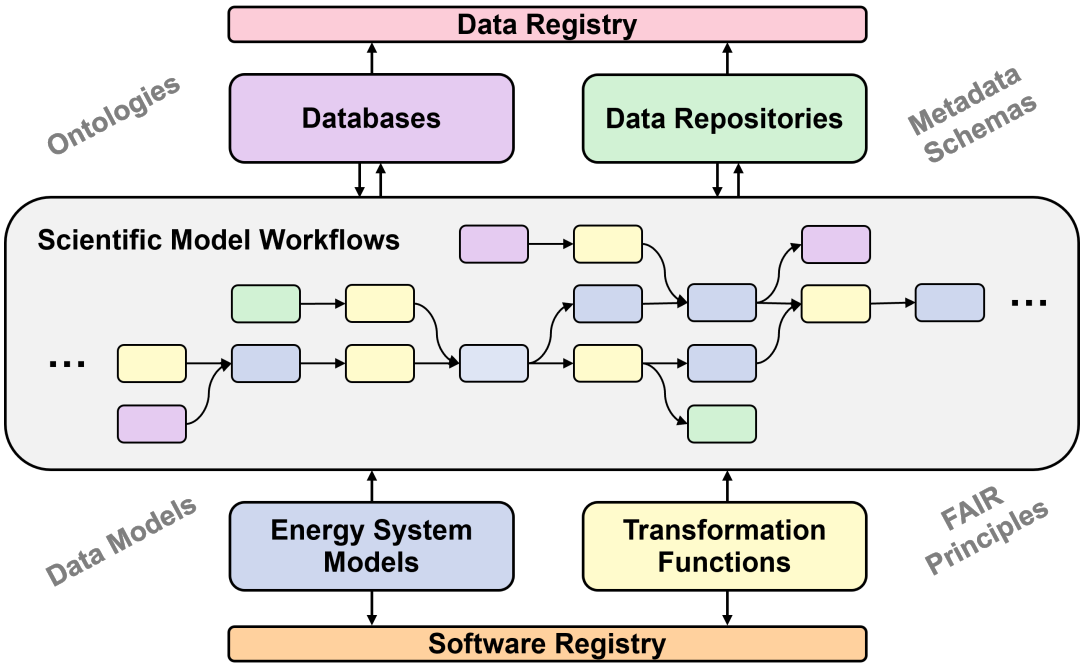

Figure 1: A modular workflow concept for the flexible integration of software and data in the context of a distributed research data infrastructure. The original figure by Kuckertz et al.[49] was supplemented by a software metadata registration component to facilitate the findability and reuse of energy systems models and data transformation functions.

ically. With DataDesc, Kuckertz et al.[30] have already developed a framework that meets these requirements, at least for the formal description of software interfaces. DataDesc centers around a software metadata schema that describes the data models on which software interfaces are based. In addition, DataDesc effectively promotes the FAIRness, i.e., findability, accessibility, interoperability, and reusability, of research software. As described in the *Related Work* section, there is currently no metadata schema for datasets that can be used to map their data models at a comparable level of detail. In order to present an approach to compensate for this omission, it is shown in the following to what extent the elements of the DataDesc metadata schema are also suitable for describing data models of datasets, and how the schema structure has been refined for this purpose.

The part of the DataDesc schema that is the focus of this work is the *Data Schema Object* (cf. Figure 2). Its properties can be used to describe even deeply nested data models in terms of contents, formats, value ranges and structures. The contents of data structures can be documented using *description* and clearly assigned to individual ontology classes using *semanticConcept*. It is of no consequence whether these contents are expected by a software interface or provided by a data file. The properties *type*, *format*, *mediaType*, and *charSet* can be used to describe data model components of both persistent and non-persistent variables. For example, the type and format can be used to specify not only the valid file type for variables that expect a file as input, but also the assumed data structure within a transferred file. The various *minimum* and *maximum* properties can be used to define value ranges that must be adhered to when using an interface in order to ensure error-free data processing. With regard to files, these properties describe specific value ranges contained in the dataset. Finally, the description of nested, grouping or dimensionally resolving data structures with *items* and *properties* remains the same, regardless of whether they are used in a function variable of complex type or a data file. Overall, it becomes evident that the Data Schema Object can be used to describe data models in files in the same way as in software interfaces.

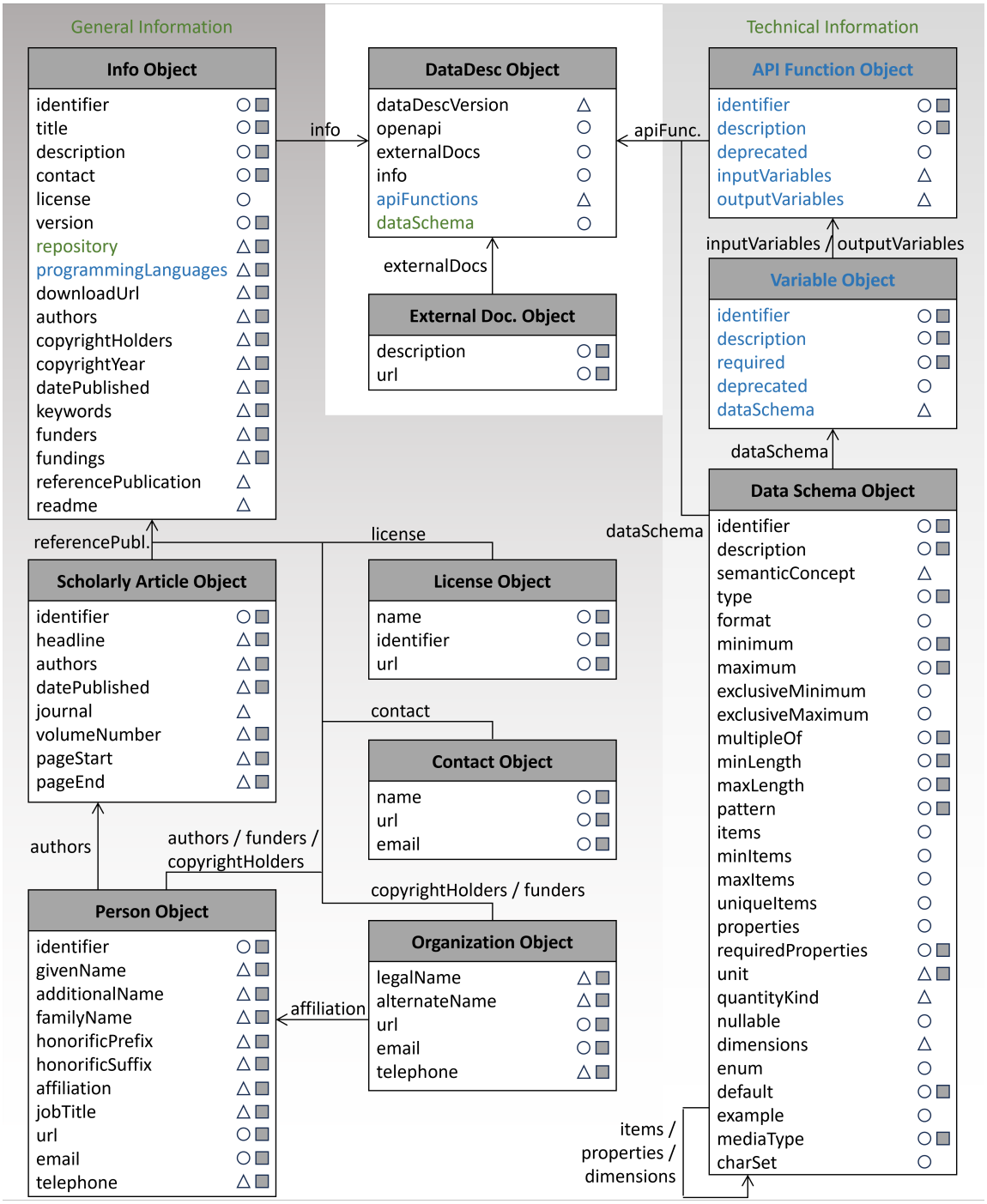

Figure 2: Structure and content of the DataDesc schema for describing software and their interface data models, adapted from Kuckertz et al.[30]. Both the general and the technical information are organized in information objects, with arrows indicating the various relationships between them. DataDesc properties that map directly or via extensions with the OpenAPI specification are indicated by white circles and white triangles, respectively. Properties that map to the Schema.org ontology are indicated by a gray square. The schema and figure from Kuckertz et al.[30] have been slightly adapted by the changes shown in green to accommodate the description of data models mapped in files. Components written in blue are intended exclusively for the description of interfaces, while all other objects and properties are also used to describe datasets. Individual property definitions can be viewed at GitHub: `https://github.com/FZJ-IEK3-VSA/DataDesc/blob/main/schema/DataDesc_schema_v1.2.md`.[51]

In Figure 2, object types and properties of the DataDesc schema that are intended exclusively for the description of interfaces are highlighted in color. It is clear to see that the majority, including the part required for the description of data models, is just as suitable for the annotation of data files as for software interfaces. The only structural difference is that the *Data Schema Object* is assigned directly to the *DataDesc Object* when describing files. By reusing the schema, also the exchange format of the DataDesc framework can be reused, which inherently ensures the homogeneity of the metadata descriptions.

On this basis, the DataDesc framework has been expanded to include analysis functionality that enables the direct and systematic comparison of a data model mapped in a file with the one implemented in a software interface and reveals the degree of their compatibility. In addition to the structures of the respective data models, also the properties of the variables they contain, such as dimensionality, value ranges and units, are contrasted individually. During the comparison process, the software interface is taken as the reference point from which the extent to which the data model of the dataset corresponds to the necessary interface structures is assessed. While the absence of expected variables or deviations from expected units or value ranges reduce the degree of compatibility, additional information contained in the file that is not required by the interface is ignored and does not have a negative effect. Essentially, the comparison process checks whether the information required by the software interface is available as a subset in the data file.

As a result of a comparison, a report is generated that provides detailed information about (in)compatibilities. Starting from an overall result that reflects whether the comparison has revealed discrepancies or not, the report goes into more and more detail along the hierarchy of the data model until it becomes clear at which points in the data model which requirements have not been met. This accurate error indication is intended to support researchers in the design and reuse of computational workflows, to avoid errors in data processing and to make software and data integration more efficient. At the same time, reports follow the DataDesc structure so that they can formally represent compatibility information as independent, machine-actionable DataDesc documents and can be further processed automatically.

If a file is to be used as input by a software despite incompatibilities, transformation requirements arise that must be covered by data processing. Thereby, necessary transformations can vary considerably in their complexity and scope, ranging from simple unit conversions to extensive data restructuring. As the report provides information about the type of incompatibility, it can be used to identify suitable transformation functions. For example, differences in currency formats can be resolved with CuCoPy[52], while discrepancies in the spatial or temporal resolution of the data can be addressed with Spagat[53] or tsam[54,55].

Many such transformation functions are currently available open-source. However, even if they are registered in software catalogs as indicated in Figure 1, they cannot yet be automatically suggested or selected on the basis of identified transformation requirements. With their formalized information on data models and incompatibilities, the machine-actionable DataDesc reports form a technical basis for the goal-oriented findability and reusability of data processing software.

Once the model and data components required for an analysis have been connected using the transformation functions identified via DataDesc, the modular concept subsequently requires them to be combined into transparent and reproducible computational workflows.

Implementing Reproducible Workflows with ioProc

ioProc is a library-level workflow manager, which distinguishes it from other tools such as the popular Snakemake workflow management system[7]. Snakemake operates on the software level,

so it ties individual software pieces together into a larger workflow. ioProc on the other hand chains functions, called actions in this context, into reproducible workflow specified by static definition file, called adapters. This adapter also serves as part of the scientific documentation of the workflow and can be used to run the workflow repeatedly and reproduce analysis results. In addition to the workflow itself, ioProc also writes log files, with one dedicated only to data modifications to adhere to good scientific practices. All read and write operations to and from data structures inside of ioProc actions are thus documented and traceable for each execution of the workflow. Furthermore, ioProc follows the open-closed principle, thus that it is open to extensions but closed to modifications. This manifests in different aspects of the software. For example, all actions are saved in an action folder, including the default set of built-in actions, that ioProc can generate on request. The ioProc generated actions are standardized, but as they are stored as a module, accessible by the user and outside of the ioProc software, opens them up for modification and extension without the need to modify ioProc itself. The user can furthermore declare own actions in additional files. ioProc, if configured with the location of these extension files, can pare these actions and apply them in a user specified workflow. In addition, users can share their actions with others so that they can use them in their own workflows. To improve reuseability, files with ioProc compatible actions can be modified with little mockup code, to make them independent from an ioProc installation. It is then possible to use these actions like ordinary Python functions in environments without ioProc, like a framework, a jupyter notebook or another workflow tool. ioProc thus creates an environment which is geared towards good scientific practices with a unique focus on library level workflows.

Application Cases

To illustrate our design and implementation of reproducible software workflows from reusable software and data components and to show the general applicability of the approaches presented, two independent and simplified model workflows are compared in this section (see Figure 3). With REMix (Renewable Energy Mix)[56] and ETHOS.FINE (Framework for Integrated Energy Assessment (FINE) of the Energy Transformation Pathway Optimization Suite (ETHOS)),[57,58] two mathematical optimization frameworks with a similar scope from the field of energy systems research are employed. Both application cases are based on pre-existing models that are instantiated from the two frameworks. While both models contain individual sets of predefined parameters and data, they also share some input datasets. The shared input data includes unresolved data regarding energy technologies on the one hand and temporally and spatially resolved renewable energy potentials on the other. During data processing within the two workflows, these differently structured datasets are adapted to the data models expected by the frameworks' interfaces along individual transformation steps, whereby some of the modular transformation methods are reused in both workflows. While the REMix workflow is implemented using ioProc and maps a file-based import of the data, the ETHOS.FINE workflow is implemented using Jupyter Notebook on the basis of a script-based data import.

When selecting the application cases, the focus was on the fact that they differ in a number of relevant characteristics, but remain structurally comparable. The model calculations carried out are realistic in terms of data structures, formats, contents and value ranges. Beyond that, however, they do not represent conclusive analyses of energy systems.

Shared Input Datasets

Both workflows ingest technology catalogues from the Danish Energy Agency[59], which provide a comprehensive but unresolved dataset of techno-economic data for technologies commonly

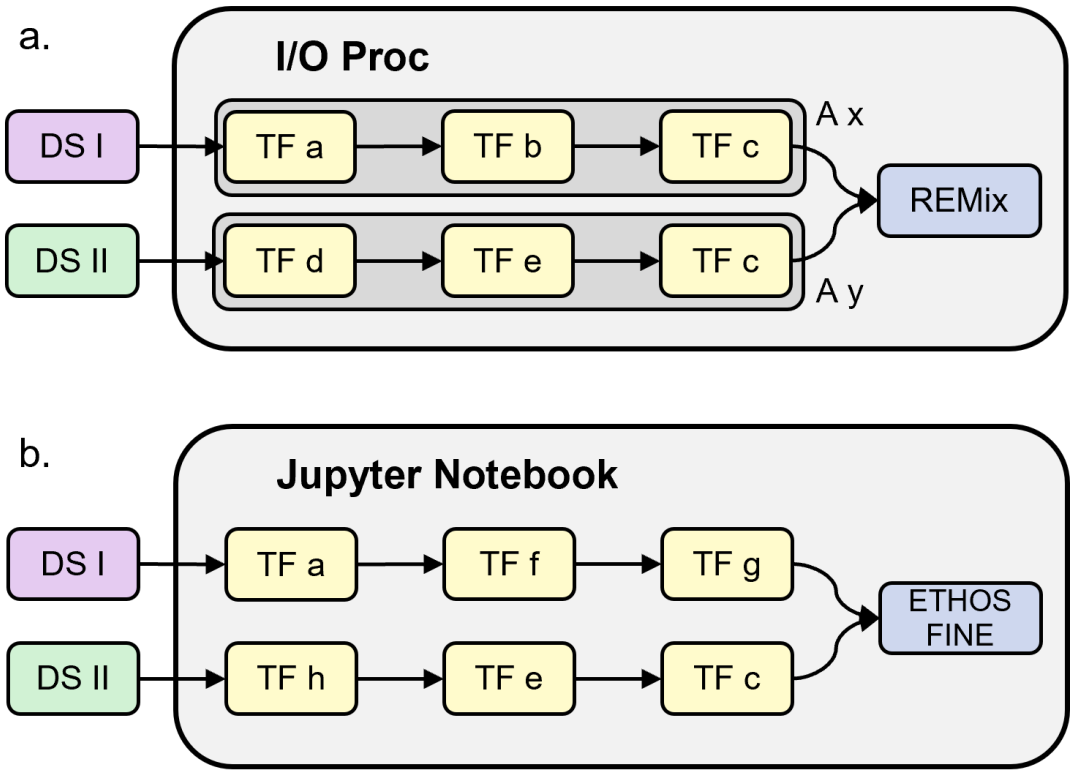

Figure 3: Schematic comparison of an ioProc (a.) and a Jupyter Notebook software workflow (b.) that use the same datasets (DS) as input and transfer their contents into the energy system models REMix and ETHOS.FINE using individually combined transformation functions (TF) and adapters (A).

used in energy systems analysis. The data is provided under the CC BY 4.0 license in Excel spreadsheets grouped by technology type, such as electricity and heat generation[60], storages[61] and renewable fuel production[62]. Each Excel file contains the worksheet *alldata_flat*, in which all data points have been converted into a row-by-row format and can therefore be easily processed by machines.

In addition, in the application cases, both workflows use PV and wind generation time series from Pfenninger and Staffell[63]. The time series are geographically and temporally detailed at country level, in this case for Germany, and in an hourly resolution for all weather years between 1980 and 2019. The data is provided by the website *renewables.ninja*[64] under the open CC BY-NC 4.0 license as CSV files, which are structured into two columns containing the timestamp and the normalized generation.

Figure 4 shows an abridged DataDesc document describing the data model of the PV generation time series for Germany. The CSV file comprises a tabular structure consisting of the two columns *national* (lines 13-18) and *time* (lines 23-25). While *national* holds the capacity factor float values for Germany in the range between 0 and 1, *time* represents the temporal dimension indexing the values using the coordinated universal time (UTC) format.

Shared Transformation Functions

In both workflows the same fundamental micro-transformations of data need to be performed. These operations were implemented to be compatible with ioProc and the general ones were applied both in the REMix and ETHOS.FINE workflows. Both workflows share the following transformation steps (we denote the exact function name of the ioProc action in brackets for

```
{
  "dataDescVersion": "0.1.0",
  "openapi": "3.0.0",
  ...
  "dataSchema" :
  {
    "type" : "object",
    "format" : {
      "example" : "text/csv"
    },
    "properties" : [
      {
        "identifier" : "national",
        "type" : "number",
        "format" : "float",
        "minimum" : 0.0,
        "maximum" : 1.0,
        "dimensions" : [ "time" ]
      }
    ],
    "dimensions" : [
      {
        "identifier" : "time",
        "type" : "string",
        "format" : "YYYY-MM-DD HH:MM:SS"
      }
    ]
  }
}
```

Figure 4: Abridged DataDesc document describing the data model of a country's PV generation time series in an hourly resolution as provided by *renewables.ninja*[64].

each step):

The workflow starts with reading the source data into a memory data structure for further processing (action *read_excel()*). We then clean the input data from whitespaces and special characters in the text based raw data (action *strip_data()*) and convert numerical data to python numerical types (action *remove_non_digit_values()*). The next steps are the reduction of the dataset to the technologies we are interested in (in this case gas turbines, Li-ion batteries, heat pumps and electrolyzers) (action *filter_technologies()*), to the relevant technology parameters (action *filter_parameters()*) and finally to the target years (action *filter_years()*) and the mean estimate (action *filter_estimate()*). Finally we exclude all columns with irrelevant data for our use case from the dataset (action *drop_columns()*).

In the next phase of the data preparation, we have to make adjustments of the dataset labels for example to be compatible with the REMix conventions. We hence rename the technologies (action *rename_technologies()*) and parameters (action *rename_parameters()*) to their given standard names. Afterwards, we convert all values to consistent base units (action *convert_units()*) and discount the costs to a uniform reference year (action *convert_currency()*). For the cost conversion we delegate the task to the Currency Conversion for Python (CuCoPy) library[52,65], which we call from inside of the action. CuCoPy is an open-source software package developed to streamline the integration of heterogenous currency data into research software. The package enables the conversion of currencies from over 260 countries, along with the ability to adjust financial data for inflation dating back to 1960, on an annual basis. As CuCoPy is independently released under the open MIT license, it seamlessly integrates into existing applications

but also works autonomously within processing workflows like the ones described here.

REMix Workflow Utilizing ioProc

REMix is a modular codebase that allows to set up linear optimization models and was developed by the Institute of Networked Energy Systems (VE) at the German Aerospace Centre (DLR) to address research questions in the field of energy systems modeling.[56] The framework is programmed in Python and relies on GAMS as its mathematical programming language. REMix can be used to address problems related to the optimal design and operation of future integrated energy systems, both for target systems and transformation pathways. It implements methodologies like Pareto optimal search, modeling to generate alternatives, perfect foresight and myopic optimization. One of the most important strengths of REMix is its high-performance scalability, since it is designed for processing large spatio-temporal optimization scopes. This makes it feasible to conduct path optimizations of high complexity with multiple years in hourly resolution and a high spatial resolution including balance area sizes of up to continental level and individual transmission lines. REMix is based on an implementation agnostic data model with makes it applicable for different scopes. The data model consists of five basic building blocks, nodes and goods. Nodes represent installations or connection points and goods describe everything that is exchanged withing the system. Converters balance the consumption and production of goods, storages allow the retention of goods across time, transfer links allow the transfer of these goods between nodes. Sources and sinks allow respectively the input and output of goods from and into the modeling scope. Complemented by indicators, users can model complex accounting relations like carbon taxes or electricity market prices.

The primary input to REMix models consists of structured data, stored in CSV or DAT files, which are loaded into the REMix model at the beginning of each model execution. These datasets are comprehensive, covering all aspects of the energy systems model being analyzed. They also play a crucial role in the configuration of the model, as REMix is a data-driven framework. Therefore, the combination of a specific REMix software version and a complete dataset constitutes a REMix instance, which is reproducible. In most cases, scientists do not create entirely new datasets from scratch, since research questions rarely focus on completely novel or previously not modeled systems. Instead, most modeling work is derived from a basic energy system model that defines the scope and area of interest. This requires scientists to provide data for specific parts of the model, such as particular technologies or regions, and properly link them to the overarching energy system data. Technically, this often involves modifying or replacing certain parts of an existing dataset. The core dataset that represents the energy system is referred to as the baseline dataset. For a dataset to be usable as a baseline, it must conform to the REMix conventions and be complete enough to execute the model without the need for additional data.

This application case focuses on the common task of adding specific data to a baseline model in REMix. To this end, we developed an exemplary workflow that integrates publicly available open data into a baseline dataset. The main tasks involved are transforming the data to fit REMix conventions and creating a new set of REMix-compatible input data files. We implemented this workflow in ioProc, which consists of a series of distinct, specific actions. The implementation is independent of any particular workflow or model context, enabling the exchange and sharing of source code between the REMix and ETHOS.FINE workflows.

The REMix workflow begins with 12 general data cleaning and transformation actions that focus on parsing and interpreting the input data format. These general actions are followed by REMix-specific steps to clean and transform the raw input data to comply with REMix conventions. In the final stage, the data is converted into the standard input data structure for REMix: a two-dimensional table format with standardized labels and columns. The next step involves

incorporating the new dataset into the baseline model. To do this, we read the baseline dataset, parameterize the entire model instance in REMix, and replace the relevant subset of the existing baseline data with the new, more detailed dataset. In this example, the new dataset introduces additional technologies, which must also be declared in REMix. This step completes the data preparation process. At this point, the first dataset is ready to be written as REMix input files, marking the final step of the workflow.

In this example, we incorporate two datasets, with the second being timeseries data. The timeseries data is handled using the same workflow described above, though the required transformations are fewer since the data is already in a format compatible with REMix. As a result, we can skip the cleaning and conversion steps. The next steps involve reducing the dataset to the target year for our model run and converting the labels to the REMix standard. The data is now ready to be integrated into the base REMix model, following the same procedure as with the technology dataset. We have now created a complete REMix input dataset consisting of the baseline dataset, along with the incorporated technology and timeseries data. These files are now ready for processing by REMix.

Reading a dataset into REMix is facilitated by a call to the *read_remix_csv* function, which we have documented as part of this project with the DataDesc schema to provide additional meta information and make it machine readable. The description includes a detailed description of the two input parameters *file* and *schema* which were described as separate variables with their own data schemas. While *file* is a simple string that refers to a CSV file *schema* is a complex data structure which refers to a JSON file that is composed of various fields (including *name*, *title*, *type*, *isAbout*, etc.). The return value of the function is the REMix interface internal data format, which is a *pandas.DataFrame*.

To demonstrate how specific operations in one workflow can be generalized, such that they become applicable in another workflow, we implemented two operations in the action format of ioProc andused them in also in the ETHOS.FINE workflow. The selected actions are the reading of the raw source data into memory (more precisely into pandas DataFrames) and the currency conversion, which are based on CuCoPy. This demonstrates, that atomic transformation operations, can be used across different workflow implementations and can be used to create a shared basis of operations.

The full ioProc workflow encompasses the raw input datasets and the baseline REMix model data, the workflow specification in the ioProc YAML file format, and the action folder containing all ioProc actions. This bundle is published alongside with this publication on GitHub and constitutes a reproducible workflow artefact.[66] With the documented ioProc version, it is now possible to reproduce the workflow. Furthermore, in the spirit of good scientific practice, the action folder, containing all operations needed for the workflow, is included and describes all scientific relevant transformation of the data. It therefore constitutes a different form of documentation, which can be used to examine the process in the future independent of an ioProc installation.

This approach supports reproducibility and transparency, as a separation of the basic data, the workflow software components and the scientifically relevant transformations is achieved. The ioProc actions can be used in the future to rebuild the workflow in any python context since the ioProc actions do not depend on an ioProc installation. But also the reverse is true. As easy as it is to use an ioProc action in a script, it is to modify existing source code in notebooks or scripts to be usable as ioProc actions. This enables, without compromising the ability to continue using the same code in existing scripts. The transformation into ioProc actions comes with the added benefit that the meta data handling, logging and limited change tracking of ioProc workflows are applicable without additional work.

From a research software engineering perspective, the ability to use ioProc action in other environments is an enabling feature, as the individual testing of actions becomes easy to implement. Such tests also serve as another form of scientific documentation beyond their usual

benefit of guaranteeing technical correctness. ioProc provides to scientist a framework in which to build flexible and granular transformation operations, which can be combined and extended as needed and applied in a broad range of software environments reaching from scripts and notebooks to full software applications.

ETHOS.FINE Workflow Utilizing Jupyter Notebook

The *Framework for Integrated Energy Assessment* is part of the *Energy Transformation Pathway Optimization Suite* at the division Jülich Systems Analysis of the the Institute of Energy and Climate Research at Forschungszentrum Jülich.[67] The Python-based framework serves as the basis for a diverse set of energy systems analyses conducted at the institute.[68–72] Their outcomes support investment decisions and policy-making in the transformation towards fully renewable energy systems. ETHOS.FINE leverages the capabilities of the general framework for linear optimization, Pyomo, to model linear optimization problems for energy systems analysis. ETHOS.FINE introduces the concepts of source, sink, storage and transmission components at different spatial and temporal resolutions as building blocks of an energy systems model. Typical problems that can be analyzed with such models are cost-optimized future energy systems of single buildings[68], municipalities[69,70], countries[71–73], or world-wide energy carrier transport[74]. Furthermore, ETHOS.FINE incorporates methods for spatial and temporal aggregation[54], tackling the problem of long calculation times for large-scale mixed-integer linear energy system optimization problems.

The relevant model information is stored in the *EnergySystemModel* container class. This class holds general information such as units and location lists, as well as instructions for the solving algorithm. In addition, instances of components are added to the *EnergySystemModel* class in a modular way with the help of the *EnergySystemModel.add()* method. The output of the optimization consists of values for the optimal design and operation of a minimum-cost system.

For the documentation of the programmatic interface, the constructors of the *EnergySystemModel* class and the component classes were described using DataDesc (*FINE.json* in the supporting material).[66] Lines 1-243 contain general metadata of the software. From line 264 onward the *EnergySystemModel.add()* constructor function is described. The data schema description includes an explanation of the properties, types, ranges, and default values and indicates required properties. Based on an example from the ETHOS.FINE repository[57], two Jupyter notebooks have been created that contain the processing of raw input data to the format that is needed for the model (*01_data_processing.ipynb*) as well as the model generation (*02_model_calculation.ipynb* in the supporting material). The preprocessing step reads timeseries for renewable energy potentials and energy demands from existing CSV files as well as techno-economic parameters from a JSON file. The renewable energy potential timeseries collected from the *renewables.ninja* website can be inserted into the models input CSV without transformation because they are a unitless values between 0 and 1. Parameter values for an absorption heat pump have been collected from the dataset of the Danish Energy Agency. Here, the preprocessing methods described in the previous sections have been used for reading and transforming the data. The raw data has been read from Excel into dictionary format with methods described in section *Shared Transformation Functions*. Economic values are described as EUR/MW in the input data while the model uses EUR/GW. The transformation has been conducted with the *convert_units()* method. Also, all monetary values are transformed to the year 2022 using CuCoPy.

In contrast to REMix, where one of the core functions with very few parameters was described, only a small function was described for ETHOS.FINE, the *EnergySystemModel.add()* function. The *add()* function expects many more parameters, some of which have custom complex data types that are not found in the standard Python package library. These user-defined

data type descriptions can be summarized in unique schemas and referenced multiple times in the document to avoid redundancy.

A helper script (*tools/comparison/main.py*) was used to automatically identify transformation needs between the API and the datasets. The script was called via the command line, where a DataDesc interface description, an input file description as well as a file path for the output file were specified. As software descriptions usually consist of more than just one interface, it was also necessary to specify which function and which parameter of said function should be compared with the input data. The comparison is always carried out from the perspective of the interface; meaning that a comparison is only successful if the input data serves all the required parameters of the interface. On execution, JSON-formatted report files were generated (see example in Figure 5), which provide information about overall compatibility (line 2), but also more detailed information about missing expected data (lines 9-10) and mismatching data formats (lines 18-23).

```
1  {
2    "match": false,
3    "detail": {
4      "dataSchema": {
5        "properties": [
6          {
7            "identifier": "windSpeed",
8            "unit": {
9              "expected": "https://qudt.org/vocab/unit/KiloM-PER-HR",
10             "received": null
11           },
12         }
13       ],
14       "dimensions": [
15         {
16           "identifier": "time",
17           "type": {
18             "expected": "integer",
19             "received": "string"
20           },
21           "format": {
22             "expected": null,
23             "received": "YYYY-MM-DD HH:MM:SS"
24           },
25         },
26       ]
27     }
28   },
29 }
```

Figure 5: Abridged DataDesc report[66] contrasting and comparing individual data model components of the ETHOS.FINE software interface and the wind generation time series from Pfenninger and Staffell[63] as provided by *renewables.ninja*[64].

Discussion

The coupling of research software and data into comprehensive workflows is common procedure in the computational sciences to help answer complex research questions. Thereby, workflows

are not only relevant as versatile and reusable software tools, but are also a central component of scientific exchange when it comes to tracing, reproducing and interpreting results data and ultimately increasing the reliability of derived findings.[3]

Leipzig et al.[14] point out that metadata plays a special role in the context of reproducible computational research. They characterize the description of input or raw data and file intermediates as they are processed in workflows as an essential core task of metadata. However, since none of the established metadata schemas provide a sufficient description of the data models used in data sets and software interfaces, information that can be automatically processed to enable easy coupling of components in workflows is rarely available. The presented refinement of the DataDesc schema, which allows data models to be annotated not only in software interfaces but also in data sets, closes this gap and thus represents a necessary addition to established domain-agnostic metadata standards like Dublin Core or schema.org.

The added value resulting from the ability to formally describe information of this kind in the form of metadata is manifested in the downstream applications that this enables. Leipzig et al.[14] state file format and content sanity checks, which are defined by input metadata but implemented at the workflow level, as a prominent use case that has not yet been realized. To address this use case, the DataDesc framework was expanded to include a tool that not only automatically compares data content and formats, but also data structures and value ranges, identifies discrepancies as transformation requirements for successful integration, and describes them in a machine-actionable form. The ioProc workflow manager presented here then offers the option of addressing identified transformation requirements by bundling and integrating transformation paths in the form of modular adapters and making these available for reuse. Based on the utilized modular workflow concept for the integration of software and data components, DataDesc and ioProc support researchers in their manual design of research software workflows.

A promising application based on this comes from the innovative field of computer-aided workflow design and integration, which could use such machine-actionable information to automatically identify suitable data conversion and integration paths for model chains and suggest them to researchers. The web-based scientific data processing platform Galaxy,[75] for example, serves as a platform for describing, sharing, and publishing scientific calculation processes and is designed to facilitate their discoverability and reusability in an accessible and interoperable manner. Kumar et al.[76] have developed an approach that enables the platform to suggest tool combinations based on usage patterns identified using deep learning. At this point, DataDesc's automated comparison of the data models would provide information about transformation requirements and a qualitative assessment of the proposed workflows. Computational sciences have access to huge amounts of heterogeneous research data and research software. These can no longer be fully evaluated using conventional data and tool searches, which often leads to low reuse rates and redundant developments.[77,78] Against this background, the computer-assisted selection and integration of components into one's own research workflows is of particular importance.

In addition to designing and implementing scientific software workflows, ensuring that they are used transparently and comprehensibly in studies and analyses remains a key challenge. In this context, the problem of reproducibility in research has already been comprehensively documented.[79] Leipzig et al.[14] describe computational process pipelines as a widely used method for performing scientific analyses, that encourages parameterization and configuration that promote reproducibility. With ioProc, a lightweight workflow manager has been released that natively implements the modular workflow concept for integrating research data and software in the form of adapters within reproducible and reusable data pipelines. Overall, the concept and tool developments presented here aim to help close the gap between abstract guidelines, such as those systematized in the FAIR principles, and the daily work reality of researchers and to balance the effort of metadata documentation with the corresponding practical benefits in handling scientific

workflows.

Limitations

In this work, we have for the first time presented application examples to demonstrate in a coherent and comprehensive manner how metadata-based design and modular coupling processes interact and can be implemented. Although the FAIR principles and open source and open data are already becoming increasingly widespread, it will take some time before the publication of entire workflows and modular transformation functions becomes standard practice in science. As the approach presented here benefits from the high availability of existing and well-annotated software and data components to quickly identify and integrate needed components into the user's own workflows, its effect is initially limited and will only develop its full potential with the increasing expansion and use of digital research infrastructures such as workflow repositories and software registries. The DataDesc schema and the ioProc workflow manager are complementary components in this context, which, through further integration into the infrastructure, will offer even better embedded and holistic solutions for everyday use.

The approaches and tools presented in this paper aim at loosely coupling models via transformation adapters, with the goal of achieving flexible interoperability between calculation steps that may differ in data formats, data types, syntax, systems, and execution times. No adaptation of data sets or software to predefined data and interface standards is required. In application contexts where workflows are mostly hard coupled, the direct benefit of the presented approach is limited. In co-simulation, for example, models are usually provided with standardized interfaces that do not require further transformation when interconnected. Only when hard and loose coupling of models is combined within hierarchical workflows should clearly annotated and reusable adapters be used again.

Outlook

Based on this work, future studies could further develop both the presented implementation and design aspects. A reasonable next step would be the automated annotation of data files, starting with a selection of especially compatible formats and working from there. Also, further metadata publication pipelines should be included in the framework in order to supplement existing connections to software publication platforms with those for data publications, such as the Open Energy Databus[80]. Correspondingly the creation of a public transformation function repository or catalogue with a stable API together would greatly improve transparency and contribute to findability and reusability, i.e., core values of the FAIR principles. Based on such a data and transformation function availability, tools can be developed which automatically identify the needed transformation functions for connecting two different data formats (such as the format of a data source and the format of a model data input) and the ability to create lists of needed and missing transformations for scientists to build their workflows from. This would then open up the possibility to further investigate limitations and best practices for automated generation of workflows in different scientific contexts. Additional tooling can then be developed like recommendation systems for transformation functions or alternative workflow formulations based on external factors like license compatibility. A different direction for future work would be the improvement of the workflow tools themselves, such that they include automatic tracing of data modifications along a specified workflow and including this information into metadata formats. This can then be expanded towards systems that trace also the reason behind modifications and support scientists in adding contextual information during their workflows in a user-friendly way.

Resource Availability

Lead Contact

Further information and requests for resources on the DataDesc framework should be directed to and will be fulfilled by Patrick Kuckertz (p.kuckertz@fz-juelich.de).

Further information and requests for resources on the ioProc workflow manager should be directed to and will be fulfilled by Benjamin Fuchs (benjamin.fuchs@dlr.de).

Materials Availability

This study did not generate new unique reagents.

Data and Code Availability

The DataDesc metadata schema and the source code of the developed comparison tool for DataDesc documents are publicly available under the open MIT license at `https://github.com/FZJ-IEK3-VSA/DataDesc`.[51] In addition, these resources have been made available in their current version 1.0 in the JülichDATA repository under the CC0 public domain dedication at `https://doi.org/10.26165/JUELICH-DATA/DLCYV5`.

The ioProc workflow manager in version 2.2.0 under MIT license was used for this work and is publicly available at `https://pypi.org/project/ioproc/` and as a git repository at `https://gitlab.com/dlr-ve/esy/ioproc`. Furthermore the addon ioprocmeta was also used in this work and is available under BSD-3-clause license at `https://pypi.org/project/ioprocmeta/` and as a git repository at `https://gitlab.com/dlr-ve/esy/ioprocmeta`.

The example and documentation files related to the presented application case are published in a git repository at `https://github.com/dlr-ve-esy/modelcouplingworkflows` under BSD-3-clause and CC-BY 4.0 licenses.

Acknowledgements

The authors would like to thank the Federal Ministry for Economic Affairs and Climate Action of Germany (BMWK) for supporting this work with a grant for the project LOD-GEOSS (03EI1005A-G). Furthermore, the authors are grateful to the German federal government, the German state governments, and the Joint Science Conference (GWK) for their funding and support as part of the NFDI4Ing consortium, managed by the German Research Foundation (DFG) – 442146713. This work was also supported by the Helmholtz Association as part of the program ”Energy System Design”.

Author Contributions

Conceptualization: P.K., and B.F.; methodology: P.K., B.F., and J.S.; software: B.F., H.G., K.K., and J.S.; validation: K.K., H.G., J.S., B.F., and P.K.; investigation: P.K., B.F., J.S., and J.G.; data curation: H.G., B.F., E.A.R., and J.S.; writing – original draft: P.K., B.F., J.G., E.A.R., K.K., H.G., and J.S.; writing - review & editing: P.K., B.F., J.S., H.G., K.K., E.A.R., J.G., H.C.G., J.M.W., P.J.,

and J.L.; visualization: P.K.; supervision: P.K., B.F., J.M.W., J.L., and P.J.; project administration: P.K., B.F., J.L., and P.J.; funding acquisition: P.J., and D.S.

Declaration of Interests

The authors declare no competing interests.

Declaration of generative AI and AI-assisted technologies

During the preparation of this work the authors used the tools DeepL and ChatGPT to check grammar and spelling in a few places, and to make minor improvements to readability and style. After using these tools, the authors reviewed and edited the content as needed, and take full responsibility for the content of the publication.

References

1. Askeland, M., Morch, A., Papadimitriou, C., Di Somma, M., Coccia, A., Pinel, D., Richardson, P., and Sforza, G. (2023). Workflow-based architecture for optimal planning of integrated local multi-energy systems. In: 2023 International Conference on Smart Energy Systems and Technologies (SEST). IEEE (1–6).

2. Liu, J., Braun, E., Düpmeier, C., Kuckertz, P., Ryberg, D. S., Robinius, M., Stolten, D., and Hagenmeyer, V. (2019). Architectural concept and evaluation of a framework for the efficient automation of computational scientific workflows: An energy systems analysis example. Applied Sciences *9*, 728.

3. Pelser, T., Weinand, J. M., Kuckertz, P., and Stolten, D. (2025). ETHOS.REFLOW: An open-source workflow for reproducible renewable energy potential assessments. Patterns.

4. Pueblas, R., Kuckertz, P., Weinand, J. M., Kotzur, L., and Stolten, D. (2023). ETHOS.PASSION: An open-source workflow for rooftop photovoltaic potential assessments from satellite imagery. Solar Energy *265*, 112094.

5. Pfenninger, S., Hirth, L., Schlecht, I., Schmid, E., Wiese, F., Brown, T., Davis, C., Gidden, M., Heinrichs, H., Heuberger, C. et al. (2018). Opening the black box of energy modelling: Strategies and lessons learned. Energy Strategy Reviews *19*, 63–71.

6. Kieser, S., Brown, J., Zdobnov, E. M., Trajkovski, M., and McCue, L. A. (2020). Atlas: a snakemake workflow for assembly, annotation, and genomic binning of metagenome sequence data. BMC Bioinformatics *21*, 257. `https://doi.org/10.1186/s12859-020-03585-4`.

7. Mölder, F., Jablonski, K. P., Letcher, B., Hall, M. B., Tomkins-Tinch, C. H., Sochat, V., Forster, J., Lee, S., Twardziok, S. O., Kanitz, A., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., and Köster, J. (2021). Sustainable data analysis with snakemake. F1000Research *10*.

8. Pfenninger, S. (2017). Energy scientists must show their workings. Nature *542*, 393–393.

9. Pelser, T., Weinand, J. M., Kuckertz, P., McKenna, R., Linssen, J., and Stolten, D. (2023). Reviewing accuracy & reproducibility of large-scale wind resource assessments. Advances in Applied Energy (100158).

10. Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M. R., Peters, K., and Schober, D. (2020). Fair computational workflows. Data Intelligence *2*, 108–121.

11. Pimentel, J. F., Murta, L., Braganholo, V., and Freire, J. (2019). A large-scale study about quality and reproducibility of jupyter notebooks. In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). (507–517). https://doi.org/10.1109/MSR.2019.00077.

12. Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B. (2016). The fair guiding principles for scientific data management and stewardship. Scientific Data *3*, 160018. https://doi.org/10.1038/sdata.2016.18.

13. Booshehri, M., Emele, L., Flügel, S., Förster, H., Frey, J., Frey, U., Glauer, M., Hastings, J., Hofmann, C., Hoyer-Klick, C., Hülk, L., Kleinau, A., Knosala, K., Kotzur, L., Kuckertz, P., Mossakowski, T., Muschner, C., Neuhaus, F., Pehl, M., Robinius, M., Sehn, V., and Stappel, M. (2021). Introducing the open energy ontology: Enhancing data interpretation and interfacing in energy systems analysis. Energy and AI *5*, 100074. https://doi.org/10.1016/j.egyai.2021.100074.

14. Leipzig, J., Nüst, D., Hoyt, C. T., Ram, K., and Greenberg, J. (2021). The role of metadata in reproducible computational research. Patterns *2*.

15. Fuchs, B., Riehm, J., Nitsch, F., and Wulff, N. (2020). ioproc - a light-weight workflow manager in python. https://gitlab.com/dlr-ve/esy/ioproc.

16. Chan, L. M., and Zeng, M. L. (2006). Metadata interoperability and standardization - A study of methodology, part I: achieving interoperability at the schema level. D Lib Mag. *12*. https://doi.org/10.1045/JUNE2006-CHAN.

17. Dublin Core Metadata Initiative (DCMI). DCMI Metadata Terms. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/.

18. DataCite – International Data Citation Initiative e.V.. DataCite Metadata Schema 4.5. https://schema.datacite.org/.

19. World Wide Web Consortium (W3C). Data Catalog Vocabulary (DCAT) - Version 3. https://www.w3.org/TR/vocab-dcat-3/.

20. Schema.org. Schema.org Dataset. https://schema.org/Dataset.

21. Soiland-Reyes, S., Sefton, P., Crosas, M., Castro, L. J., Coppens, F., Fernández, J. M., Garijo, D., Grüning, B. A., Rosa, M. L., Leo, S., Carragáin, E. Ó., Portier, M., Trisovic, A., Community, R., Groth, P., and Goble, C. A. (2022). Packaging research artefacts with ro-crate. Data Sci. *5*, 97–138. https://doi.org/10.3233/DS-210053.

22. Frictionless Data. Data Package (v1). https://specs.frictionlessdata.io/.

23. Hülk, L., Huber, J., Hofmann, C., and Muschner, C. (2024). Open Energy Family - Open Energy Metadata (OEMetadata). GitHub. https://github.com/OpenEnergyPlatform/oemetadata.

24. World Wide Web Consortium (W3C). The RDF Data Cube Vocabulary. http://www.w3.org/TR/vocab-data-cube/.

25. Vu, B., Pujara, J., and Knoblock, C. A. (2019). D-repr: A language for describing and mapping diversely-structured data sources to rdf. In: Proceedings of the 10th International Conference on Knowledge Capture. K-CAP '19 New York, NY, USA: Association for Computing Machinery. ISBN 9781450370080 (189–196). https://doi.org/10.1145/3360901.3364449.

26. Garijo, D., Ratnakar, V., Gil, Y., and Khider, D.. The software description ontology. revision: 1.9.0. https://w3id.org/okn/o/sd/1.9.0.

27. The HDF Group. Hierarchical Data Format, version 5. https://www.hdfgroup.org/HDF5/.

28. Unidata. Network Common Data Form (NetCDF). https://doi.org/10.5065/D6H70CW6.

29. MLCommons. Croissant Format Specification - Version 1.0. http://mlcommons.org/croissant/1.0.

30. Kuckertz, P., Göpfert, J., Karras, O., Neuroth, D., Schönau, J., Pueblas, R., Ferenz, S., Engel, F., Pflugradt, N., Weinand, J. M., Nießе, A., Auer, S., and Stolten, D. (2024). Datadesc: A framework for creating and sharing technical metadata for research software interfaces. Patterns *5*. https://doi.org/10.1016/j.patter.2024.101064.

31. Foundation, F. S.. Gnu diffutils - comparing and merging files. https://www.gnu.org/software/diffutils/manual/.

32. Project, G.. git-diff - show changes between commits, commit and working tree, etc. https://git-scm.com/docs/git-diff.

33. Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady *10*, 707.

34. Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. Preprint at arXiv. https://doi.org/10.48550/arXiv.1301.3781.

35. Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). (1532–1543).

36. Fellbaum, C. WordNet: An Electronic Lexical Database. The MIT Press (1998). ISBN 9780262272551. https://doi.org/10.7551/mitpress/7287.001.0001.

37. Möbius, M., Popendicker, K. J., and Böning, S. (2022). A-match. Zenodo. https://doi.org/10.5281/zenodo.6641652.

38. World Wide Web Consortium (W3C). Shapes Constraint Language (SHACL). https://www.w3.org/TR/shacl/.

39. Gil, Y., Garijo, D., Khider, D., Knoblock, C. A., Ratnakar, V., Osorio, M., Vargas, H., Pham, M., Pujara, J., Shbita, B., Vu, B., Chiang, Y.-Y., Feldman, D., Lin, Y., Song, H., Kumar, V., Khandelwal, A., Steinbach, M., Tayal, K., Xu, S., Pierce, S. A., Pearson, L., Hardesty-Lewis, D., Deelman, E., Silva, R. F. D., Mayani, R., Kemanian, A. R., Shi, Y., Leonard, L., Peckham, S., Stoica, M., Cobourn, K., Zhang, Z., Duffy, C., and Shu, L. (2021). Artificial Intelligence for Modeling Complex Systems: Taming the Complexity of Expert Models to Improve Decision Making. ACM Transactions on Interactive Intelligent Systems *11*, 1–49. https://doi.org/10.1145/3453172.

40. Yoo, A. B., Jette, M. A., and Grondona, M. (2003). Slurm: Simple linux utility for resource management. In: Feitelson, D., Rudolph, L., and Schwiegelshohn, U., eds. Job Scheduling Strategies for Parallel Processing. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-39727-4 (44–60).

41. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., and development team, J. (2016). Jupyter notebooks - a publishing format for reproducible computational workflows. In: Loizides, F., and Scmidt, B., eds. Positioning and Power in Academic Publishing: Players, Agents and Agendas. Netherlands: IOS Press (87–90).

42. Köster, J., and Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. Bioinformatics *28*, 2520–2522. https://doi.org/10.1093/bioinformatics/bts480.

43. Chang, M., Lund, H., Thellufsen, J. Z., and Østergaard, P. A. (2023). Perspectives on purpose-driven coupling of energy system models. Energy *265*, 126335. https://doi.org/10.1016/j.energy.2022.126335.

44. Wierling, A., Schwanitz, V. J., Altinci, S., Balazinska, M., Barber, M. J., Biresselioglu, M. E., Celino, M., Demir, M. H., Dennis, R., Dintzer, N., el Gammal, A., Fernandes-Peruchena, C. M., Gilcrease, W., Gladysz, P., Hoyer-Klick, C., Joshi, K., Kruczek, M., Lacroix, D., Molgorzata, M., Mayo-Garcia, R., Morrrison, R., Paier, M., Perenato, G., Ramakrishan, M., Reid, J., Sciullo, A., Solark, B., Suna, D., Süß, W., Unger, A., Fernandez Vanoni, M. L., and Vasiljevic, N. (2021). Fair metadata standards for low carbon energy research - a review of practices and how to advance. Energies *14*, 6692. https://doi.org/10.3390/en14206692.

45. Schriml, L. M., Chuvochina, M., Davies, N., Eloe-Fadrosh, E. A., Finn, R. D., Hugenholtz, P., Hunter, C. I., Hurwitz, B. L., Kyrpides, N. C., Meyer, F., Mizrachi, I. K., Sansone, S.-A., Sutton, G., Tighe, S., and Walls, R. (2020). Covid-19 pandemic reveals the peril of ignoring metadata standards. Scientific Data *7*, 188. https://doi.org/10.1038/s41597-020-0524-5.

46. Forschungsgemeinschaft, D. (2022). Guidelines for Safeguarding Good Research Practice. Code of Conduct. Deutsche Forschungsgemeinschaft. https://doi.org/10.5281/zenodo.6472827.

47. Federal Ministry of Education and Research. Aktionsplan Forschungsdaten. https://www.bmbf.de/bmbf/de/forschung/digitale-wirtschaft-und-gesellschaft/aktionsplan-forschungsdaten/aktionsplan-forschungsdaten_node.html.

48. European Commission. Open science. https://rea.ec.europa.eu/open-science_en.

49. Kuckertz, P., Göpfert, J., Pueblas, R., Schönau, J., Weinand, J. M., and Stolten, D. (2023). Software and data integration. Forschungszentrum Jülich GmbH. https://doi.org/10.5281/zenodo.8117988.

50. Booshehri, M., Emele, L., Flügel, S., Förster, H., Frey, J., Frey, U., Glauer, M., Hastings, J., Hofmann, C., Hoyer-Klick, C. et al. (2021). Introducing the open energy ontology: Enhancing data interpretation and interfacing in energy systems analysis. Energy and AI *5*, 100074.

51. Kuckertz, P., Göpfert, J., Karras, O., Neuroth, D., Schönau, J., Pueblas, R., Ferenz, S., Engel, F., Pflugradt, N., Weinand, J. M., Kotzur, L., Nießе, A., Auer, S., and Stolten, D. (2024). DataDesc - A framework for machine-actionable software metadata. Forschungszentrum Jülich GmbH. https://github.com/FZJ-IEK3-VSA/DataDesc.

52. Schönau, J., Kuckertz, P., Weinand, J. M., Kotzur, L., and Stolten, D. (2024). Currency Conversion for Python (CuCoPy). Forschungszentrum Jülich GmbH. https://github.com/FZJ-IEK3-VSA/CuCoPy.

53. Patil, S., Kotzur, L., and Stolten, D. (2022). Advanced spatial and technological aggregation scheme for energy system models. Energies *15*, 9517. https://doi.org/10.3390/en15249517.

54. Hoffmann, M., Kotzur, L., and Stolten, D. (2022). The pareto-optimal temporal aggregation of energy system models. Applied Energy *315*, 119029.

55. Forschungszentrum Jülich GmbH - Jülich Systems Analysis (ICE-2) (2022). tsam - Time Series Aggregation Module. Forschungszentrum Jülich GmbH. https://github.com/FZJ-IEK3-VSA/tsam.

56. German Aerospace Center (DLR) - Institute of Networked Energy Systems (2022). REMix - Framework for energy system optimization models. GitLab. https://gitlab.com/dlr-ve/esy/remix/framework.

57. Forschungszentrum Jülich GmbH - Jülich Systems Analysis (ICE-2) (2023). ETHOS.FINE - Framework for Integrated Energy System Assessment. Forschungszentrum Jülich GmbH. https://github.com/FZJ-IEK3-VSA/FINE.

58. Forschungszentrum Jülich GmbH - Jülich Systems Analysis (ICE-2). ReadTheDocs Documentation of ETHOS.FINE - Framework for Integrated Energy System Assessment. https://vsa-fine.readthedocs.io/en/master/.

59. Danish Energy Agency (2025). The danish energy agency. Danish Energy Agency. https://ens.dk/en.

60. Danish Energy Agency (2024). Technology data - energy plants for electricity and district heating generation. Danish Energy Agency. https://ens.dk/media/6378/download.

61. Danish Energy Agency (2024). Technology data – energy storage. Danish Energy Agency. https://ens.dk/media/6446/download.

62. Danish Energy Agency (2024). Technology data – renewable fuels. Danish Energy Agency. https://ens.dk/media/6444/download.

63. Pfenninger, S., and Staffell, I. (2016). Long-term patterns of european pv output using 30 years of validated hourly reanalysis and satellite data. Energy *114*, 1251–1265.

64. Pfenninger, Stefan and Staffell, Iain. Renewables.ninja. https://www.renewables.ninja/.

65. Weinand, J. M., Hoffmann, M., Göpfert, J., Terlouw, T., Schönau, J., Kuckertz, P., McKenna, R., Kotzur, L., Linßen, J., and Stolten, D. (2023). Global lcoes of decentralized off-grid renewable energy systems. Renewable and Sustainable Energy Reviews *183*, 113478.

66. Kuckertz, P., Fuchs, B., Schönau, J., Gardian, H., Knosala, K., Arellano Ruiz, E., Göpfert, J., Gils, H. C., Weinand, J. M., Jochem, P., Linßen, J., and Stolten, D. (2025). Supplementary material to 'Model coupling through reproducible adapter workflows based on shared transformation functions'. Forschungszentrum Jülich GmbH, Deutsches Zentrum für Luft- und Raumfahrt (DLR). https://github.com/dlr-ve-esy/modelcouplingworkflows.

67. Groß, T., Knosala, K., Hoffmann, M., Pflugradt, N., and Stolten, D. (2023). Ethos.fine: A framework for integrated energy system assessment. Preprint at arXiv. https://doi.org/10.48550/arXiv.2311.05930.

68. Omoyele, O., Matrone, S., Hoffmann, M., Ogliari, E., Weinand, J. M., Leva, S., and Stolten, D. (2024). Impact of temporal resolution on the design and reliability of residential energy systems. Energy and Buildings *319*, 114411.

69. Risch, S., Weinand, J. M., Schulze, K., Vartak, S., Kleinebrahm, M., Pflugradt, N., Kullmann, F., Kotzur, L., McKenna, R., and Stolten, D. (2024). Scaling energy system optimizations: Techno-economic assessment of energy autonomy in 11 000 german municipalities. Energy Conversion and Management *309*, 118422.

70. Weinand, J. M., Vandenberg, G., Risch, S., Behrens, J., Pflugradt, N., Linßen, J., and Stolten, D. (2023). Low-carbon lithium extraction makes deep geothermal plants cost-competitive in future energy systems. Advances in Applied Energy *11*, 100148.

71. Jacob, R., Hoffmann, M., Weinand, J. M., Linßen, J., Stolten, D., and Müller, M. (2023). The future role of thermal energy storage in 100% renewable electricity systems. Renewable and Sustainable Energy Transition *4*, 100059.

72. Tsani, T., Pelser, T., Ioannidis, R., Maier, R., Chen, R., Risch, S., Kullmann, F., McKenna, R., Stolten, D., and Weinand, J. (2024). Out of sight, out of mind? cost of minimizing visibility of nationwide renewable energy systems. Preprint at Research Square. https://doi.org/10.21203/rs.3.rs-5017073/v1.

73. Schöb, T., Kullmann, F., Linßen, J., and Stolten, D. (2023). The role of hydrogen for a greenhouse gas-neutral germany by 2045. international journal of hydrogen energy *48*, 39124–39137.

74. Franzmann, D., Heinrichs, H., Lippkau, F., Addanki, T., Winkler, C., Buchenberg, P., Hamacher, T., Blesl, M., Linßen, J., and Stolten, D. (2023). Green hydrogen cost-potentials for global trade. international journal of hydrogen energy *48*, 33062–33076.

75. Community, T. G. (2024). The galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. Nucleic Acids Research *52*, W83–W94. https://doi.org/10.1093/nar/gkae410.

76. Kumar, A., Rasche, H., Grüning, B., and Backofen, R. (2021). Tool recommender system in galaxy using deep learning. GigaScience *10*, giaa152.

77. Habermann, T. (2020). Metadata and reuse: Antidotes to information entropy. Patterns *1*.

78. Palmblad, M., Lamprecht, A.-L., Ison, J., and Schwämmle, V. (2019). Automated workflow composition in mass spectrometry-based proteomics. Bioinformatics *35*, 656–664.

79. Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. Nature Publishing Group UK London.

80. DBpedia Association affiliated with Institut für Angewandte Informatik e. V. (InfAI). Open Energy Databus. https://databus.openenergyplatform.org/.