Advanced Engineering Materials

ADVANCED ENGINEERING MATERIALS
www.aem-journal.com

RESEARCH ARTICLE OPEN ACCESS

A Lightweight Procedural Layer for Hybrid Experimental–Computational Workflows in Materials Science

Steffen Brinckmann[1] | Sarath Menon[2,3] | Raphael Röske[1] | Johannes Steinhülb[4] | Michael Selzer[4] | Jan Janssen[2] | Tilmann Hickel[5] | Ruth Schwaiger[1,6] | Jörg Neugebauer[2]

[1]Institute of Energy Materials and Devices, Structure and Function of Materials (IMD-1), Forschungszentrum Jülich GmbH, Jülich, Germany | [2]Computational Materials Design, MPI-SUSMAT, Düsseldorf, Germany | [3]Atomistic Modeling and Simulation, ICAMS, Bochum, Germany | [4]Institute for Nanotechnology, Karlsruhe Institute for Technology, Karlsruhe, Germany | [5]Materialinformatik, BAM, Berlin, Germany | [6]Chair of Energy Engineering Materials, Faculty 5, RWTH Aachen University, Aachen, Germany

Correspondence: Steffen Brinckmann (s.brinckmann@fz-juelich.de)

Received: 8 December 2025 | **Revised:** 10 March 2026 | **Accepted:** 11 March 2026

Keywords: data model | FAIR | materials science | metadata | provenance | workflow interoperability

ABSTRACT

We present a prototype implementation of a framework for hybrid workflows that integrates automated computation and analysis with manual experimental measurements. Leveraging the pyiron workflow engine, we introduce a lightweight, parameterized procedure description layer that can adjust instrument settings and orchestrate human interventions. Rather than replacing the existing execution engine, we add a minimal abstraction layer that translates procedure descriptions into executable steps for manual operations, enabling seamless handoffs between automated tasks and manual experimental tasks. We demonstrate the approach on a use case that combines manual tensile testing with subsequent analytical evaluation and result aggregation, illustrating how parameters and metadata propagate through the workflow and how instrument state changes and measurement results are captured. We also report a usability study that quantifies the ease with which lab scientists can create and modify workflows. Finally, we summarize lessons learned from this prototype, including improved provenance capture and streamlined experimental orchestration, as well as current limitations. We conclude that the proposed lightweight hybrid workflow description offers a promising path to bridging automation, computation and manual experimentation, and we outline directions for future work.

1 | Introduction

Materials science workflows increasingly span automated computation, data analysis, and hands-on experimental laboratory work, creating a pressing need for universal workflow descriptions that enable reproducibility, FAIR [1] data practices, and robust provenance across diverse frameworks and even multiple laboratories. Experimental researchers routinely combine command-line utilities, Python analysis scripts, Excel-tables, instrument controllers, and manual operations on experimental equipment without native computer interfaces, while institutional systems such as electronic lab notebooks (ELNs) and laboratory information management systems (LIMS) must be able to ingest and annotate results. Without a common, lightweight way to describe procedures and their parameters, handoffs and knowledge transfers become error-prone and fragile, and reusing or validating experiments across research groups is challenging.

In materials science and engineering (MSE), the challenges differ substantially from domains where workflow languages originate. Experimental workflows typically include multi-instrument characterization sequences, instrument-dependent procedures, and

manual alignment or calibration steps that cannot be expressed in existing workflow language. Additionally, the present workflow languages do not include explicit methods for including standard operating procedures (SOPs), which are essential in manually executed steps. Hence, research data management (RDM) often breaks down at handoffs: instrument states, operator decisions, and implicit assumptions are rarely recorded, leading to poor reproducibility and limited reuse across laboratories. These pain points motivate lightweight hybrid workflow descriptions.

Universal workflow descriptions can bridge these gaps by making the sequence of computational and manual experimental steps explicit. They capture instrument state and metadata, enabling seamless integration with ELNs and existing workflow engines. This integration paves the way toward more automated materials acceleration platforms, which combine autonomous experimentation and computation in one solution.

Computational workflow engines allow to compose, execute, and track sequences of data processing, orchestration across local workstation computers, high performance computing (HPC) clusters, and cloud computing resources while capturing provenance and enabling reproducibility. A diverse ecosystem of workflow languages and workflow management systems (WfMSs) has emerged, differing in their design goals, target audiences, etc. An overview of WfMSs used in the MSE domain is available in the benchmark of Diercks et al. [2]. They evaluate the WfMSs for their ability to describe, reproduce, and reuse scientific workflows based on user stories and an exemplary GitHub-hosted workflow. They also derive general requirements and compare selected WfMSs to guide researchers in choosing suitable solutions. In the following, we provide a concise, nonexhaustive list of established WfMSs in the MSE domain, as WfMSs have evolved since the study by Diercks et al. [2].

The Common Workflow Language (CWL) [3] is a community-driven, declarative specification that describes portable workflows with file-based communication and tools with explicit inputs, outputs, and runtime requirements to facilitate reproducible execution across multiple WfMSs. The Workflow Description Language (WDL) [4] is a declarative workflow language and execution stack popular in genomics and large-scale data analysis for expressing tasks and scatter/gather patterns. Nextflow [5] and Snakemake [6] are domain-agnostic WfMSs that combine concise, script-like descriptions with strong support for HPC schedulers, container runtimes, and incremental re-execution for file-based workflows; they are widely used in bioinformatics and adaptable to other domains. In the same way, the Galaxy WfMS [7] uses an extensible markup language (XML)-based workflow language to construct file-based programing language independent workflows and gains popularity based on its form-based interface, which simplifies the workflow creation for lab scientists.

Based on these developments, the workflows integrated into the Kadi4Mat [8] ecosystem enable the modeling and execution of scientific processes via a simplified, framework- and language-independent visual programing environment [9, 10]. Within the workflows, the repository component of Kadi4Mat can be directly addressed to store and manage (meta)data and to capture provenance. In the same way, the Chaldene [11] WfMS provides a visual programing interface for workflows directly integrated in the Jupyter notebook environment with a specific focus for image analysis and postprocessing workflows.

A third, complementary direction in addition to the file-based programing-language-independent WfMSs and the visual-programing-ased WfMSs are Python-based WfMSs. Initially developed for high throughput screening, pyiron [12] and jobflow [13] embed workflow capability within computational materials science environments to manage simulations, data models, and analysis tightly coupled to materials databases and tools. These two Python-based WfMSs have recently been extended to be interoperable based on the Python Workflow Definition (PWD) [14] to enable the seamless sharing of workflows between them. Moreover, pyiron-workflow [15] is a mature, widely adopted framework for orchestrating compute-intensive tasks on HPC resources using the executorlib [16] backend. This solution targets the rapid prototyping using Python-based workflows for HPC up to the Exascale and it is compatible to exporting to the PWD [14].

In addition to the workflow developments in the computational materials science community, there is a growing work on automated experimentation toward materials acceleration platforms [17]. Most notably, broker platforms orchestrate access to experimental resources. The fast intention-agnostic learning server (FINALES) exposes experimental data via a standardized API to enable remote, automated orchestration and interoperability across laboratories and autonomous materials-discovery workflows [18]. A community-scale data aggregation initiative is exemplified by the collaboration of partners from different institutions to work on enhanced battery workflows, exploring automation architectures, and implementing closed-loop optimization [19]. These efforts illustrate how coordination, standardized data exchange, and autonomous decision loops can accelerate materials discovery and characterization. However, they typically presuppose instrument connectivity and full automation.

In summary, workflows and their provenance are central to reproducible research because they record the sequence of tasks, parameters, software environment, and outputs; ro-crate [20] provides a practical, standards-based way to capture this information by packaging files together with machine-readable JSON-LD metadata (using schema.org semantics) that describe datasets, software, agents, licenses, and relationships. Commonly [21], provenance is captured retrospectively by embedding the workflow script (or a pointer to it), container or environment descriptors, execution logs, intermediate artifacts, and final results into a single ro-crate bundle, thereby preserving the contextual links between steps, inputs, and outputs and making it easier to share, validate, and reuse the complete provenance record.

These tools collectively advance FAIR principles by standardizing the description of inputs, parameters, and outputs, and by enabling provenance capture, containerized execution, and integration with schedulers and storage backends. However, most are optimized for automated, compute-centric tasks and assume programmatic control of resources. They less frequently address hybrid scenarios, in which manual, instrument-driven experimental steps and human decision points must be coordinated alongside automated steps. Additionally, unnetworked experimental instruments present an ongoing challenge. Bridging this gap requires lightweight, interoperable extensions or abstractions that can represent manual actions, instrument state, and human–machine handoffs, while remaining compatible with existing workflow engines and institutional systems (e.g., ELNs, LIMS).

Designing workflow tools for hybrid computational-experimental research requires interfaces that allow scientists to construct and execute procedures with minimal cognitive effort. The concept of user experience (UX) offers a comprehensive framework for evaluating and improving interaction design, and encompasses the user's entire set of perceptions and responses resulting from the use—or anticipated use—of software [22]. This includes instrumental qualities such as efficiency and error avoidance, but also broader experiential factors, such as trust, perceived clarity, and esthetics. In experimental materials science, where workflows combine manual operations, instrument control, and computational analysis, a positive UX is essential for reliable execution, reduced error rates, and long-term adoption.

A core component of UX is usability [23], defined as the effectiveness, efficiency, and satisfaction with which specified users achieve their goals. Usability refers to observable interaction performance, while UX describes the broader qualitative and emotional interpretations of that performance. Several human–computer interaction (HCI) models inform—the evaluation, including Nielsen's usability heuristics, which provide practical criteria such as system visibility, alignment with real-world conventions, and consistency [24]. The system usability scale (SUS) [25] complements these qualitative principles by offering a standardized tool for quantifying users' subjective usability assessments.

Our contribution does not introduce a new WfMS. Instead, we provide a minimal, interoperable layer that complements existing WfMSs by representing manual steps and instrument-dependent procedures that existing systems cannot express. Current WfMSs assume programmatic control, lacking mechanisms for expressing operator-driven actions, instrument states, or structured manual outputs. Our approach fills this gap by providing a lightweight intermediate representation that enables seamless handoffs between automated and manual steps while preserving provenance.

We employ—tensile testing as a canonical materials-science exemplar to evaluate the execution of hybrid workflows that integrate manual experimental procedures with automated data-analysis steps. For the usability study, which assesses how readily nontechnical scientists can *create and modify* workflow plans, we select—a *nanoindentation* protocol owing to the greater availability of researchers experienced in micromechanical testing. This article is organized to cover design principles, implementation details, UX, use case results, lessons learned, limitations, and future directions.

2 | Requirements and Specification

A workflow provides a top-level, structured description of a multistep process and the alternative execution paths that may be taken. At a finer granularity, functions represent discrete computational steps, whereas SOPs specify human-operated experimental tasks. Functions and SOPs each encapsulate the actions required to transform inputs into outputs and may be parameterized, and reused across workflows. Each function or SOP typically declares explicit inputs and outputs (e.g., files, numerical values, or other data artifacts) and is accompanied by metadata and provenance information—such as parameter settings, execution context, timestamps, version identifiers, and lineage—that support reproducibility, auditing, and traceability.

Human-readability is essential for adoption and maintenance: a common workflow format should be concise and transparent so that advanced users can write, inspect, and debug procedures without needing specialized frameworks. Moreover, domain experts must be able to verify workflows. At the same time, the format must remain machine-actionable and easily embedded into institutional systems—ELNs, LIMS, and repositories—so that procedures, parameters, and provenance flow seamlessly between human-facing documentation and automated execution.

The materials science community comprises both researchers who operate analysis pipelines via graphical user interfaces (GUIs) and developers who implement numerical simulations for execution on HPC systems. Consequently, a universal workflow description must accommodate diverse WfMSs and workflow languages—from Python scripts and services to mobile applications developed with Flutter [26]. To meet these requirements, we emphasize interoperability across workflow languages and WfMSs, portability between execution environments, and machine-actionability. In this study, we adopt pyiron-workflow [15] because it is a mature, widely adopted framework and because workflow specifications can be expressed concisely as compact Python scripts. As such, these scripts can be automatically version-controlled via git version control infrastructure [27].

Each instrument-specific SOP shall specify functional requirements, list the tasks (including stepwise operations and anticipated results), enumerate explicit inputs and outputs, permit configurable parameters with documented default values, and support version control. In this study, SOPs are authored in Markdown because it is human-readable and widely accessible; many laboratory personnel are familiar with Microsoft Word, which can produce Markdown and it integrates readily with common tooling. Configurable parameters and their default values are encoded as key-value pairs in reStructuredText (reST) using "|" triplets. Units are incorporated directly within the Markdown narrative. These parameters also include instrument output: numerical output values may be provided, and the file path for generated outputs can be specified. Allowable ranges and validation rules are omitted in this prototype to reduce complexity for technicians during SOP creation. When maintained within a Git-based workflow, SOPs are inherently version-controlled; each document includes a concise footer containing the SOP version and author. Future developments may extend this metadata to include date, repository link, and other provenance information.

3 | Reference Implementation

We implement—a prototype in Python using the pyiron-workflow layout, supporting explicit chaining of steps (e.g., `step1` ⇒ `step2`) and separate specification of the workflow's initial step. Each node is annotated with `@Workflow.wrap.as_function_node()` to enable integrated workflow services such as structured input/output aggregation and logging. Because the pyiron-workflow package carries numerous external dependencies (for high-performance computing,

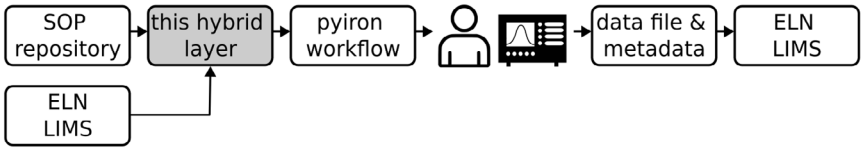

FIGURE 1 | Overview of architecture of hybrid workflow layer: a SOP from a curated repository, and default values from ELNs/LIMs of past experiments are joined in this hybrid layer. The description of the different steps is handed off to the pyiron-workflow engine that then presents the GUI to the scientist operating the instrument. The recorded data file and metadata flow back into the ELN/LIMS, to record provenance.

visualization, etc.) that complicate—environment setup, we also develop—a single-file, minimal engine that implements only output aggregation and logging. This lightweight implementation enables rapid testing and iterative refinement of the prototype and provides practical insights to guide subsequent development. For manual workflow steps, the SOP content is rendered in a GUI. Text fields are provided for entry of required parameters, including data output file paths, and the interface is implemented using the Tkinter GUI toolkit for Python. The data flow is shown in Figure 1.

As this work is a prototype, several features common to production systems are intentionally omitted. Validator components are necessary to verify the integrity of workflow and SOP scripts and to prevent the execution of malicious code. Converter modules enable translation of pyiron-workflow scripts into widely used workflow description formats. GUIs support creation, visualization, and modification of workflows and SOPs; here, we explore—GUI layouts and interaction concepts but do not yet integrate—those interfaces with runtime workflow execution. Likewise, we do not define a formal versioning strategy or mechanisms to ensure backward compatibility, as those considerations fall outside the scope of this study.

4 | Use Case for Workflow Execution

Tensile testing is selected as a pilot because it exhibits several characteristics typical of MSE workflows: multi-instrument characterization (optical microscopy to scanning electron microscopy (SEM)), strong instrument-dependency (fixture configuration, alignment, gauge measurement), manual sample preparation steps, and a clear separation between experimental data acquisition and computational postprocessing. These characteristics make it an ideal exemplar for studying hybrid workflows in which manual, instrument-specific actions are executed.

Specimens are metallographically prepared, and surface quality is verified by light microscopy to ensure a smooth, artifact-free finish prior to testing. The specimen geometry (gauge length and cross-sectional dimensions) is recorded, and tensile loading is conducted using a universal testing machine (UTM). After failure, fracture surfaces are first inspected by light microscopy to obtain a macroscopic overview and are subsequently examined by SEM for higher-resolution characterization. During data analysis, engineering stress and strain are calculated from the recorded load–displacement data using the measured specimen geometry. The elastic (Young's) modulus is determined by applying a linear least-squares fit to the initial, linear portion of the stress–strain curve, using a fitting function implemented in Python. The workflow therefore, includes both manual experimental steps and computational postprocessing.

An example of an SOP is shown in Figure 2. The SOP is represented in Markdown, where parameter definitions are encoded in bulleted lists within their respective sections; nested bullet levels are conveyed by indentation to represent hierarchical relationships. The document concludes with a footer that records SOP metadata and explicitly indicates that a file must be specified during manual execution of the step. In the GUI, the SOP is displayed with interactive controls for entering the required parameters and for selecting the results file.

The GUI displays the measurement curves using Python's matplotlib [28] library and presents the associated log file (see Figure 3). The figure view supports interactive zooming and panning, while the log records timestamps, acquisition parameters, and the identifiers of generated data files. These records are

(a)

```
# Tensile testing with Doli
...
- Button Bike "Travel order"
 - distance-controlled, relative:
 - direction clear
- Measure the sample:
 - |width||
 - |thickness||
- Mount sample:  top first and then
- Button Setup experiment:  Hammer
...
Author:  Steffen Brinckmann, IMD-1
     <!-- version:1.0 |filename|| ->
```

(b)

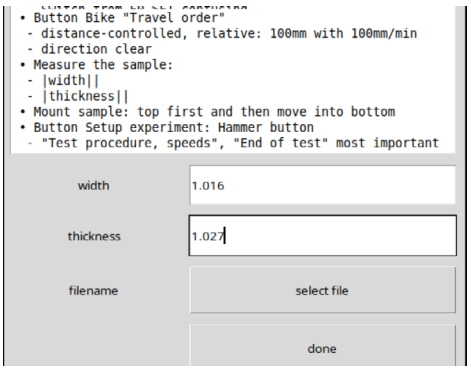

FIGURE 2 | Standard operating procedures. (a) Markdown-style specification: a headline supplies the human-readable procedure label; unordered bullets enumerate discrete tasks; parameter values are delimited by vertical bars (|); the footer indicates that an associated measurement file must be uploaded. (b) GUI rendering for a tensile test: editable text fields capture specimen geometry and test parameters, and a file-upload control permits attachment of the corresponding measurement data.

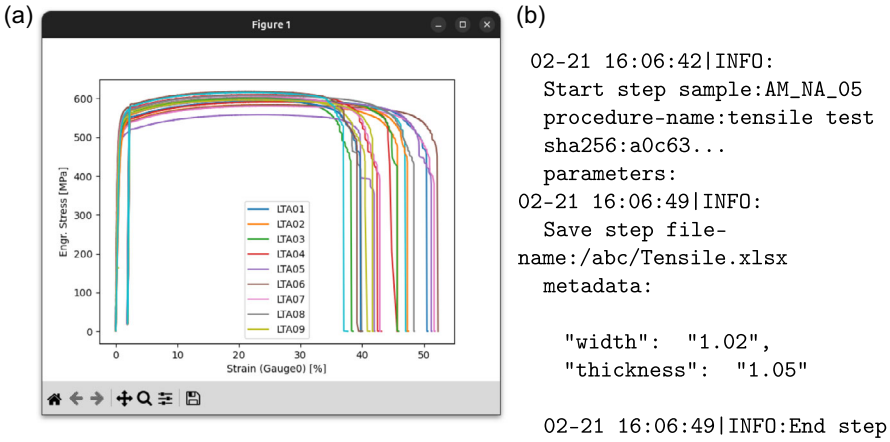

FIGURE 3 | Prototype results. (a) Computed stress–strain curves are displayed as interactive matplotlib plots enabling region-specific zooming and saving of results. (b) An execution log records the timestamp, experimental metadata, and the locations of relevant input and output files.

intended to be incorporated into ELNs to ensure preservation of provenance metadata.

5 | User Experience

We investigate—the design of a workflow interface that reflects experimental steps typical of materials science. Participants are instructed to construct a workflow comprising sample preparation, calibration measurement, primary measurement, and data analysis. For the user-experience evaluation, we use—nanoindentation experiments employing the Oliver–Pharr method [29], given the greater availability of personnel for micromechanical studies. To support this task, we develop—three GUI prototypes using Figma [30] (see Figure 4). The first prototype implements a compact two-pane layout: the left pane presents a sequential list of steps with step-specific properties integrated into the list view, while the right pane displays detailed information for the selected step. The second prototype, inspired by the galaxy workflow platform [7], uses four horizontally arranged panes: a step-selection list, a procedure view, a parameter panel, and a sequential-step overview. The third prototype provides an empty canvas on which participants construct—a graph of rectangular nodes that are manually linked; selecting a node opens a pane showing the procedure, while key parameters are displayed directly on the node.

Using a combination of think-aloud observations, task-based interactions, and the SUS questionnaires, the study examines how effectively and intuitively participants can construct workflows, locate required functions, understand parameter dependencies, and maintain a coherent mental model of the sequence of scientific activities.

The number of participants in this study is limited, as recruiting experimental materials scientists for structured UX evaluations is unexpectedly challenging. Even so, the sample size is adequate for the goals of this design stage. Nielsen and Laundauer's empirical model of usability problem discovery [31] demonstrates that a sample of approximately five participants is sufficient to identify the majority of usability issues within a single design iteration, due to the sharply diminishing returns of additional testers. This study uses six participants and the results reflect—the results of Nielsen and Landauer: despite the small group, participants reveal—consistent usability challenges and clear preferences among the prototypes. While larger samples would be required for broader generalization or later validation phases, the number of expert users involved here is sufficient to identify major issues and inform the next iteration of the interface.

Across all collected results, as shown in Figure 5, the galaxy prototype emerges as the most usable and best-received interface. Participants consistently describe this representation as intuitive and well-aligned with how they conceptualize laboratory processes. The SUS ratings reflect this impression: this prototype receives the highest average score by 11 points above the other prototypes—and no participant rates it lowest. Even participants who give relatively modest numerical scores verbally identify the galaxy prototype as their preferred solution. These findings indicate that straightforward, linear representations provide strong

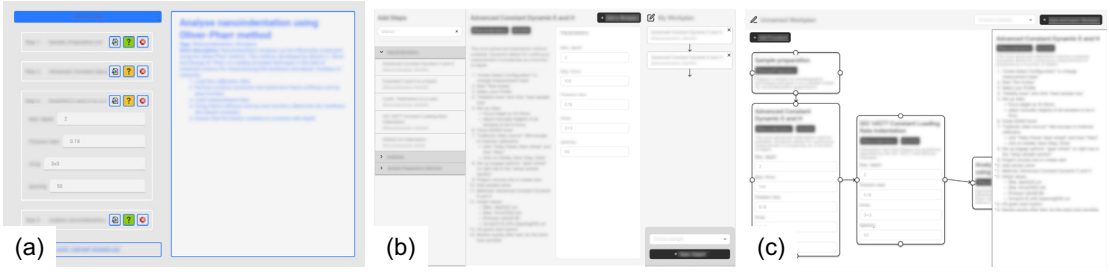

FIGURE 4 | Schematic representations of the three blurred prototypes: (a) compact layout showing list of steps and procedure, used from a previous implementation. (b) Prototype inspired by galaxy-workflow [7], consisting of panes for step selection, task description, parameter definition and step list. (c) Prototype showing the graph prototype.

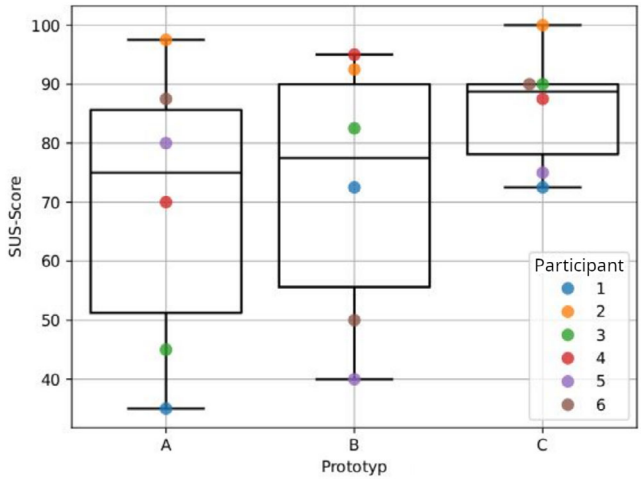

FIGURE 5 | The prototypes are assessed post-test using the system usability scale (SUS), with scores ranging from 1 (lowest) to 5 (highest). For each prototype, the mean SUS score and its standard deviation are computed across all participants. Error boxes indicate the mean ± standard deviation.

mental scaffolding for scientists performing sequential experimental tasks.

The compact prototype is valued for its compactness and orderly appearance, but suffers from difficulties in discoverability. For example, users frequently overlook—the button required to edit parameters, suggesting mismatches between iconography and user expectations. Although participants find it easy to understand once functions are located, such hidden features reduce—both usability and trust.

While participants appreciate—the freedom of the graph prototype in principle, in practice it imposes substantial cognitive load. Several users attempt—to create semantic input–output connections, even though the prototype only supports directional arrows for ordering. This mismatch between expectations and system behavior, combined with unclear error messages, contributes to perceptions of complexity and unnecessary effort. The design appears better suited for visualizing completed workflows or data-flow structures rather than authoring strictly sequential experimental procedures.

The study also reveals a consistent preference for a particular task flow: participants tend- to fill in parameters before adding a step to the workflow, a sequence supported naturally in the galaxy prototype. This finding highlights the importance of aligning interface behavior with users' established work patterns, especially in complex laboratory environments. Furthermore, several missing or incomplete features—such as the inability to edit procedures, open saved workflows, assign different samples to different measurements, or reuse workflows as modular components—directly reduce usability and thus UX. Participants additionally express—a need for clearer visual feedback and more informative error messages.

6 | Discussion

We define—the requirements for a hybrid data and workflow specification for manual experiments, data analysis, and computational simulations in materials science. The specification must be transparent to facilitate human readability and domain-expert verification, while remaining machine-actionable to enable integration with ELNs and LIMS. Our long-term objective is bidirectional interoperability in which input parameters and procedural steps for both computational and manual tasks are automatically populated from ELNs/LIMS and the resulting outputs—including structured results and metadata—are propagated back into these systems to preserve provenance. Finally, the specification must be platform-agnostic, supporting deployment across heterogeneous environments ranging from high-performance computing to mobile clients.

To investigate these requirements, we construct—a prototypical implementation following the "development by prototyping" paradigm [32]. We formalize—a workflow schema and an SOP specification to evaluate requirement compliance and to identify any overlooked needs. The prototype comprises reusable, parameterized low-level analysis functions and SOPs, each defined by explicit inputs, outputs, and provenance metadata to ensure reproducibility and traceability. Implementation is performed using the pyiron-workflow framework.

The prototype implementation consists of a lightweight execution engine and a Tkinter-based GUI to support manual protocol steps. This configuration is adequate for assessing HCI and the effects of manual interventions on workflow execution during experimental procedures.

The results of UX for workflow creation underscore the central role of UX in supporting hybrid workflows. A system that communicates structure clearly, reveals parameters at appropriate moments, and avoids unnecessary complexity allows scientists to focus on their experimental intentions. A successful interface should combine the intuitive sequential clarity of the galaxy prototype with selected strengths of the other prototypes, such as the compact prototype's small footprint and the graph prototype's richer representation of inputs and outputs, where appropriate. Enhancing parameter visibility, enabling modular reuse of workflows, and improving error communication will further strengthen both usability and overall UX.

These findings also reinforce the value of iterative, human-centered design methods [22]. By incorporating feedback from representative user groups early and repeatedly, workflow tools can evolve along with the actual practices and expectations of laboratory researchers. As hybrid computational-experimental WfMSs grow, maintaining a strong UX foundation will be essential for ensuring that workflow descriptions remain not only machine-actionable and interoperable but genuinely supportive of scientific thinking and practices.

This study intentionally omits production-grade features—such as data validators, format converters, and a formal versioning strategy. These capabilities can be readily incorporated in subsequent implementations, as a broad spectrum of well-established technical solutions exists.

Validation of manually entered experimental parameters is important but often impractical. Although validation can be implemented by enforcing allowable value ranges or enumerated lists, such constraints are frequently study-specific (for example, expected widths of 1–2 mm in one context versus 10–20 mm in another), limiting their generalizability. Moreover, other attempts even define the datatype (integer vs. float). A more

practical strategy is post hoc outlier detection, wherein new entries are compared against existing datasets or cohort statistics to identify anomalous values. Requiring explicit parameter ranges or valid-item lists also increases the workload for creating SOPs and can impair usability. Consistent with observations from this study (UX and SOP creation), reducing user burden is essential for adoption and effective use of any RDM solution.

Prototyping reveals additional requirements and blind spots. In some cases, users require the ability to add optional annotations or comments to individual steps to capture unforeseen events (for example, instrument malfunction). This capability can be accommodated by appending an optional "|comment|" field to each SOP step, allowing structured recording of ad hoc observations without interrupting normal execution.

Another requirement is support for pausing or aborting a workflow when an unresolved condition occurs. A pragmatic implementation is for SOP step executions to return explicit error codes or status flags that the workflow engine interprets to pause, abort, or escalate execution as appropriate. This approach provides deterministic handling of exceptional events.

In this study, we employ—the pyiron-workflow script schema (version 0.11). While this representation is sufficient for exploratory development, production deployment requires a semantically stable, well-documented format; notably, the pyiron-workflow specification has undergone substantive revision by version 0.15. For production use, the workflow format should provide compatibility guarantees and natively support machine-readable validators and automated translators to ensure reliable interoperability with other workflow languages.

This study does not resolve the intrinsic ambiguity inherent to workflow languages, an ambiguity that impedes the standardization of both manual and computational procedures. There is no single, universally accepted definition of a "step." For manual operations, we adopt an ad hoc criterion: the smallest unit that is scientifically meaningful to execute independently as an atomic operation and that may depend on preceding units (see [10]). In the context of SEM, representative steps include "secondary/primary electron imaging", "electron backscatter diffraction (EBSD)", and "energy-dispersive X-ray spectroscopy (EDS)"; the latter two require prior execution of secondary/primary electron setup for locating features. Automating the enforcement of such dependencies would require a more expressive formalism for output types and system states—specifically, explicit output data type definitions and state-transition semantics—which can be incorporated in future versions.

Below we provide a bullet-point list of elements that are critical for future MSE RDM systems:

- explicit recording of instrument state (configuration, calibration status, environment),
- structured representation of manual operations and operator decisions,
- propagation of parameters across characterization modalities,
- integrated provenance for both computational and experimental components.

Reflecting on the study presented, we identify a number of aspects that are unexpectedly difficult:

- parameter validation due to highly variable experimental ranges,
- mismatches between user expectations and workflow-language semantics,
- UX challenges around clarity, feedback, and cognitive load,
- capturing ad hoc events (e.g. instrument malfunction) in structured but lightweight ways.

As lessons learned, we recommended the following community practices:

- authoring SOPs in version-controlled text formats,
- treating manual steps as first-class workflow nodes,
- modular reuse of parameterized experimental procedures.

The proposed procedural layer is intentionally lightweight so that it can be embedded into community platforms such as NFDI-MatWerk. Because it is text-based, versionable, it can scale from single-lab adoption to multi-institution infrastructures. In a community-wide deployment, SOP repositories can be managed similarly to code libraries, enabling validation, sharing, and reuse of experimental procedures. Future integration with NFDI-MatWerk tools would allow automatic ingestion of workflow parameters, instrument metadata, and provenance into central repositories, while supporting export to common workflow standards.

7 | Conclusions

We present a prototype framework that integrates automated computation and analysis with manual experimentation by layering a lightweight, parameterized procedure description over an established WfMS. This study primarily targets experimental scientists who perform data analysis for postprocessing. It is demonstrated in a tensile-testing use case and is evaluated in a usability study with lab scientists; this design-by-prototyping approach highlights parameter propagation, improved provenance and streamlined orchestration, while also revealing current limitations and motivating concrete directions for future work.

The UX study indicates that meaningful evaluations are feasible even with a small number of participants, and that ease of use is critical for lab scientists.

Using a markdown format with reStructured-style parameters proves sufficient for defining parameters with scientific units and semantic Persistent Uniform Resource Locator (PURL) links. Its flexibility also enables unanticipated features, such as adding free-form comments.

For production-ready solutions, the workflow format should offer long-term compatibility guarantees, along with automated validators and translators to other workflow languages.

Author Contributions

Steffen Brinckmann: conceptualization (lead), software (equal). **Sarath Menon**: methodology (equal), software (equal). **Raphael Röske**: methodology (equal), software (equal). **Johannes Steinhülb**: methodology (equal), software (equal). **Michael Selzer**: software (equal), supervision

(equal). **Jan Janssen**: methodology (equal), writing – original draft (equal). **Tilmann Hickel**: supervision (equal). **Ruth Schwaiger**: supervision (equal), writing – review & editing (lead). **Jörg Neugebauer**: supervision (equal).

Acknowledgments

The authors acknowledge Pavlina Kruzikova (BAM) for her contributions to the organization and coordination of this project. This work is part of the consortium NFDI-MatWerk, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the National Research Data Infrastructure—NFDI 38/1—project number 460247524. The authors thank the Federal Government and the Heads of Government of the Länder, as well as the Joint Science Conference (GWK), for their funding and support within the framework of the NFDI4Ing consortium. Funded by the German Research Foundation (DFG)—project number 442146713. J. J., T. H. and J. N. acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG) through the CRC1394 "Structural and Chemical Atomic Complexity—From Defect Phase Diagrams to Material Properties", project ID 409476157. This publication is supported by the Helmholtz Metadata Collaboration (HMC), an incubator-platform of the Helmholtz Association within the framework of the Information and Data Science strategic initiative, within the MetaSurf project.

Open Access funding enabled and organized by Projekt DEAL.

Funding

This work was supported by Deutsche Forschungsgemeinschaft (460247524, 442146713 and 409476157) and Helmholtz-Gemeinschaft (Helmholtz Metadata Collaboration (HMC)).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in A Lightweight Procedural Layer for Hybrid Experimental–Computational Workflows in Materials Science at https://doi.org/10.5281/zenodo.17854403, reference number 17854403.

References

1. M. D. Wilkinson, M. Dumontier, I. J. J. Aalbersberg, et al., "The Fair Guiding Principles for Scientific Data Management and Stewardship," *Scientific Data* 3 (2016): 160018.

2. P. Diercks, D. Gläser, O. Lünsdorf, M. Selzer, B. Flemisch, and J. F. Unger, *Evaluation of Tools for Describing, Reproducing and Reusing Scientific Workflows*, arXiv preprint, arXiv:2211.06429, 2022, https://doi.org/10.48550/arXiv.2211.06429.

3. M. R. Crusoe, S. Abeln, A. Iosup, et al., "Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language," *Communications of the ACM* 65, no. 6 (2022): 54–63, https://doi.org/10.1145/3486897.

4. K. Voss, G. Van der Auwera, and J. Gentry, *Full-Stack Genomics Pipelining with GATK4 + WDL + Cromwell*, (2017), https://doi.org/10.7490/f1000research.1114634.1, 6 (ISCB Comm J): 1381 (slides).

5. P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow Enables Reproducible Computational Workflows," *Nature Biotechnology* 35, no. 4 (2017): 316–319, https://doi.org/10.1038/nbt.3820.

6. F. Mölder, K. P. Jablonski, B. Letcher, et al., "Sustainable Data Analysis with Snakemake," *F1000Research* 10 (2021): 33, https://doi.org/10.12688/f1000research.29032.3.

7. The Galaxy Community, "The Galaxy Platform for Accessible, Reproducible, and Collaborative Data Analyses: 2024 Update," *Nucleic Acids Research* 52, no. W1 (2024): W83–W94, https://doi.org/10.1093/nar/gkae410.

8. N. Brandt, L. Griem, C. Herrmann, et al., "Kadi4Mat: A Research Data Infrastructure for Materials Science," *Data Science Journal* 20 (2021), https://doi.org/10.5334/dsj-2021-008.

9. R. Al-Salman, C. A. Teixeira, P. Zschumme, et al., "Kadistudio use-Case Workflow: Automation of Data-Processing for In Situ Micropillar Compression Tests," *Data Science Journal* 22 (2023), https://doi.org/10.5334/dsj-2023-021.

10. L. Griem, P. Zschumme, M. Laqua, et al., "Kadistudio: Fair Modelling of Scientific Research Processes," *Data Science Journal* 21 (2022), https://doi.org/10.5334/dsj-2022-016.

11. F. Chen, P. Slusallek, M. Müller, and T. Dahmen, "Chaldene: Towards Visual Programming Image Processing in Jupyter Notebooks," in *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (IEEE, 2022), 1–3, https://doi.org/10.1109/VL/HCC53370.2022.9832910.

12. J. Janssen, S. Surendralal, Y. Lysogorskiy, et al., "pyiron: An Integrated Development Environment for Computational Materials Science," *Computational Materials Science* 163 (2019): 24–36, https://doi.org/10.1016/j.commatsci.2018.07.043.

13. A. S. Rosen, M. Gallant, J. George, et al., "Jobflow: Computational Workflows Made Simple," *Journal of Open Source Software* 9, no. 93 (2024): 5995, https://doi.org/10.21105/joss.05995.

14. J. Janssen, J. George, J. Geiger, et al., "A Python Workflow Definition for Computational Materials Design," *Digital Discovery* 4 (2025): 3149–3161, https://doi.org/10.1039/D5DD00231A.

15. pyironWorkflow, *pyiron_workflow—Graph-and-Node Based Workflows*, (2025), accessed November 10, 2025, https://github.com/pyiron/pyiron_workflow.

16. J. Janssen, M. G. Taylor, P. Yang, J. Neugebauer, and D. Perez, "Executorlib—Up-Scaling Python Workflows for Hierarchical Heterogenous High-Performance Computing," *Journal of Open Source Software* 10, no. 108 (2025): 7782, https://doi.org/10.21105/joss.07782.

17. S. P. Stier, C. Kreisbeck, H. Ihssen, et al., "Materials Acceleration Platforms (MAPs): Accelerating Materials Research and Development to Meet Urgent Societal Challenges," *Advanced Materials* 36, no. 45 (2024): e2407791, https://doi.org/10.1002/adma.202407791 doi:.

18. M. Vogler, J. Busk, H. Hajiyani, et al., "Brokering between Tenants for an International Materials Acceleration Platform," *Matter* 6, no. 9 (2023): 2647–2665, https://doi.org/10.1016/j.matt.2023.07.016.

19. M. Stricker, L. Banko, N. Sarazin, et al., "Computationally Accelerated Experimental Materials Characterization—Drawing Inspiration from High-Throughput Simulation Workflows," (2025), https://arxiv.org/abs/2212.04804.

20. S. Soiland-Reyes, P. Sefton, M. Crosas, et al., "Packaging Research Artefacts with Ro-Crate," *Data Science* 5, no. 2 (2022), https://doi.org/10.3233/DS-210053.

21. S. Leo, M. R. Crusoe, L. Rodríguez-Navas, et al., "Recording Provenance of Workflow Runs with Ro-Crate," *PLoS ONE* 19, no. 9 (2024): e0309210, https://doi.org/10.1371/journal.pone.0309210.

22. ISO (the International Organization for Standardization), "Iso 9241-210: 2019, Ergonomics of Human-System Interaction—Part 210: Human-Centred Design for Interactive Systems," Technical report (International Organization for Standardization, 2019), https://www.iso.org/standard/77520.html, Edition 2.

23. ISO (the International Organization for Standardization), "ISO 9241-11: 2018, Ergonomics of Human-System Interaction—Part 11: Usability: Definitions and Concepts," Technical report (International

Organization for Standardization, 2018), https://www.iso.org/standard/63500.html.

24. J. Nielsen, "Enhancing the Explanatory Power of Usability Heuristics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94 (Association for Computing Machinery, 1994), 152–158, ISBN 0897916506, https://doi.org/10.1145/191666.191729.

25. J. Brooke, *SUS—A Quick and Dirty usability Scale* (Taylor & Francis, 1996), 189–194.

26. Flutter, "Flutter: Build Apps for Any Screen," (2025), accessed November 10, 2025, https://flutter.dev.

27. git, "Git—Fast, Scalable, Distributed Revision Control System," (2025), accessed November 10, 2025, https://git-scm.com.

28. J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering* 9, no. 3 (2007): 90–95, https://doi.org/10.1109/MCSE.2007.55.

29. G. M. Pharr and W. C. Oliver, "Measurement of Thin Film Mechanical Properties Using Nanoindentation," *MRS Bulletin* 17, no. 7 (1992): 28–33.

30. Figma, "Figma—The Collaborative Interface Design Tool," (2025), accessed November 10, 2025, https://www.figma.com/.

31. J. Nielsen and T. K. Landauer, "A Mathematical Model of the Finding of usability Problems," in *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems*, CHI '93 (Association for Computing Machinery, 1993), 206–213, ISBN 0897915755, https://doi.org/10.1145/169059.169166.

32. B. Camburn, V. Viswanathan, J. Linsey, et al., "Design Prototyping Methods: State of the Art in Strategies, Techniques, and Guidelines," *Design Science* 3 (2017): e13.