FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

An Intelligent Platform for Reproducibility and Data Management with an Hybrid User Interface

Anu Moses Atolagbe

Mestrado em Engenharia Informática

Supervisors: Jácome Cunha
Lázaro Costa

July 21, 2026

An Intelligent Platform for Reproducibility and Data Management with an Hybrid User Interface

Anu Moses Atolagbe

Mestrado em Engenharia Informática

Approved in oral examination by the committee:

President: Prof. José Campos

Referee: Prof. Miguel Goulão

Referee: Prof. Lázaro Costa

July 21, 2026

Resumo

A investigação computacional tornou-se uma das principais formas de produzir e partilhar conhecimento científico, e com ela cresceu a exigência de que outros investigadores consigam reproduzir as experiências publicadas. Embora a captura de ambientes de software esteja hoje bem resolvida através da contentorização, duas dificuldades permanecem por tratar de forma integrada. A gestão de conjuntos de dados de investigação tornou-se cada vez mais difícil à medida que estes crescem em dimensão, complexidade e importância nos fluxos de trabalho computacionais. Em particular, a escrita de metadados úteis e a divisão de conjuntos de dados que excedem os limites dos repositórios continuam a ser tarefas manuais que as ferramentas de reprodutibilidade não acompanham. Ao mesmo tempo, muitos dos investigadores que necessitam destas ferramentas têm formação numa área científica e não em engenharia de software, o que torna as ferramentas existentes difíceis de utilizar.

Esta dissertação apresenta o SCISUITE, uma extensão da ferramenta de reprodutibilidade conversacional SCICONV, construída sobre o motor SCIREP. A plataforma acrescenta um modo autónomo de gestão de dados, geração de metadados assistida por um modelo de linguagem, divisão determinística de grandes conjuntos de dados com ligação cruzada entre os registos resultantes, e uma interface híbrida que combina uma superfície conversacional com formulários estruturados e controlos explícitos. Desta forma, a plataforma reúne, num único fluxo de trabalho, o empacotamento reprodutível e a publicação de dados, reduzindo a barreira técnica através da conversação sem retirar ao investigador o controlo sobre o resultado estruturado que a conversação, por si só, trata de forma deficiente.

A plataforma foi avaliada num estudo com utilizadores, de desenho intra-sujeitos, com oito participantes. Cada participante publicou um conjunto de dados de grande dimensão tanto com o modo de gestão de dados do SCISUITE como manualmente através da interface Web do Zenodo, utilizando a *System Usability Scale* e o *NASA Task Load Index* como instrumentos principais. O SCISUITE obteve melhores resultados do que o processo manual para todos os participantes que completaram as duas condições, com uma pontuação SUS média de 95,0 contra 52,9 e uma carga de trabalho média de 15,8 contra 63,7. A diferença foi estatisticamente significativa em ambas as medidas segundo o teste bilateral de Wilcoxon ($p = 0{,}0156$). Os participantes completaram a tarefa em cerca de um terço do tempo, aceitaram os metadados gerados praticamente sem alterações, e os registos produzidos pela plataforma revelaram-se mais completos e melhor interligados do que os produzidos manualmente. Embora a amostra seja reduzida, os resultados apontam de forma consistente para o valor de combinar assistência conversacional com controlo estruturado na publicação de dados de investigação.

Abstract

Computational research has become one of the main ways scientific knowledge is produced and shared, and the expectation that other researchers should be able to reproduce published experiments has grown with it. While the capture of software environments is now well handled through containerisation, two difficulties remain unaddressed in an integrated way. The management of research datasets has become increasingly difficult as datasets grow in size, complexity, and importance within computational workflows. In particular, writing useful metadata and splitting datasets that exceed repository size limits remain manual tasks that reproducibility tools do not capture. At the same time, many of the researchers who need these tools are trained in a scientific domain rather than in software engineering, which makes existing tools difficult to approach.

This dissertation presents SCISUITE, an extension of the conversational reproducibility tool SCICONV, built on the SCIREP engine. The platform adds a standalone data management mode, language-model-assisted metadata generation, deterministic chunking of datasets with cross-record linking, and a hybrid interface that combines a conversational surface with structured forms and explicit controls. In this way, the platform brings reproducibility packaging and data publication into a single workflow, lowering the technical barrier through conversation while keeping the researcher in control of the structured output that conversation alone handles poorly.

The platform was evaluated in a within-subjects user study with eight participants, in which each participant published a large dataset both with SCISUITE's data management mode and manually through the Zenodo Web interface, using the System Usability Scale and the NASA Task Load Index as the primary instruments. SCISUITE outperformed the manual workflow for every participant who completed both conditions, with a mean SUS score of 95.0 against 52.9 and a mean workload of 15.8 against 63.7. The difference was statistically significant on both measures under the two-sided Wilcoxon signed-rank test ($p = 0.0156$). Participants completed the task in roughly a third of the time, accepted the generated metadata almost without change, and the records the platform produced were more complete and better linked than those produced by hand. Although the sample is modest, the results point consistently to the value of combining conversational assistance with structured control for research data publication.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a more sustainable, equitable, and prosperous future. They define 17 interconnected goals[1] that address major global challenges including education, innovation, infrastructure, sustainability, and institutional effectiveness.

This dissertation contributes to these goals through the development of SCISUITE, a platform that supports computational reproducibility, research data publication, FAIR-aligned data management, and language-model-assisted metadata generation. By reducing technical barriers to reproducible research and improving the accessibility of research data management workflows, the platform promotes more transparent, reusable, and sustainable scientific practices.

The specific Sustainable Development Goals addressed by this work are:

SDG 4 Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.

SDG 16 Promote peaceful and inclusive societies, provide access to justice for all, and build effective, accountable and inclusive institutions.

[1] https://sdgs.un.org/goals

SDG	Target	Contribution	Performance Indicators and Metrics
4	4.4	By simplifying reproducibility packaging and dataset publication, SciSuite reduces technical barriers faced by students and researchers when sharing and reusing scientific artefacts.	Task completion rate; usability scores (SUS); successful publication of datasets.
9	9.5	SciSuite strengthens digital research infrastructure through integrated reproducibility workflows, FAIR-oriented data management, metadata generation, and large-dataset publication.	Metadata acceptance rate; dataset publication success rate; dataset reconstruction success rate.
16	16.6	The platform promotes transparency, accountability, and trust in computational research by enabling verifiable publication of datasets and reproducible artefacts.	Verification success rate; metadata completeness; reproducibility success rate.

Acknowledgements

I would first like to thank my supervisors, Professor Jácome Cunha and Professor Lázaro Costa, for their guidance throughout this work. Their feedback shaped the direction of the project at several important points, and their patience and availability made the process of writing this dissertation considerably easier than it would otherwise have been.

I am grateful to the Faculty of Engineering of the University of Porto for providing the environment and resources that made this work possible. I would also like to thank the participants who took part in the user study, who gave their time and attention so that the platform could be evaluated.

Finally, I thank my family and friends for their constant support and encouragement during my studies. Their belief in me carried me through the more difficult stretches of this work, and I could not have completed it without them.

Anu Moses Atolagbe

Generative AI Usage Disclosure

In accordance with institutional and European research integrity guidelines, the author discloses the limited use of generative artificial intelligence tools during the preparation of this dissertation. Generative AI tools were not used to generate or contribute to the core research ideas, system architecture, design rationale, experimental methodology, data analysis, or scientific argumentation presented in this work. All conceptual, technical, and analytical contributions are solely the responsibility of the author.
Generative AI systems were used for:

- Minor grammar and readability revisions of the manuscript text; and
- Language polishing to improve clarity and conciseness without altering the intended meaning or technical content.

No generative AI tools were used to conduct experiments, generate results, design evaluation protocols, or make research decisions. The author takes full responsibility for the accuracy, originality, and integrity of the work presented in this dissertation.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 2

API Application Programming Interface. 22, 30, 37, 41, 42, 48

DOI Digital Object Identifier. 8, 15, 23, 24, 26, 27, 29, 41, 43, 59

FAIR Findable, Accessible, Interoperable, and Reusable. 2, 8, 9, 15–17, 20, 57

FEUP Faculdade de Engenharia da Universidade do Porto. 45

HTTP HyperText Transfer Protocol. 30

NASA-TLX NASA Task Load Index. 44, 46, 47, 51

SUS System Usability Scale. 43, 44, 46, 47, 51

Chapter 1

Introduction

Computational research has become one of the main ways that scientific knowledge is produced and shared. In fields as different as climate science, genomics, social media analysis, and machine learning, researchers run experiments that combine code, software libraries, and data into pipelines. As this style of work has grown, so has the expectation that other researchers should be able to take those pipelines and run them again. This matters both for confirming published results and for extending the work to new questions.

Reproducibility is the term commonly used for this expectation. Peng [46] describes it as the situation where another researcher takes the same code and data as the original study and arrives at the same result. This short formulation is a useful starting point, but a complete definition must also state what is held fixed when the experiment is re-executed; Section 2.1 gives that definition and fixes the terminology this dissertation uses. The concept is simple to state, but in practice it has been one of the most discussed problems in modern science. Large surveys have reported that a substantial portion of computational findings cannot be independently reproduced [2], and a wide body of work has attempted to explain why [26, 52, 57]. The reasons usually fall into three areas: code that depends on a software environment that no longer exists, data that is too large or too sensitive to share alongside the code, and human steps that are not recorded anywhere because the original researcher carried them out manually.

1.1 Context

The reproducibility problem in computational science has been studied for more than a decade, and the broader issues are by now well documented. What is discussed less often is how the nature of the problem has shifted as datasets have grown and as research software has become more specialised. In the early 2010s, most reproducibility work was concerned with freezing code and the operating system around it. Once container technology became widely available, environment capture stopped being the central bottleneck. The difficulty then moved one step outward, to the data that the code consumes and produces, and to the people who are expected to use the resulting tools.

On the software side, considerable progress has been made. Containerisation tools such as Docker [41] and Singularity [36] allow a complete software environment to be captured as a portable image, which removes most failures caused by missing libraries and version mismatches. Building on this, several frameworks have attempted to make the packaging of reproducible experiments easier, including ReproZip [12], Binder [35], Code Ocean [13], Whole Tale [11], RenkuLab,[1] and more recently SciRep [14] and SciConv [15]. Most of these systems are reviewed in detail in Chapter 3. Taken together, they have substantially reduced the challenges associated with capturing software environments in a portable form.

On the data side, several pressures push in the same direction. Sensor networks, sequencing instruments, and simulation codes generate data at rates that grow faster than the artefact sizes most repositories will accept [10]. Funding agencies and journals have started to require that published work include both code and data, in line with the Findable, Accessible, Interoperable, and Reusable (FAIR) principles [64], which forces researchers to deposit their datasets somewhere persistent. At the same time, ethical and legal rules around clinical, financial, or personal data make it impossible to bundle the data with the code at all [51]. Repositories such as Zenodo[2] apply per-record size limits that force researchers to split large datasets across multiple deposits, and reconstruct them at execution time. These trends together mean that data must increasingly live outside the reproducible artefact, while the artefact still has to find that data again and verify what was used.

A third pressure receives less attention in the reproducibility literature, although it shapes adoption of these tools in practice. The people doing computational research are not always trained to work with the tools that reproducibility frameworks expose. A geographer studying rainfall patterns, a sociologist analysing survey responses, or a doctor running a regression on hospital data may not know what a Dockerfile is, let alone how to write one. Recent work in human-computer interaction has produced guidelines for designing systems that integrate Artificial Intelligence (AI) assistance with explicit user controls [1], and empirical comparisons have shown that conversational and graphical modalities have complementary strengths for different task types [23]. Large language models are also beginning to appear inside scientific workflows themselves, including in autonomous research systems [6]. These developments suggest that reproducibility tools can be made more accessible to non-specialists without giving up on the technical precision they require.

Taken together, the state of the field shows three trends that are not yet handled by a single tool. Software environment reproducibility has been substantially improved through containerisation technologies. Large-scale data management remains a manual task layered on top. And the population of researchers who need these tools is growing wider than the population that can comfortably write configuration files. The work in this dissertation responds to those three trends.

[1] https://renkulab.io
[2] https://zenodo.org

1.2 Motivation

The work leading to this thesis began as a narrow extension to SciConv, the conversational reproducibility tool developed by Costa, Barbosa, and Cunha [15]. The original plan was to add support for datasets too large to embed in the reproducibility artefact, by chunking them deterministically and uploading the parts to Zenodo. During design and implementation, two observations broadened the scope of the work.

The first came from supervisory discussions during the early design phase. Researchers who wanted to publish a dataset on its own, without packaging an experiment around it, had to use a separate set of tools and received little help with the metadata. Many produced repository entries with thin descriptions and missing keywords, which then made the data harder for others to find and reuse. Reusing the chunking and language-model-driven metadata inference already being developed, but exposing them through a standalone data publication workflow, would let the platform address this gap directly.

The second came from watching the existing chat interface in use. A conversational interface is well suited to expressing intent, but it is poor for inspecting and correcting structured output [53], and forces users into open-ended prompt formulation that can be cognitively demanding [18]. When SciConv inferred metadata for a dataset, the researcher had to either accept the result or describe in prose what should change. The interface therefore needed to combine free-form conversation with structured controls that allow direct correction of inferred output.

These two design decisions, the addition of a standalone data management mode and the move to a hybrid interaction model, together define the scope of the work that follows. The result is no longer a narrow extension that solves one packaging problem, but a redesign of SciConv as a unified platform for reproducibility and data management.

1.3 Problem Definition

The problem that this dissertation addresses can be stated in three parts.

First, reproducibility tools and data management tools are usually separate. A researcher who has finished an experiment typically uses one tool to package the code, another to upload the dataset, and often a third to write the metadata that describes it. No single workflow takes the researcher from a folder on their machine to a reproducible artefact and a citable dataset that are correctly linked to one another.

Second, when datasets are too large to live inside a reproducibility artefact, the work of uploading them, splitting them to fit repository limits, and writing their metadata falls on the researcher. None of this work is captured by the reproducibility tool. If the researcher leaves the project, the recipe for reconstructing the dataset goes with them.

Third, this technical work is being asked of researchers who, in many cases, were trained in a domain that has nothing to do with software engineering. A platform that requires them to write

Dockerfiles, choose chunk sizes, or compose metadata records by hand will be used reluctantly, if at all.

These three issues can be expressed as the following research problem.

Research Problem: *To what extent can a language-model-assisted platform that combines conversational and structured interaction support reproducibility packaging and large-scale dataset publication for researchers without software engineering expertise?*

1.4 Research Questions

From the research problem above, the work derives one main research question together with three sub-questions, each tied to a measurable aspect of the platform.

> *RQ. Can a hybrid platform combining conversational and structured interaction deliver reproducibility packaging and standalone research data management with quality comparable to that of using dedicated tools for each task?*

The sub-questions look at the three contributions of the platform in turn.

> *RQa. To what extent can a large language model produce repository metadata for a dataset that researchers find acceptable without major edits?*
>
> *RQb. How effectively can researchers complete a data publication task using the platform's hybrid interface, compared with the standard manual workflow on the same repository?*
>
> *RQc. When a dataset exceeds repository size limits, can deterministic chunking and cross-record linking, performed by the platform, produce a dataset that researchers can reliably reconstruct and verify?*

These questions are addressed through an implementation chapter that describes how the platform was built, and an evaluation chapter that reports on a within-subjects user study comparing the platform with a manual Zenodo workflow on the same data publication tasks.

1.5 Contributions

Existing reproducibility systems focus primarily on software environments, while research data repositories focus on data publication. This dissertation contributes by integrating both concerns into a single workflow and augmenting them with language-model-driven automation and a hybrid interaction model. The specific contributions of the work are as follows.

1. **SciSuite is a redesign of SciConv** that integrates reproducibility packaging, reproduction of published artefacts from their persistent identifiers, and standalone dataset publishing within a unified platform, building on the existing conversational reproducibility engine of Costa, Barbosa, and Cunha [15].

2. **A Direct Data Management mode** that uploads datasets, generates metadata using a large language model, chunks large datasets when necessary, and cross-links the resulting Zenodo records, removing manual repository management from the researcher's workflow.

3. **A hybrid user interface design** that combines a chat surface with structured forms, a stage tracker, and explicit action buttons, supporting both free-form description and precise correction of structured output.

1.6 Document Structure

The remainder of this dissertation is organised as follows. Chapter 2 introduces the background concepts the work depends on, covering reproducibility, research data management, containerisation, large language models, and interaction paradigms. Chapter 3 reviews the state of the art across reproducibility frameworks, data management tools, language-model-assisted research platforms, and hybrid interface design, and identifies the gaps this work addresses. Chapter 4 presents the design of the proposed platform, including the packaging workflow, the reproduction of published artefacts from their identifiers, the new Direct Data Management mode, and the hybrid interaction model. Chapter 5 describes the implementation, with attention to the language model integration, the chunking and linking mechanism, and the deployment. Chapter 6 reports the user study design, including participants, instruments, and protocol. Chapter 7 presents and discusses the results. Chapter 8 concludes with a summary of contributions and an outline of future work.

Chapter 2

Background

This chapter introduces the concepts on which the rest of the dissertation depends. The platform described in later chapters sits at the meeting point of four fields: computational reproducibility, research data management, large language models, and user interface design. Each of these fields has its own terminology and its own assumptions, and a reader familiar with one will not necessarily be familiar with the others. The sections below present each in turn, at the depth needed to follow the design decisions made in this work. A review of the concrete tools and systems built on these foundations is left to Chapter 3.

2.1 Reproducibility in Computational Research

In computational research, reproducibility is often introduced through the short formulation of Peng [46]: an independent researcher takes the same code and the same data as an original study and obtains the same results. A complete definition, however, must state what exactly is held fixed. This dissertation adopts the definition of the National Academies report [42], under which reproducibility means obtaining consistent results using the same input data, the same computational steps, methods, and code, and the same conditions of analysis. Methods here covers the concrete procedures applied during execution, from data preparation to the algorithms and statistical steps that generate the results, while conditions of analysis covers the computational environment in which those methods run, including the operating system, the software dependencies and their versions, and the configuration of the execution. Replicability, by contrast, means obtaining consistent results across studies that address the same scientific question, where each study collects its own data [42]. The vocabulary around these terms has caused persistent confusion, to the point that several papers exist purely to sort out the terminology [4, 26, 47]. Two further terms complete the vocabulary. Repeatability refers to the original researcher rerunning the experiment under the same conditions and obtaining the same result. Runnability refers to the original researcher being able to execute the experiment from start to finish at a later time or on another machine after rebuilding the computational environment. It establishes that the artefact remains executable, but does not by itself demonstrate that an independent researcher can obtain the same result. Essawy

Figure 2.1: Levels of validation of a computational result, with the dimensions held fixed at each level. Adapted from Essawy et al. [22], with definitions following National Academies of Sciences, Engineering, and Medicine [42]; the same taxonomy is used by Costa [16].

et al. [22] arrange these notions along a progression of increasing independence between the original study and the verifying one, running from repeatability through runnability and reproducibility to replicability, as illustrated in Figure 2.1.

This dissertation operates in the middle of that progression. The question it cares about is whether the same code can be re-executed on the same data, and whether the sameness of both can be demonstrated rather than assumed.

The empirical case that reproducibility is a real problem rather than a theoretical worry is by now well established [2, 57]. In a widely cited survey of over 1,500 researchers, more than half agreed that science faces a significant reproducibility crisis, and large fractions reported having failed to reproduce other researchers' results and even their own [2]. Follow-up work has examined why. Stodden et al. [57] argue that the standards applied to computational work have not kept pace with its importance, and propose that code and data availability be treated as a requirement of publication rather than a courtesy. An empirical test of that idea found that even when a major journal adopted a policy requiring data and code to be shared on request, only a minority of authors actually provided usable artefacts, and only about a quarter of the examined studies could be reproduced [58]. Community-level responses such as the Transparency and Openness Promotion guidelines [44] have tried to shift incentives by giving journals a graded framework of disclosure standards to adopt.

A complementary perspective comes from Stark [56], who argues that before reproducibility must come what he calls preproducibility: a study must report enough about its data, methods, and computational environment for anyone to attempt a reproduction at all. In this light, the tooling reviewed in Chapter 3 is best understood as automated preproducibility: these tools capture the environment itself in an executable form instead of relying on the methods section of a paper to

describe it in prose.

Practical guidance for individual researchers has also matured. Sandve et al. [52] distil the daily habits that make computational work reproducible, including avoiding manual data manipulation, recording every result's provenance, and version controlling everything. The difficulty, as a Dagstuhl seminar on the reproducibility of data-oriented experiments concluded [24], is that these habits must hold across the entire research lifecycle, from data collection through analysis to archiving, and a failure at any stage breaks the chain. The same seminar drew attention to a point that is central to this dissertation: for data-intensive research, the handling of data, not the handling of code, is where reproducibility most often fails.

One further concept from this literature recurs throughout the present work. Provenance is the recorded history of a piece of data: how it was collected, processed, and transformed into the form in which it is used [19]. Provenance records give later users the context to interpret a result and the evidence that the underlying data has not silently changed. Capturing and querying provenance at the scale of modern datasets is a research area of its own [61].

2.2 Research Data Management and the FAIR Principles

Research data management is the discipline concerned with how datasets are stored, described, published, and preserved. Its current shared vocabulary comes from the FAIR principles [64]. Each letter unpacks into concrete expectations. Findability requires persistent identifiers and rich metadata. Accessibility requires a standard retrieval protocol, though not necessarily open access; data behind a documented authentication procedure can still be FAIR. Interoperability concerns shared formats and vocabularies, and reusability requires enough description, licensing, and provenance for someone else to use the data with confidence.

Two elements of this framework matter especially for this dissertation. The first is the persistent identifier, most commonly the Digital Object Identifier (DOI)[1]. An identifier gives a dataset a stable address that survives changes in hosting infrastructure, and it is the unit of citation when datasets are referenced in publications. Managing identifiers for data that evolves over time has its own established practice in the digital preservation community [48]. Combined with checksums, identifiers also support verification: a later user can confirm that the dataset retrieved today is byte-for-byte the dataset the original study used [10].

The second element is metadata, the structured description attached to a deposit: title, abstract, creators, keywords, licence, and version information. Metadata is what makes a dataset findable and interpretable, and repositories require a metadata record before a deposit can be published. It is also, by the consistent account of those who have implemented FAIR in practice, the weak point. Writing good descriptions is time-consuming work that brings the depositor no immediate benefit, and the result is that many records carry thin descriptions and few keywords, which quietly undermines the findability the whole framework is built on [9]. This observation is one of the direct motivations for the metadata assistance built into the platform described in this dissertation.

[1] https://www.doi.org

A practical constraint of public repositories also needs to be introduced here because it shapes a substantial part of the present work. Repositories impose size limits per deposit. When a dataset exceeds the limit, the depositor must split it across several records and document, somewhere, how the parts fit back together. Nothing in the FAIR framework or in repository tooling automates this. The splitting, the per-part metadata, the cross-references between parts, and the eventual reconstruction are all left to the researcher.

2.3 Containerisation

Containerisation is the packaging technique on which nearly all current reproducibility tools rest. A container bundles an application together with its complete software environment, including system libraries, language runtimes, and configuration, into an image that executes identically on any compatible host. Unlike a virtual machine, a container shares the host operating system kernel, which makes images comparatively small and fast to start.

Docker [41] brought this technique into mainstream use by combining existing Linux kernel features with a simple build recipe, the Dockerfile, and a public registry for sharing images. For scientific computing on shared clusters, where granting users the privileges Docker requires is often unacceptable, Singularity was developed as a container system designed around the security model of high-performance computing facilities [36].

For reproducibility, containers solve the environment problem: the single most common reason an old analysis no longer runs is that some library has changed version or disappeared, and a container freezes the entire stack at the moment of packaging. What containers do not solve is the data problem. An image with a large dataset baked into it becomes expensive to store, slow to transfer, and impossible to share when the data is restricted. The standard answer is to keep data outside the image and fetch it at execution time, which is exactly the point at which the data management concerns of the previous section enter the reproducibility picture: the experiment now depends on an external deposit being findable, retrievable, and verifiably unchanged.

2.4 Large Language Models

A language model assigns probabilities to sequences of text, which in practice means it can continue a given text in a plausible way. The models in current use are built on the transformer architecture, whose attention mechanism lets the model weigh relationships between all positions in its input at once [59]. Scaled to billions of parameters and trained on large text corpora, these models acquired a property that earlier generations lacked: they can perform tasks they were never explicitly trained for, given only a natural-language description and perhaps a few examples in the prompt [8]. A further training stage, in which the model is tuned on human-written demonstrations and human preference judgements, made the models markedly better at following instructions, and is the step that turned raw text predictors into usable assistants [45].

Three properties of these models matter for the present work. First, they handle unstructured and semi-structured input well. Given a set of file names, formats, and content samples from a dataset, a model can draft a plausible title, description, and keyword list, a task that fits the metadata gap described in Section 2.2. Second, they can produce structured output on request, returning their answer as fields rather than prose, which allows their proposals to be placed into forms that a user can edit. Third, their output is not deterministic: the same request can yield different answers between runs, which has consequences for how their proposals should be treated.

The most discussed of those consequences is hallucination, the tendency of language models to generate content that is fluent and confident but unsupported by their input or by fact [33]. In the setting of this dissertation the risk is concrete. A model asked to describe a dataset may attribute variables to it that it does not contain, invent a plausible-sounding collection methodology, or assign keywords that match the file names rather than the data. Because the model presents wrong content with the same fluency as right content, the error is not visible from the output alone; it can only be caught by someone who knows the dataset. The accepted mitigation, in this setting and in most practical deployments, is human oversight: the model's output is treated as a draft, and a person who understands the material reviews it before it takes effect [1]. This division of labour is workable only if reviewing and correcting the draft is cheap, which is an interface problem as much as a model problem. A correction that requires describing the desired change in prose costs more than the error it fixes; a correction that means editing one field in a form costs almost nothing. This observation carries directly into the interface discussion in the next section, and into the design of the platform itself, where every piece of model-generated metadata is presented in an editable form and nothing is published without the researcher's confirmation.

Interacting with these models has its own difficulties. The behaviour of a model depends on the wording of its prompt, and formulating prompts that produce the intended result reliably is a skill that ordinary users do not arrive with [18]. Guidance on how applications should wrap model capabilities has accordingly become an active topic in human-computer interaction, with the guidelines of Amershi et al. [1] the most widely cited reference: among other recommendations, an application should make clear what the system can do, show why it did what it did, and support efficient correction when it is wrong. These recommendations directly shape the interface decisions discussed next.

2.5 Interaction Paradigms: Conversation and Direct Manipulation

The idea of operating software through conversation is old. ELIZA, a simple pattern-matching program by Weizenbaum [62], demonstrated a typed natural-language exchange with a computer in the 1960s, and Weizenbaum himself was struck by how readily users attributed understanding to a program that had none. What has changed with large language models is not the form of the interaction but its substance: the system on the other side of the conversation can now actually carry out a wide range of requests. Conversational interfaces built on these models are deployed

today in virtual assistance, customer support, and accessibility settings, where they lower the barrier to interaction for users who struggle with conventional interfaces [30].

The strengths of conversation as an input mode are expressiveness and forgiveness. The user can state an intent the designer never anticipated, in whatever words come naturally. The weaknesses are ambiguity and opacity. The user cannot see what actions are available, cannot easily predict what the system will do, and must phrase requests with care to get consistent behaviour [18].

The opposite profile belongs to the graphical interface built on direct manipulation, the principle that the objects of interest should be continuously visible and operated on through physical-feeling actions with immediate feedback [53]. Buttons, forms, and menus make the available actions explicit, make outcomes predictable, and make correction trivial: to fix one field, the user edits that field. The trade-off is rigidity. A form can only collect what its fields anticipate, and an interface composed entirely of predefined controls cannot express an intent its designer did not foresee. These trade-offs, and the design strategies for managing them, are treated at length in the standard interface design literature [54].

Stated this way, the two paradigms are obviously complementary, and the interesting design question is not which to choose but how to combine them: when in a task conversation should carry the interaction, when structured controls should appear, and how the two should hand off to each other. A body of recent empirical and design work addresses exactly this question, and it is reviewed alongside the platform literature in Chapter 3.

2.6 Summary

Four threads run through this chapter. Computational reproducibility is a recognised and measured problem whose hardest remaining part, for data-intensive work, is the handling of data rather than code. Research data management offers the infrastructure of repositories, identifiers, and metadata, but leaves the descriptive work and the handling of oversized datasets to the researcher. Large language models are capable of drafting exactly that kind of descriptive content, provided their output is treated as a reviewable draft. And interface research shows that conversation and direct manipulation have complementary strengths that a single-modality design cannot capture. The next chapter reviews the concrete systems that have been built along each of these threads, and identifies the gap between them that this dissertation addresses.

Chapter 3

State of the Art

The previous chapter introduced the concepts this dissertation builds on. This chapter reviews the systems that have been built on those concepts. The review is a narrative one. It covers established systems in the reproducibility and research data management literature, together with recent work on language-model-assisted tools and hybrid interfaces that bears directly on the design of the proposed platform. The aim is not exhaustive coverage of every tool in each area, but a fair account of what the strongest existing systems do and where their support ends.

The chapter is organised by the function a system serves. Section 3.1 reviews container-based reproducibility frameworks and Section 3.2 reviews workflow orchestration systems. Section 3.3 covers research data management tools and repositories. Section 3.4 reviews conversational and language-model-assisted research tools, and Section 3.5 reviews hybrid conversational and graphical interfaces. Section 3.6 draws the threads together in a comparative analysis and states the research gap.

3.1 Container-Based Reproducibility Frameworks

The systems in this section all build on the containerisation technique introduced in Section 2.3: each captures the software environment of an experiment in an executable form so that the experiment can be run again elsewhere. They differ in when the capture happens, in where the resulting artefact can live, and in how much of the surrounding workflow they take on, and the subsections below review them in roughly that order of scope.

3.1.1 ReproZip

ReproZip [12] was among the first tools to make environment capture practical for working researchers. It operates by observation. While an experiment runs, ReproZip traces the process at the operating system level and records every file, library, and configuration item the execution touches. From this trace it assembles a self-contained bundle that can later be unpacked into a container or virtual machine and re-executed elsewhere. The approach requires no upfront effort from the researcher, since the trace happens at packaging time rather than during development.

The design assumes the experiment and its data run together on one machine and travel together in one bundle. For the small and medium experiments the tool was designed around, this assumption holds. For data-intensive work it does not: a trace-based bundle containing a hundred gigabytes of input data is no easier to store or share than the data itself, and ReproZip offers no mechanism for keeping the data elsewhere.

3.1.2 Binder

Binder [35] approaches reproducibility from the publishing side. Given a public repository containing notebooks and a declared environment specification, the Binder service builds a Docker image automatically and serves an interactive session in the browser. A reader of a paper can click a link and, once the automated build completes, be running the analysis without installing anything. This has made Binder the most accessible entry point to executable research, and it is widely used in teaching and in the notebook-based corners of computational science.

Its constraints follow from its hosting model. The code must be public, the build must complete within service limits, and the data must be small enough to live in the repository or be fetched quickly at start-up. Binder sessions are also ephemeral, so nothing about the platform addresses archiving or citation. It solves the demonstration half of reproducibility very well and leaves the preservation half untouched.

3.1.3 Code Ocean

Code Ocean [13] provides a commercial hosted environment in which a complete computational artefact, called a capsule, can be executed directly from a journal page. A capsule bundles code, environment, and data, receives a persistent identifier, and can be rerun by reviewers and readers with one click. Several publishers have integrated the platform into their review workflows, which has given it a role in formal peer review that the other tools in this section do not have.

Code Ocean's own documentation states that there is no limit to the size of data a capsule can hold, and its enterprise deployment, Code Ocean VPC, installs directly into the customer's own AWS account, with capsule data stored in the customer's own S3 buckets rather than Code Ocean's infrastructure. This removes the data-size ceiling for institutions able to provision and pay for their own cloud storage, but the claim does not extend to the individual free tier: an account on Code Ocean's free tier is limited to 5 GB of total storage and one hour of compute per month, so a researcher without an institutional deployment behind them cannot publish a dataset larger than this, or run an experiment longer than an hour, regardless of how the data is organised. Within these limits, Code Ocean offers the most polished execution experience of the reviewed systems, and its integration into formal peer review is a genuine strength no other reviewed tool has. Beyond them, the researcher is back to depositing data externally and linking it by hand, and the platform's reproducibility guarantees stop at its own boundary: a capsule reproduces inside Code Ocean's infrastructure, not independently of it.

3.1.4 Whole Tale

Whole Tale [11] links container execution with data publication in a single research environment. A Tale packages code, environment, and references to data; execution happens in the platform's infrastructure; and finished Tales can be published with persistent identifiers. The system integrates federated authentication through Globus, so researchers log in with institutional credentials, and it can register external datasets so that an analysis references data held elsewhere rather than embedding it.

Whole Tale comes closest, among the container-based systems, to treating data as a first-class concern. Its registration mechanism, however, assumes the data already sits in a supported repository, correctly described. The work of getting a large dataset into a repository, splitting it if it exceeds deposit limits, and writing its metadata happens before Whole Tale becomes useful, and the platform offers no help with it.

3.1.5 SciRep

SciRep [14] consolidates much of the prior work in this section into a single multi-domain framework, and it is the engine on which the platform in this dissertation is built. Given a project, SciRep inspects the files to identify languages, dependencies, and datasets, generates a Dockerfile and the supporting artefacts needed to recreate the environment, executes the experiment, and validates outputs by comparing them across runs. The accompanying survey of eighteen reproducibility tools, organised around ten characteristics in four dimensions, found that existing systems recreate environments well but remain weak in validation, data handling, and support across scientific domains, and SciRep was designed to address the first and third of these.

Two limitations of SciRep define the starting point of the present work. The first is data: the framework embeds datasets in the artefact, with no support for repository deposit, partitioning, or metadata. The second is audience: SciRep is operated through technical configuration, which places it out of reach of the researchers without a software background identified in Chapter 1.

3.2 Workflow Orchestration Systems

The container-based frameworks of the previous section package individual experiments. A different family of systems approaches reproducibility from another direction, through the automation of multi-step analyses.

Galaxy [31] is the longest-established example. It provides a web platform on which researchers, primarily in the life sciences, compose analysis workflows through a graphical interface, execute them on local or institutional infrastructure, and share both the workflows and their execution histories. Every step is recorded automatically, which makes analyses reproducible within the platform without the researcher doing anything beyond using it. Galaxy is also relevant to this dissertation for a second reason: it demonstrates that a graphical, form-driven interface can make

complex computational work accessible to researchers who do not program, which is the same accessibility goal the proposed platform pursues through different means.

Nextflow [21] is a domain-agnostic workflow language and engine. Pipelines written in it run unchanged from a laptop to a high-performance cluster or commercial cloud, with each step executed in a container and the whole pipeline versioned alongside the code. This combination has made it a standard in bioinformatics. Kepler [49] represents the provenance-oriented strand of workflow research, capturing data dependencies during execution and emphasising the reuse and recombination of workflow components across studies.

A related form of automation is borrowed from software engineering rather than from scientific workflow research. Continuous integration and continuous deployment (CI/CD) services, such as GitHub Actions and GitLab CI, build and test software automatically whenever its source changes, and the same machinery can re-execute a computational analysis on every change to its code or environment. Beaulieu-Jones and Greene [5] apply this idea directly in continuous analysis, which pairs a CI service with a container so that an analysis is rebuilt and re-run whenever its inputs change and the resulting figures are committed back for inspection, letting a result be reproduced without contacting its authors. The Popper convention [34] generalises the same practice to systems evaluation, treating an article and its experiments as a versioned repository whose results are regenerated and validated automatically. What CI/CD adds is continuous, automated verification that code and environment still produce the expected output, which the container-based frameworks of the previous section perform only on demand.

These systems automate the procedure of an analysis. What they assume is that the data the procedure consumes is already where it needs to be, in institutional storage or an open repository. Depositing data, describing it, satisfying repository size limits, and citing the result remain outside their scope.

3.3 Research Data Management Tools and Repositories

Where the previous two sections dealt with the code side of an experiment, this section turns to its data. It reviews the repositories to which datasets are published, the tools that package, reference, and verify data around those repositories, and the systems that demonstrate FAIR data handling at large scale.

3.3.1 Repositories: Zenodo and Dataverse

Zenodo,[1] operated by CERN (the European Organization for Nuclear Research), is the public repository targeted by the platform in this dissertation. It accepts deposits from any discipline and any file type, assigns a DOI to every record, supports versioned records whose identifiers resolve correctly as datasets evolve, and provides a metadata form covering titles, descriptions, creators, keywords, licences, and related identifiers. The related identifiers field allows records to declare

[1] https://zenodo.org

typed relationships to other records, which the proposed platform uses to link the parts of a split dataset to each other. Zenodo imposes a size limit per record, and when a dataset exceeds it the depositor must divide the data across multiple records by hand.

Dataverse [17] provides comparable functionality as an open-source platform that institutions host themselves. It assigns persistent identifiers, stores structured metadata, supports versioning, and serves as the publication backend for several of the reproducibility tools reviewed above. Together, the two represent the mature state of repository infrastructure: deposit, identification, and versioning are solved problems. What neither provides is assistance. The metadata form is empty until the depositor fills it, and the size limit is the depositor's problem to work around.

3.3.2 Packaging and Integrity: RO-Crate, BDBag, and DataLad

Around the repositories, a set of tools addresses how data is bundled, referenced, and verified. RO-Crate [55] defines a lightweight packaging convention in which datasets, code, and metadata travel together in a structured directory whose descriptions are readable by both humans and machines. It has been adopted as an exchange format by several research platforms and suits long-term archiving and citation.

BDBag and Minid [10] take the opposite approach to the same integrity problem: instead of bundling data, they exchange bags of references and checksums. The recipient fetches the data from wherever it lives and verifies that what arrived matches what was described. This keeps transfers small and proves identity without proving location, which suits very large datasets held in specialised storage.

DataLad [27] brings these guarantees into the daily workflow of a researcher. Built on Git[2] and git-annex [3], it gives datasets the same version history that code enjoys, tracks content by hash across local and remote copies, and retrieves only the files an analysis actually needs. It is the most complete answer available to the question of how a researcher should manage data day to day, and it presumes a comfort with Git that the audience identified in Chapter 1 does not generally have.

3.3.3 Scaling FAIR to Large Data

For data at the terabyte scale, several systems demonstrate that FAIR processing remains feasible. FAIRly Big [28] processed a population-scale imaging dataset through a fully tracked DataLad-based pipeline, keeping computation close to the data and capturing provenance throughout. IOSPReD [43] reduces the cost of packaging data-intensive experiments by including only the data fragments an experiment actually reads. The continuous FAIRness case study of Madduri et al. [40] documents the sustained effort required to keep a large biomedical analysis findable, citable, and re-executable over years, and is candid that the effort is considerable.

The collective picture from this section is that deposit, identification, integrity, packaging, and even scale are well served. The two tasks consistently left to the researcher are the descriptive

[2] https://git-scm.com
[3] https://git-annex.branchable.com

work of metadata, identified in Chapter 2 as the weak point of FAIR adoption in practice [9], and the mechanical work of splitting oversized datasets to satisfy repository limits and documenting how the parts reassemble.

3.4 Conversational and LLM-Assisted Research Tools

Language models entered research software first as conversational front ends and increasingly as working components. At the autonomous end of the spectrum, Boiko et al. [6] describe a system in which a language model plans and executes chemical experiments with limited human supervision, combining web search, documentation reading, and control of laboratory hardware. Systems of this kind show how much of a research workflow a model can in principle absorb, and equally how much oversight the absorption demands.

Most deployed systems are more conservative and keep the researcher in the loop. The recurring pattern is delegation with review: the model interprets a request, drafts code, configuration, or descriptive text, and the researcher accepts, rejects, or edits the draft. The pattern works when the surrounding interface follows the human-AI interaction guidance reviewed in Chapter 2, making the system's capabilities visible and correction cheap [1]. It fails when the only correction channel is more conversation, because describing a precise edit in prose is slower and less reliable than making the edit directly [18].

Within reproducibility specifically, the conversational approach is represented by SciConv [15], the system this dissertation extends. SciConv was developed by the same authors as the SciRep framework reviewed in Section 3.1.5, as a direct response to the second of its limitations: SciRep recreates computational environments well, but it is operated through technical configuration, which places it out of reach of the researchers without a software background identified in Chapter 1. SciConv removes that barrier by placing a chat interface over the SciRep engine, so that the engine's capabilities are reached through guided conversation rather than configuration files. The researcher uploads project files and answers questions in ordinary language; the system identifies dependencies, generates the container configuration, runs the experiment, and produces the reproducibility artefact. An experiment that would previously have required writing a Dockerfile becomes a guided conversation, which directly addresses the accessibility gap that motivates this work.

The original SciConv carries two limitations that define the present project. First, it inherits SciRep's data model: datasets are embedded in the artefact, and there is no support for depositing data in a repository, splitting it when it exceeds deposit limits, or generating its metadata. Second, its interaction is purely conversational. Structured output the system produces, such as inferred configuration values, can only be adjusted by describing the desired change in prose, which is exactly the correction pattern the interaction literature identifies as weak.

3.5 Hybrid Conversational and Graphical Interfaces

The complementary strengths of conversation and direct manipulation, introduced in Chapter 2, raise the design question of how to combine them. A growing body of empirical and design work addresses it.

The empirical comparisons establish that neither modality dominates. Flohr et al. [23] compared a chatbot against a conventional graphical interface for booking autonomous mobility services and found the graphical version more efficient for the structured parts of the task, while the conversational version absorbed requests the forms had not anticipated. Liu, Rietz, and Maedche [37] add that the better-performing modality also depends on the user: in repeated decision-making tasks, individual decision style influenced which interface served participants better. Results of this kind argue for designs that offer both modalities rather than betting on one.

Design-oriented work has begun to chart how such combinations should behave. Ribeiro [50] explored a hybrid graphical and conversational interface for interacting with a generative model and found that graphical elements help users navigate and act on large volumes of generated text. Järvinen [32] derived visual and interaction design guidelines for hybrid generative interfaces, including when predefined actions should accompany free-form input and how the two should share screen space. Liu and Martens [38] built a conversation-based hybrid interface for a structured survey technique and showed in a lab experiment that the combination guided participants through the procedure more reliably than conversation alone.

The design consensus emerging from these studies is that the modalities should coexist within a single interaction rather than occupy separate modes. Structured controls should appear at the points in a task where the options are few and well defined, conversation should remain available for everything the controls do not anticipate, and the transition between the two should be fluid rather than a mode switch. What this literature so far lacks is deployment evidence: the studies are prototypes and lab experiments, and evaluations of hybrid interfaces embedded in working domain platforms, performing realistic tasks against a meaningful baseline, are scarce. The user study reported later in this dissertation is designed to contribute exactly that kind of evidence.

3.6 Comparative Analysis and Research Gaps

Table 3.1 summarises the systems reviewed in this chapter against the capabilities that matter for the problem stated in Chapter 1: whether the system captures computational environments, whether it publishes data with persistent identification, whether it handles datasets that exceed repository deposit limits, whether it assists with metadata creation, and what interaction model it offers its users.

Reading the table column by column makes the situation plain. Environment capture has been substantially improved by modern containerisation technologies, as Chapter 2 anticipated. Data publication is served by the repositories and by the platforms that integrate with them. But no reviewed system automates the handling of datasets that exceed deposit limits, and none assists

Table 3.1: Capabilities of reviewed systems. Env. = environment capture; Publ. = data publication with persistent identifiers; Large = automated handling of datasets exceeding repository deposit limits; Meta. = assistance with metadata creation; Interaction = primary interaction model. A check mark indicates full support, "partial" indicates limited or indirect support for the capability in that column, and an en dash indicates no support.

System	Env.	Publ.	Large	Meta.	Interaction
ReproZip	✓	–	–	–	Command line
Binder	✓	–	–	–	Notebook
Code Ocean	✓	✓	partial	–	Graphical
Whole Tale	✓	✓	–	–	Graphical
Galaxy	partial	–	–	–	Graphical
Nextflow	✓	–	–	–	Command line
DataLad	–	partial	–	–	Command line
RO-Crate	–	partial	–	–	Library/format
Zenodo	–	✓	–	–	Graphical
Dataverse	–	✓	–	–	Graphical
SciRep	✓	–	–	–	Configuration
SciConv	✓	–	–	–	Conversational
SciSuite	✓	✓	✓	✓	Hybrid

with metadata creation, despite metadata quality being the consistently reported weak point of data publication in practice. The interaction column shows a second pattern: the systems accessible to non-technical researchers are graphical and offer no conversational flexibility, the conversational system offers no structured controls, and no system combines the two.

The gap this dissertation addresses is therefore the integration of capabilities that exist separately, together with two capabilities that do not yet exist in any reviewed system. By extending SciConv [15] and the SciRep engine [14] with a standalone data management mode, language-model-driven metadata generation, deterministic chunking with cross-record linking for oversized datasets, and a hybrid conversational and structured interface, the proposed platform fills the two empty columns of Table 3.1 and unifies the rest. Its evaluation, a within-subjects user study against the manual repository workflow on realistic publication tasks, additionally supplies the deployment evidence that the hybrid interface literature currently lacks. The architecture of the platform is presented in the next chapter.

Chapter 4

The SCISUITE Platform

This chapter presents the platform developed in this dissertation. It describes the platform at the level of its design: what it does, how it is organised, and why it is built the way it is. The engineering behind the design, including the technology stack, the backend interfaces, the chunking mechanism, and the deployment, is the subject of Chapter 5.

The platform extends SCICONV [15], the conversational reproducibility tool built on the SCIREP engine [14].

4.1 Design Goals

The comparative analysis in Chapter 3 identified what existing systems leave undone. Those findings translate into four design goals that shaped every decision described in this chapter.

G1: One workflow from project folder to published, linked artefacts.

A researcher should be able to start with code and data on their machine and finish with a reproducible artefact and a citable dataset, correctly connected to each other, without leaving the platform. No reviewed system offers this; the platform should.

G2: Automated handling of oversized datasets.

When a dataset exceeds the repository deposit limit, the platform, not the researcher, should split it, deposit the parts, link them to each other, and record how they reassemble. This fills the column of Table 3.1 that no reviewed system fills.

G3: Assisted metadata.

The descriptive work of data publication, identified in Chapter 2 as the weak point of FAIR practice, should be drafted by the platform and reviewed by the researcher, rather than written by the researcher from nothing.

G4: Accessibility without loss of control.

The platform should be usable by researchers with no software engineering background, which argues for conversation, while still giving precise control over structured output, which argues for forms and buttons. The hybrid interaction model of Section 4.4 is the response to this tension.

4.2 Architecture

Figure 4.1 shows the architecture of the SciSuite platform. Five parts matter for this discussion.

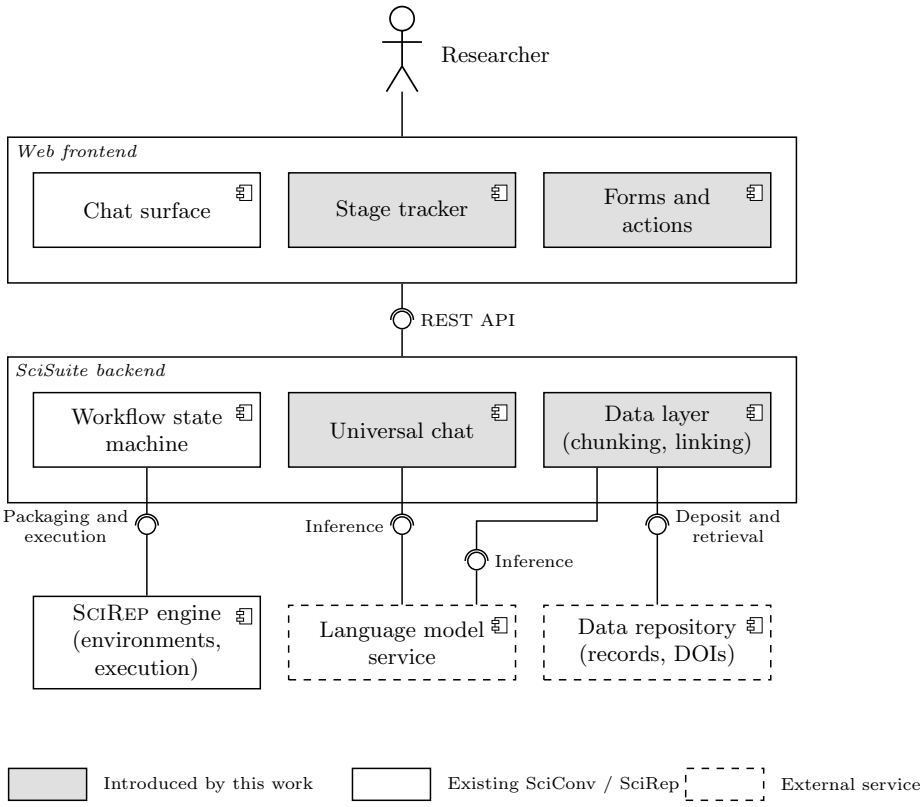


Figure 4.1: Architecture of the SciSuite platform. Shaded components are introduced by this dissertation, white components belong to the original SciConv and SciRep stack, and dashed components are external services.

The **web frontend** is what the researcher sees. It is a single page that combines a conversational surface, where the system and the researcher exchange messages, with structured elements that appear and disappear as the workflow progresses: a stage tracker across the top, action buttons, file upload controls, and editable forms. The coordination of these elements is the subject of Section 4.4.

The **backend** maintains the state of each session. Every project moves through a defined sequence of stages, and the backend records which stage a project is in, what information has been collected, and what remains to be decided. The backend also hosts the universal chat: a single channel through which any free-text message from the researcher, at any stage, is interpreted by a language model that knows the current project state and the actions available at that point. This

design means the researcher is never told that typing is unavailable; whatever they write is read in context and either acted on or answered.

The **SCIREP engine** [14] performs the reproducibility work: it inspects uploaded code to identify languages and dependencies, generates the Dockerfile and supporting artefacts, builds the container, executes the experiment, and collects the outputs. The platform does not modify the engine's responsibilities; it changes how the engine is driven and what happens to data around it.

The **data layer** is introduced by this dissertation. It measures datasets against the repository deposit limit, splits those that exceed it into deterministic parts, deposits each part as its own repository record, writes cross-references between the records, and includes in each record's description a statement of which part it is and how many parts exist. For datasets under the limit, it performs a single deposit. In both cases it drives the metadata workflow described in Section 4.3.

The **external services** are the data repository, which holds the published records and issues their persistent identifiers, and a hosted large language model accessed through an Application Programming Interface (API), which the backend calls for the inference tasks listed in Section 4.5.

4.3 The Three Workflows

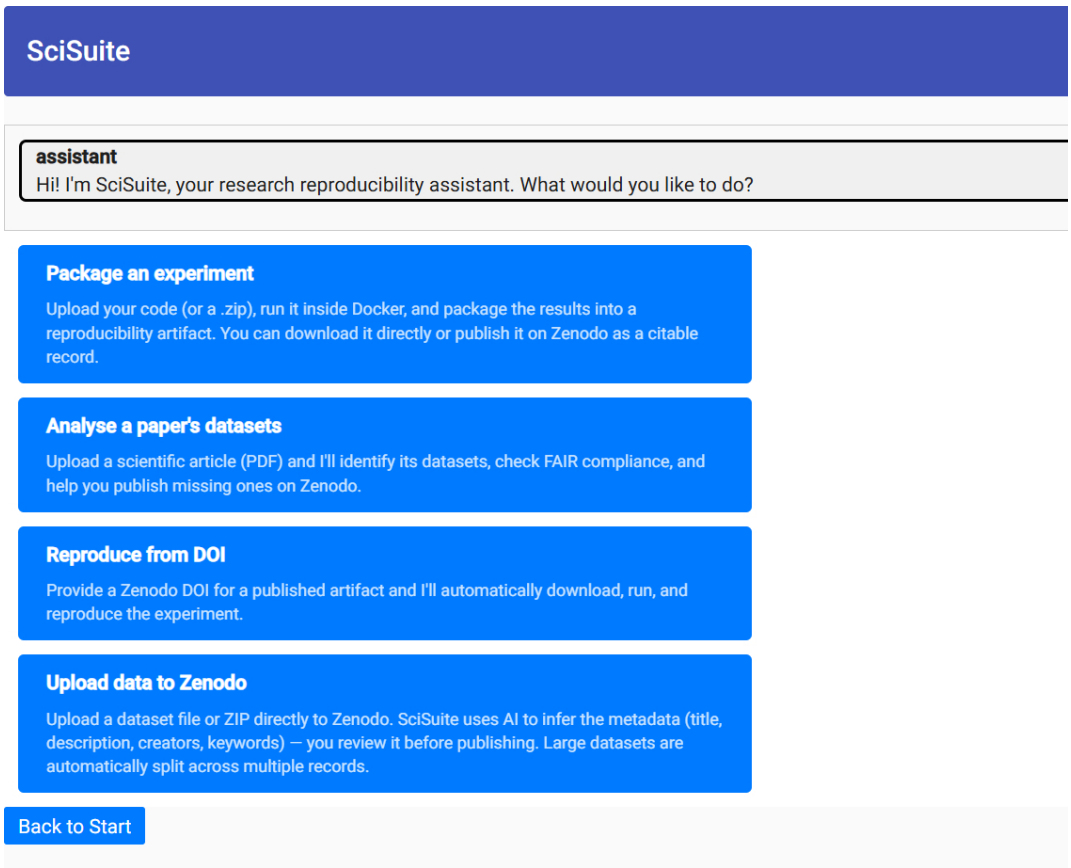


Figure 4.2: The conversational welcome screen. The researcher starts by selecting one of the displayed workflow options.

The platform opens on the welcome screen shown in Figure 4.2, where the researcher selects the workflow they want to start. The screen presents the available workflows as explicit options, so the researcher begins by choosing the task that best matches their goal. Once a workflow has been selected, the platform continues through the guided conversational interface, combining assistant messages with the structured controls described later in this chapter. The original SciConv presented its modes as separate sections of a form-based homepage; the redesign into a workflow-selection entry point followed feedback that a single guided interaction was easier to approach.

The welcome screen offers four options. One of them, the analysis of a published paper's dataset, belongs to the original SciConv and is not modified by this dissertation, so it is only noted here for completeness. The remaining three are the workflows through which the contributions of this work run, and the rest of this section describes them in turn: packaging an experiment, reproducing an experiment from its identifier, and the new Direct Data Management mode.

4.3.1 Packaging an Experiment

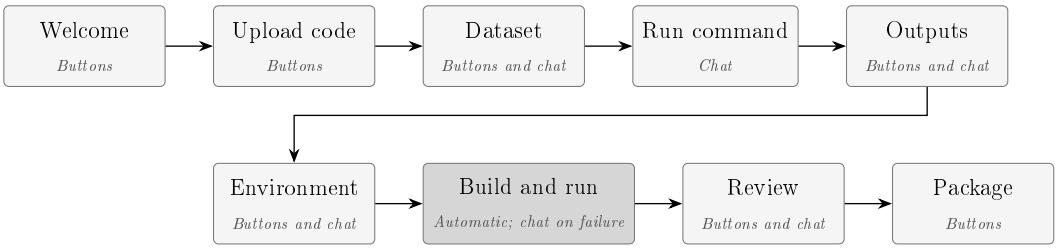


Figure 4.3: Stages of the packaging workflow. The chip beneath each stage names the interaction modality offered at that point.

The first workflow takes a researcher from a project folder to a reproducible artefact. It proceeds through the stages shown on the stage tracker: uploading the code, providing a dataset, confirming the run command, identifying the expected outputs, reviewing the inferred environment, building and executing the container, reviewing the result, and packaging.

The dataset stage accepts three answers, reflecting the three situations a researcher can be in. They can upload a data file, which the platform will embed directly in a Docker image if it is smaller than the defined limit, and externalise to a data repository if it is above the limit. They can provide the DOI of a dataset already published in a repository, in which case the platform records the reference for retrieval at execution time. Or they can state that the experiment uses no dataset and move on. The first two answers connect this workflow to the data layer; the third keeps the original SciConv behaviour.

After execution, the researcher reviews the result. If the output is wrong, they describe what happened in the chat, and the platform attempts a repair: adjusting the run command, correcting a dependency, or rebuilding as needed. When the researcher confirms the result, the platform

produces the artefact, which can be downloaded directly or published to Zenodo with its own metadata and identifier.

Figure 4.3 summarises the workflow as a stage sequence, with the interaction modality offered at each stage. Reading across it shows that the workflow is not uniformly conversational or uniformly structured; each stage is assigned the modality that fits it.

4.3.2 Reproducing from a DOI

The second workflow is introduced by this dissertation and closes the loop that the first one opens. Given the DOI of an artefact previously published through the platform, the platform retrieves the artefact from Zenodo, restores its environment, fetches any externally referenced data, re-executes the experiment, and presents the outputs for comparison with the originals. The original SCICONV produced artefacts but offered no path for consuming them; this workflow adds that path.

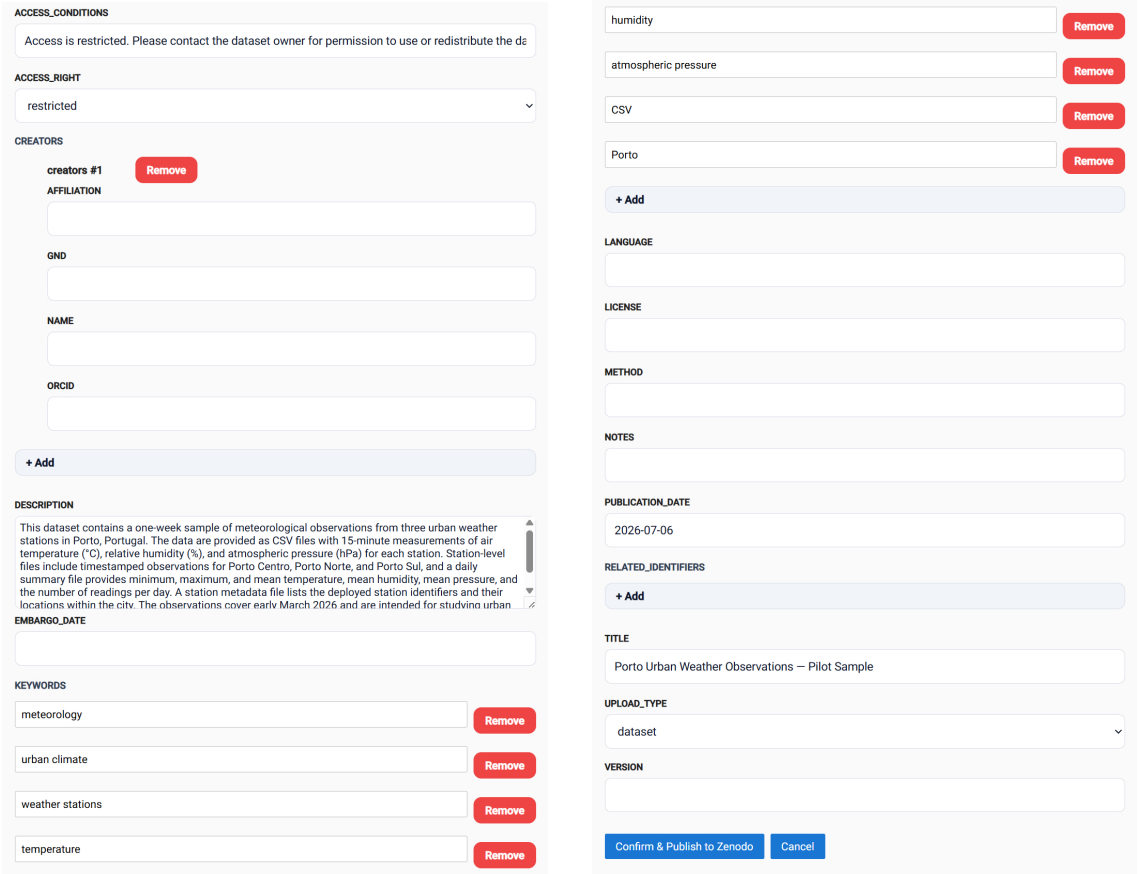
This workflow is what makes the platform's artefacts more than archives. An examiner of a published result, a reviewer, or the original researcher two years later can reproduce the experiment with nothing but its identifier. Where the dataset was externalised at packaging time, the reproduction depends on the data layer's records: the parts are retrieved, reassembled in their recorded order, and supplied to the container as the original execution received them.

4.3.3 Direct Data Management

The third workflow is new in this dissertation and exists for the researcher who has data to publish and no experiment to package. It was added after supervisory discussions made clear that this situation is common and poorly served: the researcher in it would otherwise leave the platform for the repository's web forms, where they receive no assistance with description or with size limits.

The workflow is short by design. The researcher uploads a dataset. The platform inspects it, drafts the repository metadata, and presents the draft in an editable form, shown in Figure 4.4: containing fields such as title, description, keywords, creators, and the remaining fields the repository requires. The researcher corrects whatever needs correcting and confirms. The platform then measures the dataset against the deposit limit. If it fits, it is published as a single record. If it does not, the data layer splits it, publishes each part as its own record with metadata derived from the confirmed draft, marks each description with the part's position in the whole, and writes the cross-references that let a later user, or the reproduction workflow above, find every part from any part.

The result, in either case, is a citable, described, FAIR-aligned deposit produced in minutes from a conversation and a form. The user study in Chapter 6 evaluates exactly this workflow against the manual alternative.



(a) Access, creators, and description fields.

(b) Keywords, licensing, and publication fields, with the confirmation controls.

Figure 4.4: The metadata editor of the data management workflow. The fields arrive populated with the language model's draft; the researcher edits whatever needs correcting and nothing is published before confirmation.

4.4 The Hybrid Interaction Model

Chapter 2 argued that conversation and direct manipulation have complementary strengths, and Chapter 3 reviewed the evidence that combining them helps. This section describes how the platform actually combines them. The design follows one principle: the modality offered at each moment should match the structure of the decision at that moment. Where a step has a small number of well-defined options, the platform shows them as buttons. Where a step requires precise structured input, it shows a form. Where the space of possible inputs is open, it relies on the chat. And at many steps it offers both, because the same decision can be easy to click for one researcher and easier to say for another.

Three recurring elements implement the principle.

The **stage tracker** runs across the top of the interface and shows every stage of the active workflow, with the current one highlighted. Conversational interfaces tend to obscure where the user is in a task and how much remains; the tracker restores that visibility without taking any

expressiveness away. It also gives the researcher a stable mental model of the workflow that the conversation below it can refer to.

The **contextual actions** are the buttons and controls that appear alongside chat messages when the workflow reaches a decision with known options. The dataset stage, shown in Figure 4.5, is the clearest example: the platform asks whether the experiment uses a dataset and simultaneously offers an upload control, a field for a DOI, and a button stating that there is no dataset. The researcher can click any of these or answer in the chat in their own words; both routes converge on the same state change. The earlier design, in which the researcher was instructed to type the words "no data", was replaced by the button after observation showed the instruction itself caused hesitation.

Figure 4.5: The dataset stage of the packaging workflow. The stage tracker shows the position in the workflow, the assistant message states the available options, and the same decision can be made through the buttons or through the chat input.

The **editable forms** appear where the platform has produced structured content that the researcher must be able to inspect and correct: above all, the metadata editor of the data management workflow, shown in Figure 4.4. The form is the answer to the correction problem identified in Chapter 2: when a language model drafts a record, fixing one field by editing it directly costs almost nothing, whereas fixing it by describing the change in prose costs a round trip through the model and a re-check of everything else. Nothing in the form is published until the researcher confirms it.

Beneath all three elements, the chat input remains permanently available. The universal chat design described in Section 4.2 means a typed message is interpreted in the context of the current stage whatever that stage is: a researcher who types a DOI at the dataset stage, or writes "the output folder is called results" at the outputs stage, or says "something went wrong, the numbers are all zero" at the review stage, is understood. The interaction therefore never has modes in the user's sense. Structure is added to the conversation where it helps and the conversation continues through it.

Looking across the stages of Figure 4.3, the combinations the platform actually uses settle into three recurring patterns, and the choice between them follows from the shape of the decision at each stage.

The first pattern offers **both modalities at once**. It appears wherever a stage has a small set of well-defined alternatives but a researcher might also hold information the options do not anticipate. The dataset stage is the standing example: most researchers will click one of the three offered answers, but one who already has a published dataset can simply type its DOI, and the conversation absorbs it. Offering both costs little and lets each researcher take whichever path is faster for them.

The second pattern is **structured only**. It appears where the task demands complete and precise input whose shape is fully known in advance. The metadata editor is the clearest case: a repository record has fixed fields, every field must be inspected, and any of them may need a small correction. Conversation adds nothing here except the risk of imprecision, so the platform presents a form and waits.

The third pattern is **conversational only**. It appears where the space of possible situations is open and cannot be enumerated as options, above all after an execution. An experiment can fail or mislead in too many ways for any fixed set of buttons to cover: a missing dependency, a wrong command, an output that exists but contains the wrong numbers. Here the researcher describes what they see in their own words, and the language model maps the description onto a corrective action. No structured control could do this work, and pretending otherwise would only force the researcher's open-ended observation through a closed set of choices.

The three patterns are not modes the researcher selects between; they are decisions the designer has already made, stage by stage, about where structure helps and where it would get in the way. What remains constant across all of them is the visible position in the workflow, provided by the stage tracker, and the always-open chat input, so that moving between patterns feels like the conversation continuing rather than the interface changing.

4.5 Language Model Integration and the Oversight Pattern

The platform calls a language model at the points in its workflows where the input is unstructured and the required output is a draft that a person can check. There are four such points.

Dataset metadata inference. In the data management workflow, the model receives a sample of the uploaded dataset, including file names, formats, and content excerpts, and drafts the repository metadata: title, description, and keywords. The draft populates the metadata editor for review. Figure 4.6 shows this path, together with the oversight pattern that governs it.

Artefact metadata inference. In the packaging workflow, when the researcher chooses to publish the finished artefact, the model drafts the equivalent record for the artefact itself, from the project's code, structure, and outputs.

Output identification. The model assists in identifying which files an experiment produces as its results, proposing candidate output locations that the researcher confirms or corrects.

Conversation and repair. The universal chat is itself a model call: every free-text message is interpreted against a description of the current stage, the available actions, and the project state.

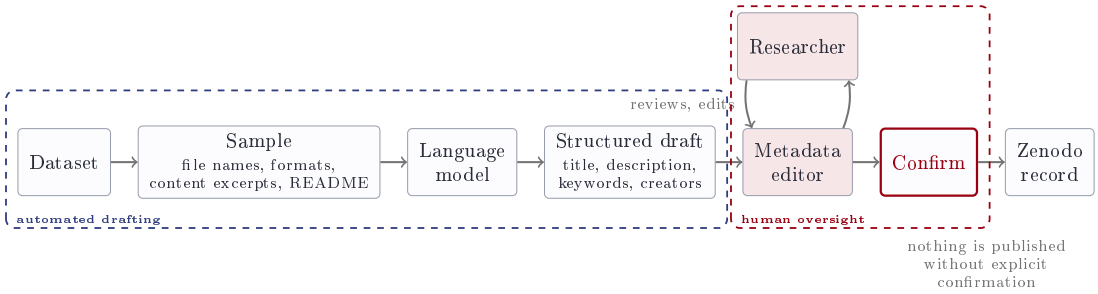


Figure 4.6: Dataset metadata inference and the oversight pattern. The model drafts from a sample of the dataset; the draft populates the metadata editor, where the researcher reviews and corrects it, and nothing reaches the repository without explicit confirmation.

The same channel carries the repair dialogue after a failed execution, where the researcher describes the problem and the model maps the description onto a corrective action.

Every one of these integration points follows the oversight pattern established in Chapter 2. The model's output is a proposal. It is rendered in a form or a confirmation step, the researcher reviews it, and nothing irreversible, in particular nothing involving publication to the repository, happens without explicit confirmation. The hallucination risk discussed in Section 2.4 is managed by exactly this division of labour: the model does the drafting that researchers neglect, and the researcher does the checking that the model cannot be trusted to do. Whether the drafts are good enough that the division actually saves work is an empirical question, and it is one of the questions the user study in Chapter 6 is designed to answer.

4.6 Summary

This chapter presented the design of SciSuite, the extended SciConv platform: four goals derived from the gaps identified in the state of the art, an architecture that adds a data layer and a universal conversational channel to the existing SciConv and SciRep stack, three workflows that together cover packaging an experiment, reproducing one from its identifier, and publishing a dataset on its own, a hybrid interaction model that matches modality to the structure of each decision, and a consistent oversight pattern governing every use of the language model. The next chapter descends one level and describes how this design is implemented.

Chapter 5

Implementation

The previous chapter described the platform at the level of its design. This chapter describes how that design is realised in software. It begins with the technology stack and the overall structure of the codebase, then covers the parts of the implementation that are specific to this dissertation: the conversational control layer and its action protocol, the data layer that performs chunking and repository linking, the language model integration, the reproduce from DOI mechanism, and the deployment used for the user study. Where a component was inherited from the original SciConv and used without substantial change, it is described only briefly; the emphasis is on what was built for this work.

5.1 Technology Stack

The platform is a web application with a Python[1] (3.12) backend and an Angular[2] (14) frontend.[3] The most important constraint on the technology choices was continuity: the platform extends the original SciConv rather than replacing it, and SciConv was already a Flask and Angular application wrapped around the SciRep engine. Adopting a different backend framework or frontend library would have meant reimplementing the existing integration with SciRep for no functional gain, so the established stack was kept and extended. The justifications below should be read in that light; in several cases the decisive reason is compatibility with the system being extended rather than an abstract preference.

The backend is written in Python and built on the Flask[4] web framework. Python is the natural choice because the SciRep engine, the Docker control libraries, and the OpenAI client are all Python, and because the experiments the platform packages are frequently themselves written in Python. Flask was retained from the original SciConv; its request-driven model suits the

[1] https://www.python.org

[2] https://www.angular.dev

[3] The complete source code for SciSuite is publicly available at https://github.com/anex4real/Scisuite; the archived release used for this dissertation is available at https://doi.org/10.5281/zenodo.21284348

[4] https://flask.palletsprojects.com/en/stable/

platform's workload, which is a sequence of discrete stage transitions rather than the many concurrent long-lived connections for which an asynchronous framework such as FastAPI would be the stronger choice. The backend holds the state of each session and orchestrates the other components: the SciRep engine for environment work, the OpenAI[5] API for language model inference, and the Zenodo API for repository deposits. Containerised execution is performed through Docker [6] (29.1), driven from the backend through the Docker SDK for Python. The remaining third-party dependencies are Flask-CORS for cross-origin handling, the official OpenAI Python SDK and the configured language model (gpt-5.4-mini), and supporting libraries for PDF reading and HTML parsing.

The frontend is a single-page application written in Angular, inherited from the original SciConv. Angular suits the platform's interface because the hybrid model of Chapter 4 relies on structured, form-heavy components, the metadata editor in particular, and Angular's built-in support for forms and two-way data binding reduces the work of building them; a more minimal library such as React would have required additional form-handling machinery for the same result. The frontend renders the conversational surface, the stage tracker, the contextual action buttons, and the editable forms, and communicates with the backend over HyperText Transfer Protocol (HTTP). It holds no application logic of its own beyond presentation and input handling; every decision about what happens next is made by the backend and reflected back to the frontend as state.

External services are reached over their HTTP APIs. Language model inference is performed through the OpenAI Chat Completions API, accessed by way of a single helper function so that the model identifier and sampling temperature are set in one place rather than at each call site. At the time of writing the helper requests the `gpt-5.4-mini` model with a temperature of 0.2, a low value chosen so that the structured output the platform depends on, such as inferred metadata fields and action tags, stays stable between runs. A hosted API was preferred over a locally run model for two reasons. The quality of the inferred metadata and the reliability of the action-tag protocol depend on a capable model, and the hosted models available through the API are stronger than those that run on modest hardware such as the study's virtual machine. Routing model access through a single helper also means that the choice is not permanent: because every call passes through one function with the model identifier as a parameter, moving to a different hosted model, or to a local one, would be a change in one place rather than throughout the codebase. Data publication uses the Zenodo REST API[7]; Zenodo was selected among the repositories reviewed in Section 3.3.1 because it is open to any discipline and file type at no cost, and because its related-identifiers mechanism is exposed through the API, which the cross-record linking depends on. The data layer addresses the repository through its API alone, so the choice is not structural.

[5] https://www.openai.com/API
[6] https://www.docker.com
[7] https://zenodo.org

5.2 Backend Structure and Session State

The backend organises a session as a progression through a defined set of stages. Each stage corresponds to one step of a workflow, such as waiting for a code upload, waiting for a dataset decision, inferring an environment, or reviewing a result. The backend records which stage the current session occupies, together with the information gathered so far, and every interaction either advances the session to a new stage or updates the data held at the current one.

This stage model is what allows the platform to mix conversation with structure while keeping the interaction coherent. The stage determines which contextual actions the frontend shows, what a free-text message is most likely to mean, and which transitions are valid. Because the interpretation is tied to the current stage, the researcher can drive navigation from the chat in ordinary language rather than through fixed controls. In the packaging workflow, a researcher who has already moved past the run command can type "I want to change the run command", and the universal chat resolves this to a transition back to the run-command stage and returns the session there, with the value entered earlier available for editing. The same applies to any earlier stage: the researcher describes the change they want, and the backend maps it onto the corresponding backward transition. When this happens, or when the session restarts, the backend clears the data and intermediate artefacts the abandoned path produced, so that a later stage never runs on stale inputs from an earlier attempt. This cleanup is handled centrally rather than being scattered across the individual stages.

The chat carries the recovery dialogue after a failed execution in the same way. When a build or run fails, the researcher does not have to diagnose the problem from raw logs; they can ask what went wrong, and the platform interprets the failure and replies in plain language with a likely cause, such as a missing dependency, a wrong run command, or an incorrect output path. Having identified the probable cause, it proposes the stage where the problem can be fixed, for instance offering to return to the environment stage to add a dependency or to the run-command stage to correct the command, and carries out the transition once the researcher agrees. In this way both ordinary navigation and failure recovery run through the same conversational channel, and the stage the session lands in is reached by description rather than by the researcher having to locate the right control.

The three workflows described in Chapter 4 are all expressed as sequences of these stages, so they share the same conversational machinery, the same cleanup behaviour, and the same frontend components. Packaging an experiment moves through upload, dataset, run command, outputs, environment, build and execution, review, and packaging, shown in Figure 5.1; its build-and-run stage runs automatically, and a failed execution branches to a recovery state that returns the session to an earlier stage for correction. The data management workflow (Figure 5.2) is shorter, moving from upload through metadata inference and review to publication, and the reproduction workflow (Figure 5.3) shorter still. These two are linear: each step follows directly from the previous one, so the researcher advances through them mainly with the structured controls, and they do not rely on the chat for navigation or recovery in the way the packaging workflow does.

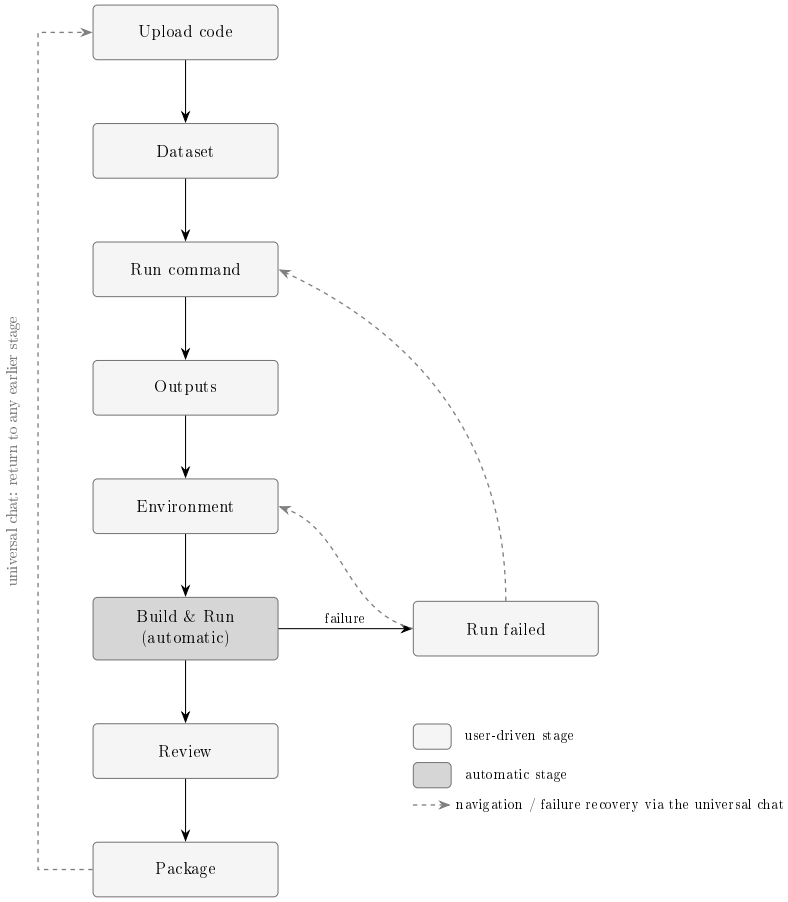


Figure 5.1: The packaging workflow as a sequence of backend stages. Solid arrows show the forward progression and the stage marked automatic is performed without researcher input. On failure the session moves to Run failed and returns to an earlier stage. Through the universal chat the researcher can also navigate back from any stage to any earlier stage, shown by the dashed path on the left.

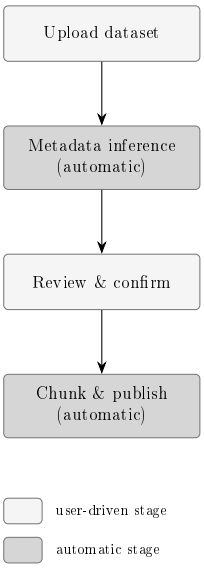


Figure 5.2: The data management workflow. The researcher uploads a dataset, the platform drafts its metadata, the researcher reviews and confirms it, and the platform then chunks and publishes the records. The automatic stages run without further input.

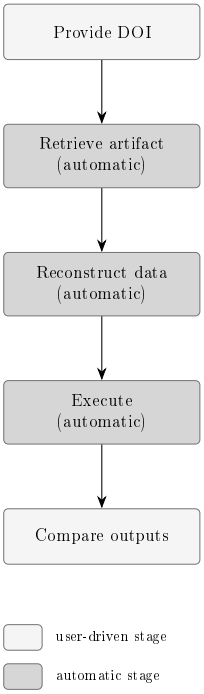


Figure 5.3: The reproduce from DOI workflow. From a single identifier the platform retrieves the artefact, reconstructs any externalised data, re-executes the experiment, and presents the outputs for comparison.

5.3 The Conversational Control Layer

The element that ties the workflows together is what Chapter 4 called the universal chat: a single channel through which any free-text message, at any stage, is interpreted in context. This section describes how that channel is implemented.

Every free-text message the researcher sends is passed to the language model together with a system prompt that describes the workflow, the current stage, the project state gathered so far, and the conversation history. The model is instructed to reply in plain language and, only when it is certain of the researcher's intent, to append a single action tag drawn from a fixed vocabulary. The backend interprets this tag, rather than the prose, and maps it onto a state transition, while the natural-language part of the reply is shown to the researcher. The tags fall into two groups: navigation tags that move the session to a named stage, and command tags that carry a piece of information the researcher has supplied. Table 5.1 lists them.

Table 5.1: The action-tag vocabulary of the universal chat. Navigation tags move the session to a named stage; command tags carry information the researcher has supplied. The model may append at most one tag per reply, and only one drawn from this fixed set.

Action tag	Meaning
`navigate:WaitForDataInput`	Return to the dataset stage to change or add a data file or DOI
`navigate:ParametersToUse`	Return to change the run command
`navigate:SpecifyOutputs`	Return to change the output file specification
`navigate:FindConfigurations...`	Return to review or change the inferred environment
`navigate:BuildDockerFile`	Fix and rebuild the Dockerfile
`navigate:RunContainer`	Retry the run immediately
`confirm`	The researcher confirmed the current prompt
`set_command`	The researcher provided a run command
`set_outputs`	The researcher provided output file paths
`update_config`	The researcher wants to change the environment configuration
`report_issue`	The researcher reported a problem after a run
`reset`	The researcher wants to start over with a new project

Two behaviours illustrate how the channel works in practice. The first is explanation. When the researcher asks a question about the current step, such as what a stage does or what input it expects, the model is instructed to answer in prose and to append no action tag. The backend, finding no tag at the end of the reply, simply shows the explanation and leaves the session where it is. Explanation is therefore the default behaviour of the channel: any message that is a question rather than an instruction is answered without changing the workflow state.

The second is navigation. When the researcher asks to return to an earlier step, for instance to change the run command or to revise the inferred environment, the model appends the corresponding navigation tag. The backend interprets the tag as a request to move the session to a named stage, validates that the transition is permitted, and carries it out, while the central cleanup

described in Section 5.2 discards any intermediate artefacts produced by the path now being abandoned, so that the earlier stage resumes from a clean state. The researcher reaches the same destination they would have reached by clicking, but expresses the request in their own words.

This design removes explicit interaction modes from the user's perspective. The researcher is never restricted to the contextual buttons on screen, because anything they type is mapped onto the same set of internal actions those buttons trigger. A researcher who prefers to click chooses an action directly; a researcher who prefers to type expresses the same intent in words and the model resolves it to the same action. The fixed tag vocabulary keeps the model's freedom bounded: the model decides what the researcher meant, but it can only choose from the actions the current stage actually permits, which prevents a fluent but invalid instruction from putting the session into an impossible state.

The bound is reinforced by the system prompt itself, which states the valid tags, instructs the model to withhold a tag when the message is ambiguous, and confines the assistant to the SCISUITE workflow so that off-topic requests are declined rather than answered. Figure 5.4 shows the relevant portion in condensed form.

Valid action tags. Only append one of these tags at the end of the reply, and only when certain the action is what the user wants:

`[ACTION:navigate:ParametersToUse]`	change the run command
`[ACTION:set_command]`	the user provided a run command
`[ACTION:confirm]`	the user confirmed the current prompt
`[ACTION:report_issue]`	the user reported a problem
`[ACTION:reset]`	the user wants to start over
(full vocabulary in Table 5.1)	

How to respond. Reply in plain natural language; be concise and specific to the current stage. Append an `[ACTION:...]` tag only when certain of the user's intent. If the message is ambiguous (*"ok"*, *"sure"*, *"thanks"*), or the user is asking a question or needs guidance, reply naturally with no action tag.

Scope restriction. Assist only with tasks related to this experiment or the SCISUITE workflow; politely decline anything outside this scope and redirect the user to the current step.

Figure 5.4: Condensed extract of the universal-chat system prompt.

The same channel carries the recovery dialogue after a failed execution. When an experiment fails to run or produces an output the researcher judges wrong, the researcher describes the problem in ordinary language, and the model maps that description onto one of a small set of corrective actions: adjusting the run command, revising the inferred dependencies, or rebuilding and re-running. This is the conversational-only pattern from Chapter 4 in concrete form. The space of things that can go wrong with an execution is too large to enumerate as buttons, so the interaction relies on description and lets the model do the mapping.

5.4 The Data Layer

The data layer is the component that handles datasets too large to embed in a reproducibility artefact and datasets published on their own through the data management workflow. It performs four tasks: deciding whether a dataset must be externalised, splitting it deterministically when it exceeds the repository limit, depositing the parts to Zenodo with linked metadata, and, at reproduction time, retrieving and reassembling them.

5.4.1 Externalisation Decision and Chunking

When a dataset enters the platform, the data layer measures its size against a threshold and decides whether it can be embedded in the artefact or must be externalised. The measurement is taken on the dataset's unpacked content rather than on the size of any compressed archive the researcher uploaded, because it is the unpacked data that must fit within the repository's per-record limit. A small archive that expands to a large dataset is treated as large.

When a dataset must be split, the data layer first writes the whole dataset directory into a single tar archive, then divides that archive into parts of a fixed maximum size by reading it sequentially and writing fixed-size byte ranges to numbered output files. Working from the tar archive rather than from the individual files means the split is purely a matter of byte offsets, which makes it deterministic: the same archive, split with the same threshold, always produces the same parts in the same order. Determinism matters because the parts must be reassembled exactly at reproduction time, and any variation in how the data was divided would change the reconstructed dataset and break the guarantee that the reproduced experiment uses the same data as the original. The data layer computes an MD5 digest of the full archive and of each part, and records each part's position in the sequence, so that reconstruction can both order the parts correctly and verify that each arrived intact. MD5 is used here for integrity rather than security: the digest exists to detect accidental corruption during transfer or reassembly, not to defend against a deliberately forged part, and for that purpose its speed and short digest are advantages over a cryptographic hash such as SHA-256. Should the threat model ever change, the digest function is confined to a single helper and could be replaced without affecting the rest of the layer.

The size of the threshold is a configuration value rather than a fixed constant of the design. In normal operation it is set to Zenodo's per-record limit of 50 GB. For the user study reported in Chapter 6 it was deliberately lowered to 1 MB, so that a modestly sized dataset would cross the threshold and split into multiple linked records, exercising the splitting and linking behaviour within the time available for a session. The mechanism is identical in both cases; only the value of the threshold differs. The reduced value is marked in the source as a temporary test setting, to be restored to the repository limit for normal use.

5.4.2 Repository Deposit and Linking

Once a dataset has been prepared, each part is deposited to Zenodo as its own record through the repository's API. Every record receives the metadata confirmed by the researcher in the review step, adjusted so that each part's description states which part it is and how many parts the dataset comprises in total. This means that a person who lands on any single record can see that it is one piece of a larger whole and how many pieces exist.

Each part's title is suffixed with its position, in the form "Part i of n", and its description is prefixed with a note stating that the dataset was split into n parts and listing the Digital Object Identifiers of the sibling parts. The records are then linked to one another using Zenodo's related-identifier mechanism, with each part declaring a `references` relation to every other part. The `references` relation is used because the version of the Zenodo deposit interface in use does not support a more specific "is part of" relation; it is sufficient for the purpose, since it lets the parts be discovered from one another and is what the reproduction workflow follows to gather every part of a dataset starting from a single reference. Linking the records, rather than leaving them as unconnected deposits, is what turns a split dataset back into a single citable object.

Figure 5.5 summarises the full path, from a dataset directory through the size decision and, when splitting is required, the production of linked Zenodo records, to the reassembly performed by the reproduction workflow.

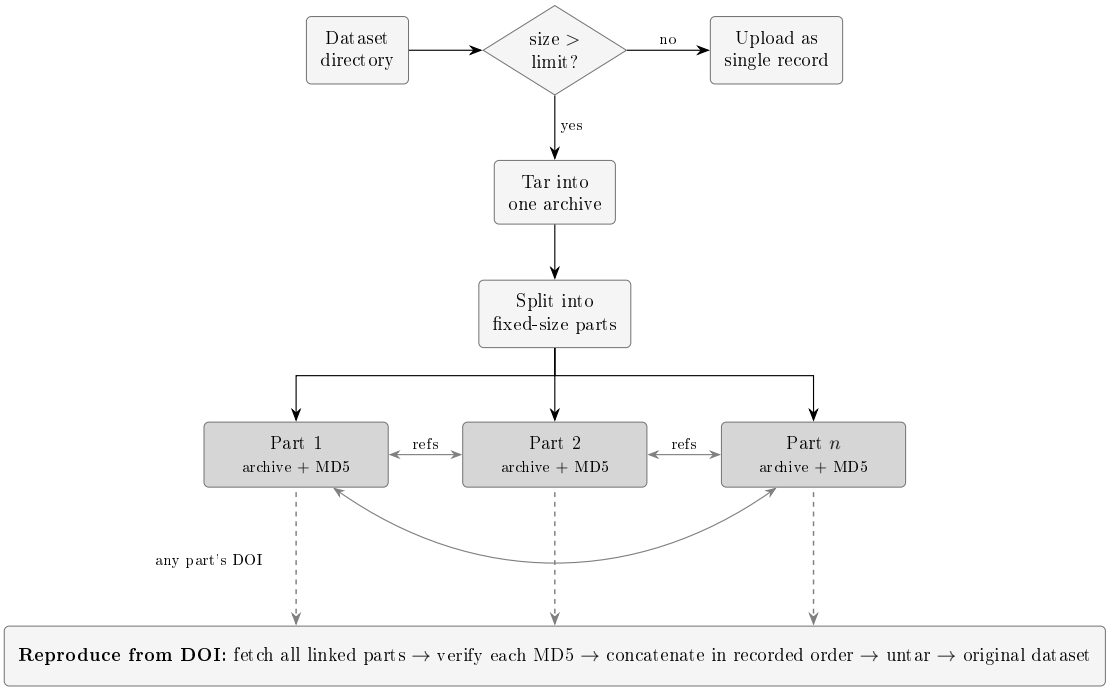


Figure 5.5: The data layer's handling of a dataset: chunking, linked deposit, and verified reassembly at reproduction time.

Whether researchers can rely on this splitting and linking in practice, and how the experience of publishing a large dataset this way compares with doing it by hand, is examined in the user

study in Chapter 6.

5.4.3 The Reproducibility Manifest

The structure that ties the data layer to the reproduction workflow is a manifest file, written when an experiment with externalised data is packaged and stored as part of the reproducibility artefact. The manifest was designed for this work, and its fields follow from a single requirement: the reproduction workflow must be able to rebuild the dataset from the artefact alone. It is therefore the authoritative record of how a dataset was handled: it states the strategy used, the size and digest of the whole archive, and, for each part, the repository record it was deposited to, its order in the sequence, and its digest. It also carries the ordered steps needed to put the dataset back together. Figure 5.6 shows an example, lightly abbreviated, produced for a dataset that was split into two parts.

Two things about the manifest are worth drawing out. First, it makes the dataset self-describing from the artefact alone: a reproduction needs nothing beyond the manifest to locate every part, check it, and rebuild the original. Second, the reconstruction it specifies is verified at two levels, each part against its own digest and the reassembled archive against the digest of the original, so that a corrupted or out-of-order reconstruction is detected rather than silently used. The retrieval itself is performed with `rclone`,[8] a tool chosen for two reasons. It supports resumable and parallel transfers, which keeps the provisioning of large datasets practical over an ordinary network connection, where a plain sequential download would be slow and fragile. It also offers a backend that can mount a repository record directly by its identifier, allowing the data to be read without first copying it in full; the platform uses this mount path where possible and falls back to a verified download otherwise. The same approach to mounting repository data is used by RenkuLab, one of the platforms reviewed in Chapter 3.

[8] `rclone` is an open-source command-line program for managing files on cloud and remote storage, available at `https://rclone.org`. Its DOI backend, which addresses repository records directly, was introduced in version 1.67.

```
{
  "manifest_version": "1.0",
  "project_uuid": "demo_code1_0405_1027",
  "data_strategy": "chunk_and_externalize",
  "dataset": {
    "external": true,
    "total_size_bytes": 1560640,
    "tar_md5": "18b2cdc1...ecc5",
    "chunked": true,
    "num_chunks": 2,
    "depositions": [
      { "doi": "10.5281/zenodo.20021706", "chunk_id": 1,
        "filename": "dataset_chunk_001.bin",
        "md5": "5b3a84d2...95fe", "size_bytes": 1048576 },
      { "doi": "10.5281/zenodo.20021708", "chunk_id": 2,
        "filename": "dataset_chunk_002.bin",
        "md5": "46d56b4d...4ae0", "size_bytes": 528384 }
    ]
  },
  "reconstruction": {
    "method": "rclone_copy",
    "steps": [
      "Download each chunk via rclone or the Zenodo API.",
      "Verify each chunk MD5 against the manifest.",
      "Concatenate chunks in order to rebuild dataset.tar.",
      "Verify reassembled tar MD5 against tar_md5.",
      "Extract: tar xf dataset.tar -C /data",
      "Mount /data as a read-only volume."
    ]
  }
}
```

Figure 5.6: An example reproducibility manifest for a dataset split into two parts, with hash values abbreviated. The manifest records the deposition, order, and digest of each part, together with the ordered steps for retrieval, verification, reassembly, and mounting. It is the contract the reproduction workflow follows.

5.5 Language Model Integration

```
You are a research data curator creating Zenodo deposit metadata.
You will be given a file inventory and data samples from a
scientific experiment dataset, plus code context (README, script
names, imports).
Goal: infer metadata suitable to CREATE a Zenodo deposit for this
dataset.

STRICT RULES:
- Return ONLY valid JSON.
- Do NOT invent DOIs, ORCIDs, URLs, grant IDs, or licenses.
- If unsure about a field, set it to the literal string
  "<TOBeFilledByUser>".
- access_right: use "restricted" unless data samples or README
  clearly indicate open sharing.
- The description must read as if written by the dataset author:
  describe what the data contains, its format, value ranges,
  measurement type, and scientific domain.
- Do NOT reference any code files, script names, or how to run
  anything in the title or description.
- ALWAYS include a 'keywords' array with 3-8 relevant terms
  inferred from the data content, file names, formats, and
  scientific domain.
... (full prompt in the platform source)
```

Figure 5.7: Condensed extract of the prompt used to infer dataset metadata in the data management workflow. The model receives a file inventory and content samples from the uploaded dataset and must return the metadata as structured fields, with unknown values marked for the researcher to complete and a restricted access right unless the data itself indicates open sharing.

Chapter 4 identified the points at which the platform calls the language model. This section describes how those calls are constructed, with attention to the two that produce repository metadata, since those are the calls evaluated in the user study.

For dataset metadata inference, the backend assembles a prompt from a sample of the uploaded dataset. The sample includes file names, formats, and excerpts of content drawn from the dataset's files, together with any descriptive material such as a README when one is present. The model is asked to return the repository metadata as structured fields, so that each field can be placed directly into the corresponding input of the metadata editor. Figure 5.7 shows the prompt in condensed form. The prompt constrains the model against the failure modes discussed in Section 2.4: it may not invent identifiers or licences, must mark uncertain fields for the researcher to complete, and defaults to restricted access unless the data itself indicates open sharing.

Artefact metadata inference works the same way but draws its prompt from the packaged experiment rather than from a dataset: the project's code, its structure, and the outputs it produced. In both cases the model's output is a draft. It is rendered into the editable form described in Chapter 4, the researcher reviews and corrects it, and only the confirmed version is sent to the

repository. No metadata reaches Zenodo without passing through this review. How good these drafts are in practice, and how much editing researchers feel they need to do before trusting them, is one of the questions the user study in Chapter 6 sets out to answer.

A separate use of the model supports output identification. When the platform needs to know which files an experiment produces as its results, it combines a scan of the code for file paths with a reconciliation step in which the model resolves ambiguous or computed paths into a concrete list of expected outputs. The researcher confirms the result. This keeps output detection independent of the programming language of the experiment, since it does not depend on language-specific conventions for where results are written.

One property of the metadata inference should be stated openly. It works by sending a sample of the dataset, including file names and excerpts of content, to a hosted third-party API, and this transmission happens before the researcher confirms anything for publication. For the datasets the workflow is built around, data the researcher intends to make publicly available, this adds no meaningful exposure, since the same content is about to be published in full. Data that must not leave the researcher's control, however, should not be passed through the workflow in its current form. A natural extension would be to let the researcher decline inference at upload and complete the metadata form directly, so that no dataset content is transmitted; since the editable form already exists, this is a small change, and it is left as future work.

5.6 Reproduce from DOI

The reproduction workflow takes a single Zenodo DOI and reconstructs a runnable experiment from it. The backend resolves the identifier through the Zenodo API to locate the published artefact and downloads it. From the artefact it restores the computational environment, using the same SciRep machinery that produced the environment originally, so that the container the experiment runs in matches the one in which it was packaged.

If the artefact's data was externalised at packaging time, the workflow follows the manifest described in Section 5.4.3 to retrieve it. Working from the manifest's list of depositions, it gathers every part from the repository, verifies each against its recorded MD5 digest, and reassembles them in their recorded order into the dataset the original execution used. The reconstructed dataset is supplied to the container, the experiment is executed, and the outputs are presented for comparison with the originals. Because retrieval and reassembly are driven by the recorded part order and verified by MD5 digest, the reproduced experiment runs on data that is demonstrably identical to the data packaged with the original.

To validate this path in the prototype, the reproduction workflow was run after publication on the generated DOI for a split dataset. Starting from the DOI, the platform retrieved the published artefact, followed the manifest to locate the linked dataset parts, verified each part against its recorded MD5 digest, concatenated the parts in the recorded order, verified the reconstructed archive against the archive digest, and extracted the original dataset. This confirmed that the

manifest and related-record links were sufficient for automated reconstruction in the deployed prototype.

5.7 Deployment

For the user study, the platform was deployed to a virtual machine on Microsoft Azure[9] running Ubuntu 24.04, with Nginx (1.24) in the cloud so that participants could reach it through a web browser without any local installation.[10] The machine hosts the full stack: the Docker engine, the Python backend, and the built Angular frontend. The backend was configured with the credentials it needs to operate, namely an API key for the language model service and an access token for the Zenodo account used in the study, together with the host path under which project working directories are stored.

The machine is a modest cloud virtual machine, and its resources proved sufficient for the single-session use the study required. Deploying to a single reachable machine, rather than asking each participant to install and run the platform locally, removed a large source of variation from the user study. Every participant interacted with the same instance, on the same hardware, with the same software versions, which means differences observed between participants can be attributed to the interaction rather than to differences in their local environments. The deployment also made remote sessions possible, allowing participants to use the same platform regardless of location.

5.8 Summary

This chapter described how the design of Chapter 4 is implemented: a Flask backend and an Angular frontend retaining the original SciConv stack, a stage-based session model that gives the three workflows a shared structure, a universal conversational channel built on a bounded action protocol over a language model, a data layer that splits large datasets deterministically and links the resulting Zenodo records, language model calls that draft repository metadata for human review, a reproduction path that rebuilds and verifies an experiment from a single identifier, and a cloud deployment that gave the user study a single consistent instance to test against. The next chapter describes that study.

[9] https://azure.microsoft.com/

[10] The live prototype used during evaluation can be accessed at https://sciconv.spaincentral.cloudapp.azure.com/

Chapter 6

Evaluation

This chapter describes the user study conducted to evaluate SciSuite's data-management mode. The goal was to determine whether the mode lowers the cognitive effort of publishing research datasets to Zenodo and improves perceived usability relative to the standard manual process. The chapter presents the research questions, the hypotheses and variables, the study design, the participants, the materials, the procedure, the measurement instruments, and the analysis plan. The results are reported in Chapter 7.

6.1 Research Questions

The evaluation is organised around the research questions defined in Section 1.4, which are not restated here. RQa is addressed through the metadata participants accepted in the SciSuite condition and the post-questionnaire trust items. RQb is addressed through the System Usability Scale and the NASA Task Load Index, together with task completion. RQc is addressed in two ways. The user study directly evaluates the publication side of the question: whether participants can use SciSuite to publish an oversized dataset as linked repository records, and how the resulting records compare with those produced manually through Zenodo. The reconstruction and verification side of the question is evaluated at the implementation level through the manifest and reproduce from DOI workflow described in Sections 5.4.2, 5.4.3 and 5.6, rather than as a separate participant task. The three sub-questions together provide the evidence from which RQ is answered.

6.2 Hypotheses and Variables

The two primary measures are perceived usability and cognitive workload, each compared between the two conditions. This gives two pairs of hypotheses.

For usability, the null hypothesis H_0^u is that there is no systematic paired difference in System Usability Scale (SUS) score between SciSuite and the manual Zenodo workflow; the alternative

H_1^u is that such a difference exists. For workload, the null hypothesis H_0^e is that there is no systematic paired difference in NASA Task Load Index (NASA-TLX) score between SCISUITE and the manual workflow; the alternative H_1^e is that they differ.

The independent variable is the tool used to publish the dataset, with two treatments: SCISUITE and the manual Zenodo workflow. The dependent variables are the SUS score, for usability, and the weighted NASA-TLX score, for workload. Task completion, time, and the metadata participants accepted are recorded as secondary observations.

6.3 Study Design

The study used a within-subjects design in which every participant completed both conditions: publishing a dataset with SCISUITE and publishing a dataset manually through the Zenodo web interface. A within-subjects design isolates the difference attributable to the tool rather than to individual variation, and uses a limited participant pool more efficiently than a between-subjects design, which would split it into two groups [65]. Its main cost is the risk of order effects, where experience in the first condition carries over to the second, a known peril of crossover designs in software engineering experiments [60].

To control for order effects, both the order of the tool conditions and the dataset-condition pairing were counterbalanced. Participants with odd identifiers used SCISUITE first and those with even identifiers used the manual Zenodo workflow first. The dataset assignment also alternated across participants: when one participant used SCISUITE with the weather dataset and the manual workflow with the biodiversity dataset, the next participant used the manual workflow with the weather dataset and SCISUITE with the biodiversity dataset. This ensured that each dataset appeared in both tool conditions, so the comparison was not tied to one fixed dataset-tool pairing.

Each condition involved publishing one large dataset: a weather dataset in one condition and a biodiversity dataset in the other. Both datasets exceeded the per-record size threshold, so both required the large-dataset handling that distinguishes the two tools: SCISUITE splits and links the records automatically, while the manual workflow requires the participant to split the dataset and link the resulting records through Zenodo's related-works field. Zenodo's real per-record limit is 50 GB, which would make a genuinely large upload impractical in a session, so the threshold was lowered to 1 MB for the study; the workflow experienced is otherwise the same as for a multi-gigabyte dataset, and the tutorials explained this adjustment.

6.3.1 Choice of the Comparison Baseline

The manual Zenodo workflow was chosen as the comparison condition in preference to the other platforms reviewed in Chapter 3, and the choice needs justification. The task under evaluation is the publication of a dataset that exceeds a repository size limit, including its metadata and the linking of the resulting records, and none of the reviewed systems performs it: the comparative analysis of Section 3.6 found no tool that splits a dataset across linked repository records to satisfy

a per-record limit or assists with metadata creation. Code Ocean, the baseline of the original SCICONV evaluation [15], was considered for an earlier design of this study and set aside for two reasons: a comparison on packaging would have evaluated the capability inherited from the original system instead of the contributions of this dissertation, and Code Ocean does not expose an external data storage workflow on the access tier available to participants, so the large-dataset task could not have been performed in that condition at all. The manual Zenodo web interface is how researchers currently publish datasets to this repository, so it is the state of practice for the task. Using the same repository in both conditions also means the two conditions share infrastructure, and observed differences are attributable to the interaction and the assistance, not to the repository itself.

The baseline was therefore not selected in the expectation that an automated tool would outperform an unassisted workflow; it was selected because no assisted alternative exists to compare against. The asymmetry between the conditions is the object of the study, not a bias in its design: the research questions ask whether language-model assistance and hybrid interaction reduce the burden of current practice, and the outcome was not a foregone conclusion. Automation can lower usability as easily as raise it, if generated metadata is poor enough to need extensive correction, if automatic chunking produces records participants do not trust, or if the automatic steps make the process opaque. Whether the platform's assistance in fact helped, by how much, and on which dimensions of workload, are the empirical questions the study was run to answer.

6.4 Participants

Participants were recruited from master's and doctoral students and research staff, both at the Faculdade de Engenharia da Universidade do Porto (FEUP) and through the study author's wider academic networks, which extended recruitment to other institutions. The study targets this population because students and early-career researchers are prospective publishers of research data and match the profile the platform is designed for: researchers who work within a scientific domain rather than in software engineering. The only requirements were that a participant was enrolled in or employed by a research institution and could follow written instructions in English. Prior experience with data repositories was not required; the pre-questionnaire recorded each participant's familiarity. Each participant received an anonymous identifier.

As described in Chapter 7, recruitment proved to be the main constraint on the study.

6.5 Materials

6.5.1 Datasets

Two datasets were prepared, both as CSV time series with an accompanying README. The weather dataset contains meteorological observations from several stations in the Porto region, and the biodiversity dataset contains bird observation records from the same region. Both were

sized to exceed the study's 1 MB threshold so that they triggered the large-dataset handling in both conditions: in SCISUITE the dataset is split into linked records automatically, and in the manual condition the participant must split and link it.

6.5.2 Platform and Environment

SCISUITE was deployed on a cloud virtual machine and accessed by participants through a web browser, so that every participant used the same instance and no local installation was needed. The manual condition used the live Zenodo platform with a dedicated study account. Records created during sessions were set to restricted access, and drafts were removed afterwards.

6.5.3 Study Materials

The study used a written consent form; two tutorials, one per condition, giving step-by-step instructions; a pre-questionnaire collecting background and prior experience; a post-questionnaire collecting open-ended feedback; a structured observation sheet used during each session; and a session guide describing the evaluation protocol. The tutorials instructed participants to enter "Anonymous" as the creator name and explained the 1 MB threshold. The anonymised study data, study materials, task datasets, and analysis scripts used in the evaluation are archived in a Zenodo reproducibility package at `https://doi.org/10.5281/zenodo.21284382`.

6.6 Procedure

Each session was conducted remotely, with the participant sharing their screen so that observations could be recorded. A session followed the same sequence for every participant:

1. Introduction and consent.
2. Pre-questionnaire.
3. Tutorial for the first condition.
4. Publishing the dataset in the first condition.
5. SUS and NASA-TLX for the first condition.
6. Tutorial for the second condition.
7. Publishing the dataset in the second condition.
8. SUS and NASA-TLX for the second condition.
9. Post-questionnaire.

The SUS and NASA-TLX questionnaires were administered immediately after each condition rather than at the end of the session, which reduces retrospective recall bias in the ratings. Participants were told throughout that the tools, not they, were being evaluated.

6.7 Measurement Instruments

6.7.1 System Usability Scale

Perceived usability was measured with the SUS, a ten-item Likert questionnaire developed by Brooke [7]. Each item is rated from Strongly Disagree to Strongly Agree, with odd-numbered items positively worded and even-numbered items negatively worded. Each item is converted to a 0–4 scale, summed, and multiplied by 2.5, giving a score between 0 and 100, with a separate score for each condition. Scores can be read against the grade boundaries proposed by Bangor, Kortum, and Miller [3], where 85–100 is excellent, 72.5–84.9 good, 52.5–72.4 acceptable, 38–52.4 poor, and below 38 not acceptable; these boundaries are used throughout Chapter 7 when interpreting the scores.

6.7.2 NASA Task Load Index

Cognitive workload was measured with the weighted NASA-TLX, developed by Hart and Staveland [29]. It assesses six dimensions: mental demand, physical demand, temporal demand, performance, effort, and frustration. Each is rated from 0 to 100, and participants then make 15 pairwise comparisons that weight the dimensions; the overall score is the weighted average. The study used an HTML implementation of the procedure that participants completed in their browser after each condition, producing an overall score and a per-dimension breakdown.

6.7.3 Observation Sheet and Questionnaires

The observation sheet was used to record task completion, the time taken, and behavioural detail such as stalls, errors, and questions asked during the session. For the SciSuite condition it also recorded how participants responded to the generated metadata and to the multi-record result; for the manual condition it recorded how participants handled splitting and linking. The pre-questionnaire collected background and prior experience, used to describe the sample. The post-questionnaire collected open-ended feedback, used to interpret the quantitative results rather than as primary evidence.

6.8 Analysis Plan

The SUS and NASA-TLX scores were compared between conditions as paired observations, since each participant provided both. The paired differences were checked for normality, and because the sample was small and not normally distributed, the Wilcoxon signed-rank test was used in preference to the paired t-test [63]. The significance threshold was $\alpha = 0.05$, and the test statistic and p-value are reported for each primary measure. The effect size is reported as the rank correlation $r = Z/\sqrt{N}$, where Z is the standardised test statistic and N is the number of non-zero paired differences; by the thresholds of Fritz, Morris, and Richler [25], values of r at or above 0.5 indicate a large effect.

Task completion, time, and the metadata participants accepted in the SCISUITE condition were summarised descriptively, and the open-ended responses were read thematically. As Chapter 7 explains, the small final sample limits what the inferential test can establish, so the results are reported as a preliminary evaluation and the analysis rests on the consistency and size of the effect as much as on the test.

As an exploratory check, the published records were also assessed with F-UJI [20], an automated tool for evaluating the FAIRness of research data records. These scores were used only to interpret record-level metadata properties and not as a primary comparison between conditions, because the tool mainly checks the presence of machine-detectable FAIR elements rather than the substantive quality of descriptions, keywords, or inter-record links.

6.9 Ethical Considerations

Participation was voluntary. Each participant signed a consent form before the session and was free to withdraw at any time without penalty, and was told that the tools, not their own abilities, were being evaluated. No personally identifiable information was stored in the data files: participants were given anonymous identifiers, and any names on the consent forms were kept separately. No audio or video was recorded. Records created during sessions were set to restricted access, and the tutorials instructed participants to publish under the creator name "Anonymous". In the SCISUITE condition, metadata inference used a hosted language-model API that received file names and excerpts from the uploaded study datasets. The datasets used in the evaluation were prepared for the study and did not contain personal or sensitive information.

Chapter 7

Results and Discussion

This chapter reports the outcome of the evaluation described in Chapter 6. It begins with the recruitment of participants, which proved to be the main constraint on the study, then describes the participants who completed it, presents the descriptive statistics for the two primary measures, reports the inferential tests, and analyses what the data shows. The chapter closes by relating the findings to the research questions and discussing the threats to validity.

7.1 Recruitment

Recruiting participants was the most difficult part of the evaluation, and it shaped the size of the study. Consequently, participants were recruited through direct contact with colleagues, students, and researchers, with recipients encouraged to share the invitation within their networks.

This difficulty is not unique to this study. The challenge of recruiting participants for human-centric computing research is a recognised problem in the field, to the point that it has been the subject of dedicated work on the strategies that make recruitment easier [39]. The session length, the requirement to share a screen, and the absence of any financial incentive all raised the cost of participation and lowered the response rate. As a result, the study was conducted with eight participants, fewer than initially planned. The sample is modest, and what this means for the strength of the conclusions is discussed in Section 7.8, but it proved sufficient for the difference between the two conditions to reach statistical significance on both primary measures. The one-on-one format of the sessions is one advantage the study design does retain despite this shortfall. Because each session was run individually and remotely, without requiring a fixed cohort or a scheduled group event, additional participants can be recruited and evaluated at any time without disrupting sessions already completed. Because the protocol uses individual remote sessions, the study design can be extended with additional participants in future work without changing the procedure. This does not remove the limitation of the current sample size, but it makes replication and extension straightforward.

7.2 Participants

Eight participants completed the study. All eight were master's students, drawn from a range of research areas and several countries, reflecting the broadened recruitment described above. Their research areas spanned information technology, information systems, animal science, information science, arts and digital media, informatics engineering, data compliance and governance, and artificial intelligence. Five worked or studied in Portugal, with one each in the United Kingdom, Nigeria, and Pakistan. Seven were male and one was female. Table 7.1 summarises the participants and their prior experience. When asked about the main challenges of publishing data, the most frequent answers were writing good metadata and descriptions, the technical difficulty of the tools, the time and effort involved, and handling large files. These responses align with the problems the platform was designed to address.

Table 7.1: Participant demographics and prior experience with research data publishing.

ID	Research area	Years	Country	Familiarity	Deposits
P1	Information Technology	< 2	United Kingdom	Somewhat	0
P2	Information Systems	< 2	Portugal	Very	5
P3	Animal Science	2–5	Nigeria	Somewhat	0
P4	Information Science	> 5	Portugal	Very	6
P5	Arts / Digital Media	< 2	Portugal	Heard of it	2
P6	Informatics Engineering	2–5	Portugal	Very	0
P7	Data Compliance & Governance	2–5	Portugal	Somewhat	2
P8	Artificial Intelligence	< 2	Pakistan	Somewhat	3

Each participant published one large dataset in each condition: a weather dataset in one condition and a biodiversity (bird observation) dataset in the other. As described in Section 6.3, the dataset-condition pairing was counterbalanced, so each dataset appeared in both the SciSuite and manual Zenodo conditions across the participant group. Both datasets exceeded the study's per-record size threshold, so both triggered the large-dataset handling: automatic chunking in the SciSuite condition, and manual splitting and linking in the manual condition. The study did not include a separate small-dataset task; the comparison is between the two conditions on the large-dataset publishing task.

One characteristic of the sample is worth noting for the analysis that follows. Participant P3 completed the session on a mobile device. This did not prevent the SciSuite condition from being completed, but it made the manual Zenodo condition impossible to finish, because the manual task requires splitting a file on the participant's own machine, which P3 could not do on a phone. P3 therefore has a complete SciSuite record but no manual record. This is itself an observation about the fragility of the manual workflow, and it is discussed in Section 7.5. For the paired statistical comparisons, P3 is necessarily excluded, leaving seven complete pairs.

7.3 Descriptive Statistics

The two primary measures are the SUS and the weighted NASA-TLX. For the paired comparisons, the seven participants who completed both conditions (P1, P2, P4, P5, P6, P7, and P8) are used. The SUS score for the SciSuite condition is also reported for all eight participants, since P3's SciSuite session was complete.

7.3.1 System Usability Scale

SUS scores range from 0 to 100. Across the seven participants who completed both conditions, SciSuite scored consistently and substantially higher than the manual Zenodo workflow. Every participant rated SciSuite above the manual process. The SciSuite scores ranged from 90 to 100 with a mean of 95.0, which falls in the "excellent" band of the adjective scale described in Section 6.7.1. The manual scores ranged from 7.5 to 85 with a mean of 52.9, which falls just above the boundary of the "acceptable" band. Including P3, whose SciSuite score was 57.5, the eight SciSuite scores have a mean of 90.3, still within the "excellent" band. Figure 7.1 shows them per participant.

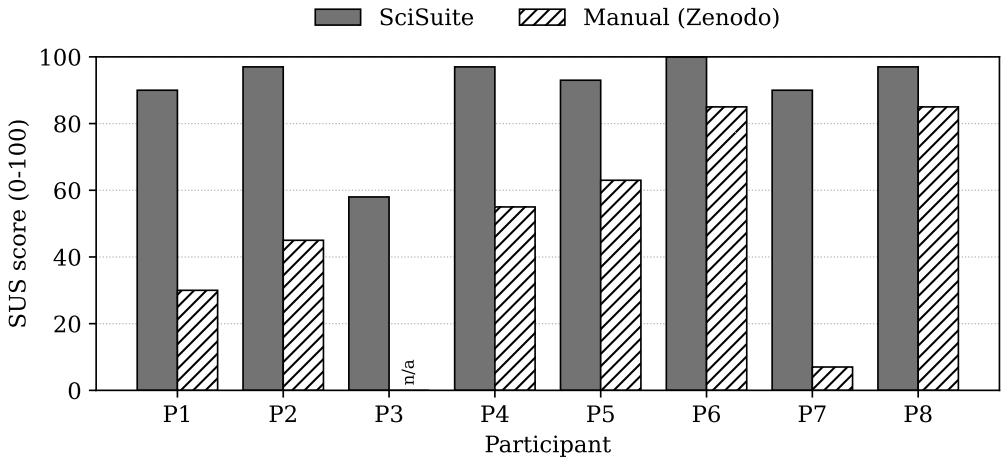


Figure 7.1: SUS score for each participant. Scores for SciSuite are shown in solid bars and the manual Zenodo workflow in hatched bars. P3 completed only the SciSuite condition.

7.3.2 NASA Task Load Index

NASA-TLX scores also range from 0 to 100, with lower scores indicating lower workload. The pattern mirrors the usability results. Every participant who completed both conditions reported lower workload for SciSuite than for the manual process. The SciSuite scores ranged from 5.0 to 35.3 with a mean of 15.8, while the manual scores ranged from 28.0 to 93.3 with a mean of 63.7. Including P3, whose SciSuite workload was 37.7, the eight SciSuite scores have a mean of 18.6, still well below the manual mean. Figure 7.2 shows them per participant.

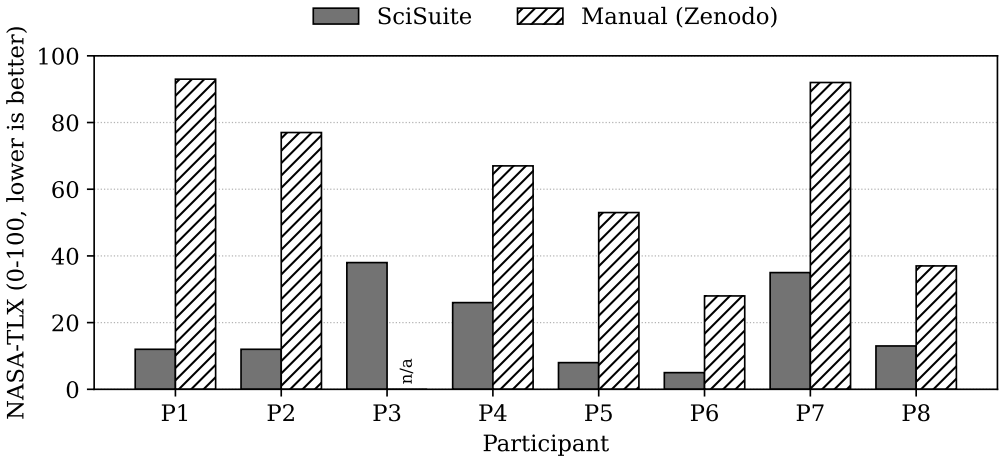


Figure 7.2: NASA-TLX overall workload for each participant (lower is better). Scores for SCISUITE are shown in solid bars and the manual Zenodo workflow in hatched bars. P3 completed only the SCISUITE condition.

The weighted NASA-TLX combines six dimensions, and breaking the overall score into them shows where the difference in workload comes from. Figure 7.3 reports the mean rating of each dimension across the seven paired participants. The manual workflow was rated higher on every dimension, with the largest gaps in mental demand, effort, and frustration, the three dimensions most directly tied to the difficulty of working out how to split and link a large dataset by hand.

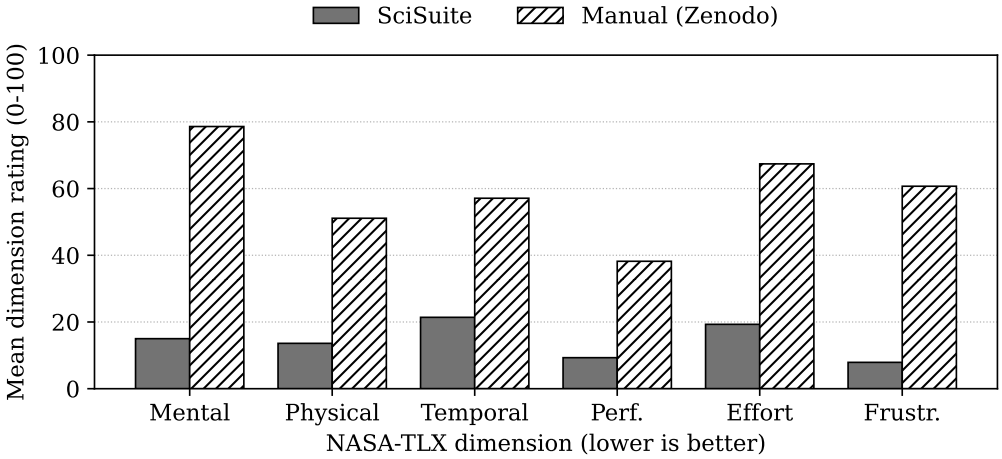


Figure 7.3: Mean NASA-TLX dimension ratings across the seven paired participants (lower is better). The manual workflow was rated higher on every dimension, with the largest gaps in mental demand, effort, and frustration.

7.3.3 Task Completion and Time

All eight participants completed the SCISUITE condition. Seven of the eight also completed the manual condition, although it took longer and produced more errors, as described below. P3 could

not complete the manual condition at all on a mobile device.

The two conditions also differed clearly in the time they took. Across the sessions, the manual Zenodo workflow took participants roughly 30 minutes to complete, while the SCISUITE workflow took roughly 10 minutes. These are approximate figures observed during the sessions rather than precisely instrumented timings, and the upload time itself is network-bound and excluded from this comparison, but the difference was large and consistent: publishing a large dataset took participants about three times as long by hand as it did with SCISUITE.

7.4 Hypothesis Testing

The two primary measures were compared between conditions using the procedure described in Section 6.8. The paired differences were first checked for normality, and because the sample is small and the distributions are not normal, the Wilcoxon signed-rank test was used in preference to the paired t-test [63].

For both measures, the two-sided Wilcoxon signed-rank test on the seven complete pairs returned $W = 0$ and $p = 0.0156$, which is below the conventional threshold of $\alpha = 0.05$. The null hypotheses of no difference in usability (H_0^u) and no difference in workload (H_0^e) are therefore rejected: the difference between SCISUITE and the manual workflow is statistically significant on both the System Usability Scale and the NASA Task Load Index. The test statistic of $W = 0$ reflects that the difference was not only significant but unanimous, with every one of the seven participants rating SCISUITE as more usable and as imposing lower workload than the manual process. The size of the difference is also large. Converting the test statistic to a standardised Z of -2.37 gives an effect size of $r = Z/\sqrt{N} = 0.89$ on both measures with $N = 7$ non-zero pairs, well above the 0.5 threshold for a large effect given by Fritz, Morris, and Richler [25]. This matches the descriptive picture: a gap of roughly 42 points in mean SUS and roughly 48 points in mean workload on the 0–100 scales.

The sample remains modest, and the threats to validity in Section 7.8 are careful about what can and cannot be concluded from seven pairs. But the result is a genuine rejection of the null hypothesis under the conservative two-sided test, not merely a favourable trend, and it is reinforced by the consistency of the direction across every participant and the magnitude of the difference. With seven pairs and a unanimous direction, this p-value is the smallest the two-sided test can return, which reflects the unanimity of the sample rather than a finer-grained estimate of the effect.

7.5 Analysis

Three things stand out from the data, beyond the headline scores.

The first is the consistency of the direction. On both measures, and for every one of the seven participants who completed both conditions, SCISUITE scored better than the manual workflow. The usability scores for SCISUITE clustered tightly in the excellent band, while the manual scores were spread across a much wider and lower range. The workload results show the same pattern

in reverse, with SCISUITE producing uniformly low workload and the manual workflow producing high and variable workload. The per-dimension breakdown locates the difference in mental demand, effort, and frustration, which is consistent with the platform's purpose: the manual workflow asks the participant to work out how to split and link a large dataset, and SCISUITE removes that work.

The second is the mobile case, P3. That this participant could complete the SCISUITE condition on a phone but could not complete the manual condition at all is a concrete illustration of the gap between the two workflows. The manual large-dataset task depends on the participant having a desktop environment in which to split a file, whereas SCISUITE performs the split server-side and asks nothing of the participant's device. The manual workflow is not merely harder; in some settings it is not possible at all, while the assisted workflow remains usable.

The third concerns the quality of what participants produced in the manual condition, which the observation of their published records revealed. Several patterns recurred. In the SCISUITE condition, every participant accepted the metadata the platform generated, and only one participant made any change at all, adding a few extra keywords; the generated titles, descriptions, and keywords were otherwise taken as they were. In the manual condition, by contrast, the descriptions participants wrote by hand were typically short and thin, lacking the detail that makes a dataset findable and reusable. In the manual large-dataset task, most participants who split the data linked the resulting records in one direction only, adding a related-work reference from the second record to the first but not the reverse, so that the connection could not be followed both ways. The relation type they chose was also frequently wrong: in the record shown in Figure 7.5, for example, the two parts are linked with the relation "is previous version of", which describes successive versions of the same dataset rather than two complementary parts of one split, and is therefore misleading about how the records fit together. Participants also generally did not indicate in the title or description which part of the split a record represented, leaving records that gave no signal that they were one piece of a larger whole. SCISUITE, by contrast, writes the cross-references in both directions and marks each record with its position in the sequence automatically. These are exactly the kinds of error that undermine the findability and reconstructability of split datasets, and they appeared precisely where the platform automates the work and the manual process leaves it to the researcher. Figure 7.4 and Figure 7.5 show a record from each condition side by side, and the difference is visible at a glance: the SCISUITE record states which part it is, explains the split, and carries a full keyword set, while the manual record gives a short description, no keywords, and a single one-directional link.

Taken together, these observations suggest that the benefit of the platform is not only that participants found it easier and less effortful, but that the artefacts it produced were more complete and better linked than those participants produced by hand.

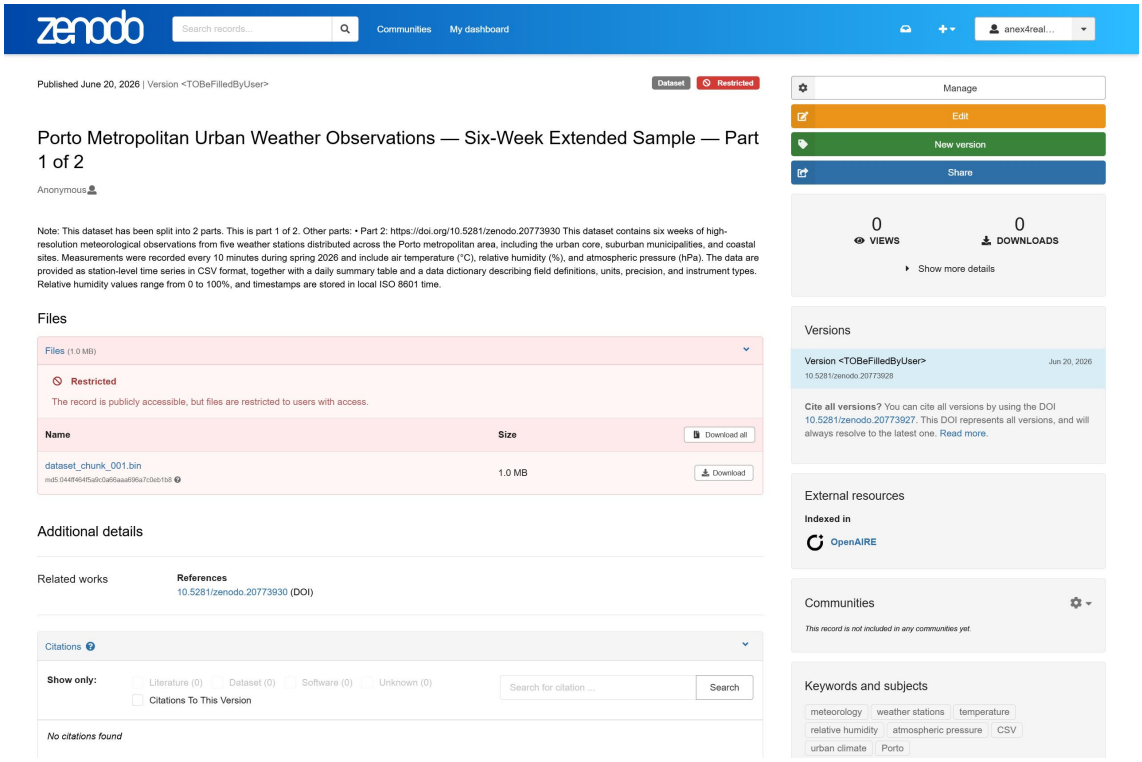


Figure 7.4: A dataset record published through SciSuite. The title carries the automatically added “Part 1 of 2” marker, the description opens with a generated note stating that the dataset was split and giving the DOI of the other part, and a full set of keywords is attached.

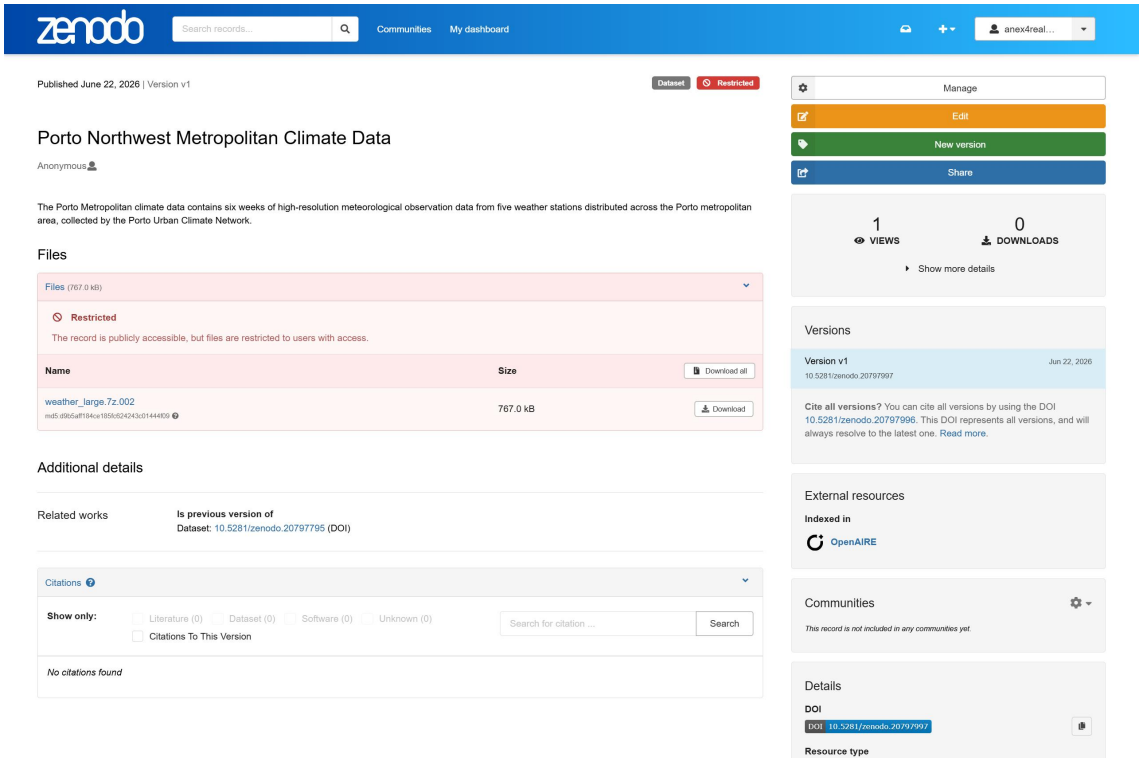


Figure 7.5: A dataset record published manually to Zenodo. The title gives no indication that the record is one part of a split dataset, the description is short, no keywords are attached, and the related-work link points in one direction only.

7.6 Discussion

The results can be read against the bodies of work this dissertation builds on, and doing so shows that the study touches each of them rather than standing apart.

The clearest connection is to the FAIR principles. Findability and reusability depend on rich metadata, and the consistent finding in the literature is that metadata is the part of FAIR practice that researchers neglect, because writing good descriptions is effort that brings the depositor no immediate reward [9, 64]. The records participants produced by hand in this study are a direct illustration of that account: their descriptions were short, their keywords absent, and their cross-references incomplete and mis-typed. The records SCISUITE produced were the opposite, and the difference did not come from participants trying harder, since they accepted the generated metadata almost without change. The platform shifts the cost of good metadata from the researcher to an automated step, which is precisely the intervention the FAIR literature implies is needed but which repositories themselves do not provide.

The usability and workload results speak to the work on hybrid interfaces. The premise of that work is that conversation and direct manipulation have complementary strengths, and that neither modality alone serves a structured task well [23, 38]. The manual condition in this study is, in effect, a pure direct-manipulation interface: a web form with no guidance. Its high and variable workload, and the errors participants made in it, are consistent with the weakness that literature attributes to unaided graphical forms for tasks where the user does not already know the correct structure. SCISUITE's lower and tighter workload scores are consistent with the claim that pairing a conversational surface with structured controls helps users through exactly such tasks. The study also addresses a gap noted in that same literature: most hybrid-interface evidence comes from prototypes and lab tasks [32, 50], whereas this evaluation tested a hybrid interface embedded in a working platform against a real repository workflow, which is the kind of deployment evidence the field has lacked.

The findings also bear on the long-standing argument about conversational versus direct-manipulation interaction. Direct manipulation was defended on the grounds that visible objects and immediate feedback make correction cheap [53], while conversation has the opposite profile, expressive but opaque, and demanding when the user must phrase a precise request [18]. The metadata editor in SCISUITE is where these meet: the language model drafts the structured record, and the form lets the participant correct it directly. That participants accepted the drafts almost unchanged suggests the combination worked as intended, with the conversation doing the drafting and the structured view available for the correction that, in this case, was rarely needed.

This connects to the question of human-AI collaboration. The accepted mitigation for the unreliability of language-model output is human oversight: the model proposes and a person who understands the material reviews before anything takes effect [1, 33]. SCISUITE is built on this pattern, and the study shows the division of labour holding up in practice: the model produced metadata good enough to publish, and the participant remained the one who confirmed it. The near-total acceptance is encouraging for the pattern, but it also carries a caution the literature

would raise, namely that acceptance is not the same as correctness; a participant who trusts a fluent draft may approve it without checking it closely, and the study measured what participants accepted rather than whether the accepted metadata was independently correct.

The published records were also assessed with F-UJI[1] 3.5.1 [20], under the FsF v0.8 metric set. Because every record is deposited on Zenodo, the Accessible, Interoperable, and Reusable scores were identical across records, reflecting the structural properties Zenodo provides. The only score that varied was Findability, and it varied solely with the presence of keywords: a record carrying a single keyword but a one-line description scored higher (86 percent) than a record with no keywords but a full, informative description (71 percent). This inversion, in which the less informative record scores higher, shows that the metric rewards the presence of metadata fields and not the quality of their content. F-UJI cannot separate the metadata the two workflows produce, since the differences between them lie in description quality, keyword completeness, and the correctness of inter-record links, none of which this metric assesses. A per-condition FAIR comparison was not drawn from these scores, particularly as participants published different datasets in the two conditions; the workflow differences are documented in the record-level comparison above.

Finally, the results extend the reproducibility line of work the platform grows from: the same conversational, oversight-based approach that lowered the barrier to packaging experiments also lowers the barrier to publishing and correctly linking the datasets those experiments depend on, which the reproducibility literature identifies as the harder remaining problem for data-intensive work.

7.7 Answering the Research Questions

The evidence collected addresses the research questions as follows.

RQa (metadata quality and trust). In the SciSuite condition, every participant accepted the metadata the platform generated, and only one participant made any change, adding a few extra keywords. The post-questionnaire responses singled out the automatic inference of titles, descriptions, and keywords as a strength. In contrast, the descriptions participants wrote by hand in the manual condition were consistently thin. This indicates that the generated metadata was not only acceptable to participants, who published it almost entirely unchanged, but in practice more complete than what they produced unaided.

RQb (hybrid interface effectiveness). On both primary measures the hybrid interface outperformed the manual workflow for every participant who completed both conditions, with a mean SUS of 95.0 against 52.9 and a mean workload of 15.8 against 63.7, and it did so in roughly a third of the time. The difference was statistically significant on both measures ($p = 0.0156$), and the interface was rated in the excellent usability band. The evidence supports the effectiveness of the hybrid interface.

[1] https://f-uji.net/index.php

RQc (chunking and reconstruction). Every participant successfully published the large dataset using SCISUITE's automatic chunking, while the manual condition produced one-directional links, often with an inappropriate relation type, and records that did not indicate their position in the split, and in one case could not be completed at all. The platform's automatic splitting and bidirectional linking produced correctly connected records where the manual process did not, supporting the value of the chunking mechanism. The reconstruction and verification side of RQc was also checked at implementation level by running the reproduce-from-DOI workflow on a generated DOI. The platform retrieved the linked parts, verified their digests, concatenated them in order, verified the rebuilt archive, and extracted the original dataset. Thus, the user study evaluates the usability and record quality of the publication process, while the implementation validation confirms the technical reconstruction path.

RQ (overarching). The combined evidence from the three sub-questions points consistently in the same direction: the platform delivered the data-management workflow at a usability and workload level significantly above the manual baseline, with more complete metadata and better-linked records. The effect was statistically significant across the completed participant pairs and large in magnitude, although it was obtained from a modest sample.

7.8 Threats to Validity

The threats to the validity of this study are organised following the categorisation of Wohlin et al. [65] into internal, external, construct, and conclusion validity.

7.8.1 Internal Validity

The within-subjects design carries a risk of learning effects, since experience in the first condition can carry over to the second. This was addressed by counterbalancing: participants with odd identifiers used SCISUITE first and those with even identifiers used the manual workflow first. With seven complete pairs the balance is approximate, so a residual order effect of the kind Vegas, Apa, and Juristo [60] describe for crossover designs cannot be ruled out. Descriptively, the direction of the difference was the same for participants in both orders, though the subgroups are too small for a formal test of order effects. Because condition order and dataset–condition pairing varied together, their individual influences cannot be separated in this sample. A second threat is the device difference of participant P3, who took part on a mobile phone; because this affected only the manual condition, P3 is excluded from the paired comparisons and reported as a qualitative observation rather than as paired evidence.

7.8.2 External Validity

All participants were master's students, recruited partly through personal networks, which may bias the sample toward people familiar with the study author or generally receptive to new technologies, and bounds how far the results generalise to the wider population of researchers. The

sample is diverse in research area and country, which supports the generality of the direction of the effect but adds variation that a sample of this size cannot absorb. The study task used prepared datasets and a lowered size threshold, so behaviour at real dataset scale remains untested.

A further threat to external validity concerns the datasets used. The study was built around two datasets, one of weather observations and one of biodiversity observations, and every participant worked with these same two. Findings about how the two workflows handle large datasets are therefore grounded in the characteristics of these datasets, such as their file formats and the way their contents divide into parts, and may not extend to datasets with very different structures, for example many small files, deeply nested directories, or binary formats that do not compress. Two considerations limit this threat. The mechanism under test operates on the dataset as an opaque archive: it measures the packed size, splits the bytes into fixed-size parts, and records digests and order, so its behaviour does not depend on the internal structure or scientific domain of the data. The two datasets were also chosen to differ from each other in domain and in content, rather than being two instances of the same kind of data. Even so, a broader evaluation across a wider range of dataset sizes, formats, and structures would be needed to establish that the results hold in general, and this is noted as future work.

7.8.3 Construct Validity

Perceived usability and workload were measured with validated instruments, but the quality of the generated metadata was measured through acceptance rather than independent correctness: a participant who trusts a fluent draft may approve it without checking it closely, so acceptance is an imperfect proxy for quality. The manual-condition observations of metadata quality and record linking were recorded from the published records rather than scored against a predefined rubric, so they measure visible record properties rather than a formally defined construct.

7.8.4 Conclusion Validity

The most significant threat is the modest sample: seven complete pairs are enough for the two-sided Wilcoxon signed-rank test to reach significance, as it did, but the confidence intervals around the effect are correspondingly wide, and with a unanimous direction the obtained p-value is the smallest the test can return, so it reflects unanimity rather than a fine-grained estimate of the effect. The timing figures are approximate, observed during sessions rather than instrumented, with the network-bound upload time excluded, and the record-quality observations were not independently coded, which limits the reliability of those secondary measures.

Chapter 8

Conclusions and Future Work

This chapter summarises the work and its findings, states the contributions, and sets out the directions that follow from the study and its limitations.

8.1 Summary of the Work and its Findings

This dissertation started from three gaps left open by existing tools: reproducibility tools and data management tools are separate, the handling of datasets that exceed repository limits and the writing of their metadata fall entirely on the researcher, and the researchers who face this work are often trained in a scientific domain rather than in software engineering. The review in Chapter 3 confirmed that no examined system automates the handling of oversized datasets or assists with metadata creation, and that none combines conversational and structured interaction.

In response, the work built SCISUITE, an extension of the conversational reproducibility tool SCICONV [15] on the SCIREP engine [14]. The platform adds a standalone data management mode with language-model-drafted metadata, deterministic chunking with cross-linked Zenodo records for datasets over the deposit limit, a workflow that reproduces a published experiment from its identifier alone, and a hybrid interface in which conversation and structured controls are assigned stage by stage, with every model output reviewed and confirmed by the researcher before publication.

The platform was evaluated in a within-subjects user study with eight participants, each publishing a large dataset both through SCISUITE and manually through Zenodo. On the seven complete pairs, SCISUITE scored a mean SUS of 95.0 against 52.9 and a mean NASA-TLX workload of 15.8 against 63.7, with every participant favouring the platform on both measures and completing the task in roughly a third of the time; both differences are significant under the two-sided Wilcoxon signed-rank test ($p = 0.0156$). Participants accepted the generated metadata almost without change, and the records the platform produced were more complete and better linked than those produced by hand, which in the manual condition showed thin descriptions, missing keywords, and one-directional or mistyped links. This evidence answers the research question of Section 1.4 affirmatively within the study's scope: Within the scope of the evaluated task, the

hybrid platform achieved higher perceived usability, lower workload, and more complete visible record structure than the manual Zenodo baseline, although the modest sample size limits how far the finding can be generalised.

8.2 Contributions

The work makes the three contributions set out in Section 1.5: a unified platform integrating reproducibility packaging, reproduction from a persistent identifier, and standalone dataset publishing (Chapters 4 and 5); a Direct Data Management mode that drafts metadata, chunks oversized datasets, and cross-links the resulting records (Sections 5.4 and 5.4.1); and a hybrid interface design pairing a conversational surface with structured forms and explicit controls (Section 4.4).

8.3 Future Work

The limitations of the study are discussed in Section 7.8; the clearest way to strengthen the evidence is to replicate it with more participants and a wider range of disciplines. Beyond replication, several directions follow from the work. The platform could be evaluated on datasets from real research projects at the repository's actual size limit, to test the chunking and metadata features under the conditions they were built for. The metadata generation could be examined for correctness as well as acceptance, for instance by having domain experts judge the drafts. Support for repositories beyond Zenodo, such as Dataverse, would test how far the approach generalises across repository interfaces. The dependency on a single hosted language model could be reduced by comparing several models, including ones that run locally, which the implementation allows because model access passes through a single configurable point. A simpler complement would let the researcher decline metadata inference and complete the form by hand, so that sensitive dataset content is never transmitted. Finally, the hybrid interaction model is not specific to data publishing, and studying the same combination of conversation and structured control in other research workflows would show whether the design generalises beyond the setting in which it was built.

The work presented in this dissertation brings reproducibility packaging and research data management into a single workflow, lowers the technical barrier to both through conversation, and keeps the researcher in control of the structured output that conversation alone handles poorly.

References

[1] Saleema Amershi et al. “Guidelines for Human-AI Interaction”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI ’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–13. DOI: 10.1145/3290605.3300233.

[2] Monya Baker. “1,500 Scientists Lift the Lid on Reproducibility”. In: *Nature* 533 (2016), pp. 452–454. DOI: 10.1038/533452a.

[3] Aaron Bangor, Philip Kortum, and James Miller. “Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale”. In: *Journal of Usability Studies* 4.3 (2009), pp. 114–123.

[4] Lorena A. Barba. “Terminologies for Reproducible Research”. In: *arXiv preprint* (2018). eprint: arXiv:1802.03311.

[5] Brett K. Beaulieu-Jones and Casey S. Greene. “Reproducibility of computational workflows is automated using continuous analysis”. In: *Nature Biotechnology* 35.4 (2017), pp. 342–346. DOI: 10.1038/nbt.3780.

[6] Daniil A. Boiko et al. “Autonomous chemical research with large language models”. In: *Nature* 624.7992 (2023), pp. 570–578. DOI: 10.1038/s41586-023-06792-0.

[7] John Brooke. “SUS: A ‘Quick and Dirty’ Usability Scale”. In: *Usability Evaluation in Industry*. Ed. by Patrick W. Jordan et al. London: Taylor & Francis, 1996, pp. 189–194.

[8] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1877–1901.

[9] Alexander Bruns. “FAIR Data in Practice: Implementing the Principles for Reproducible Science”. In: *Data Intelligence* 2.1-2 (2020), pp. 50–59. DOI: 10.1162/dint_a_00031.

[10] Kyle Chard et al. “I’ll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets”. In: *2016 IEEE International Conference on Big Data (Big Data)*. 2016, pp. 319–328. DOI: 10.1109/BigData.2016.7840618.

[11] Ryan Chard et al. “Toward Enabling Reproducibility for Data-Intensive Research Using the Whole Tale Platform”. In: *Advances in Parallel Computing* (2020). DOI: 10.3233/APC200106.

[12] Fernando Chirigati et al. “ReproZip: Computational Reproducibility With Ease”. In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD ’16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 2085–2088. ISBN: 9781450335317. URL: https://doi.org/10.1145/2882903.2899401.

[13] Code Ocean. *Code Ocean*. 2018. URL: https://codeocean.com (visited on 07/02/2026).

[14] Lázaro Costa, Susana Barbosa, and Jácome Cunha. "A framework for supporting the reproducibility of computational experiments in multiple scientific domains". In: *Future Generation Computer Systems* 174 (2025), p. 107924. ISSN: 0167-739X. DOI: 10.1016/j.future.2025.107924. URL: https://www.sciencedirect.com/science/article/pii/S0167739X25002195.

[15] Lázaro Costa, Susana Barbosa, and Jácome Cunha. "Let's Talk About It: Making Scientific Computational Reproducibility Easy". In: *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2025. DOI: 10.48550/arXiv.2504.10134. URL: https://arxiv.org/abs/2504.10134.

[16] Lázaro Gabriel Barros da Costa. "Aiding Researchers Making their Computational Experiments Reproducible". PhD thesis. Faculdade de Engenharia da Universidade do Porto, 2025.

[17] Merce Crosas. "The Dataverse Network®: An Open-Source Application for Sharing, Discovering and Preserving Data". In: *D-Lib Magazine* 17 (Jan. 2011). DOI: 10.1045/january2011-crosas.

[18] Hai Dang et al. *How to Prompt? Opportunities and Challenges of Zero- and Few-Shot Learning for Human-AI Interaction in Creative Applications of Generative Models*. 2022. arXiv: 2209.01390 [cs.HC]. URL: https://arxiv.org/abs/2209.01390.

[19] Susan B. Davidson and Juliana Freire. "Provenance and scientific workflows: challenges and opportunities". In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. Vancouver, Canada: Association for Computing Machinery, 2008, pp. 1345–1350. ISBN: 9781605581026. DOI: 10.1145/1376616.1376772.

[20] Anusuriya Devaraju and Robert Huber. "An automated solution for measuring the progress toward FAIR research data". In: *Patterns* 2.11 (2021). DOI: 10.1016/j.patter.2021.100370.

[21] Paolo Di Tommaso et al. "Nextflow enables reproducible computational workflows". In: *Nature Biotechnology* 35.4 (Apr. 2017), pp. 316–319. DOI: 10.1038/nbt.3820.

[22] Bakinam T. Essawy et al. "A taxonomy for reproducible and replicable research in environmental modelling". In: *Environmental Modelling & Software* 134 (2020), p. 104753. ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2020.104753.

[23] Lukas A. Flohr et al. "Chat or Tap? – Comparing Chatbots with 'Classic' Graphical User Interfaces for Mobile Interaction with Autonomous Mobility-on-Demand Systems". In: *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*. MobileHCI '21. New York, NY, USA: Association for Computing Machinery, 2021. DOI: 10.1145/3447526.3472036.

[24] Juliana Freire, Norbert Fuhr, and Andreas Rauber. "Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041)". In: *Dagstuhl Reports* 6.1 (2016). Ed. by Juliana Freire, Norbert Fuhr, and Andreas Rauber, pp. 108–159. ISSN: 2192-5283. DOI: 10.4230/DagRep.6.1.108. URL: https://drops.dagstuhl.de/entities/document/10.4230/DagRep.6.1.108.

[25] Catherine O. Fritz, Peter E. Morris, and Jennifer J. Richler. "Effect Size Estimates: Current Use, Calculations, and Interpretation". In: *Journal of Experimental Psychology: General* 141.1 (2012), pp. 2–18. DOI: 10.1037/a0024338.

[26] Steven N. Goodman, Daniele Fanelli, and John P. A. Ioannidis. "What Does Research Reproducibility Mean?" In: *Science Translational Medicine* 8.341 (2016), 341ps12. DOI: 10.1126/scitranslmed.aaf5027.

[27] Yaroslav O. Halchenko et al. "DataLad: distributed system for joint management of code, data, and their relationship". In: *Journal of Open Source Software* 6.63 (2021), p. 3262. DOI: 10.21105/joss.03262.

[28] Michael Hanke et al. "FAIRly big: A Framework for Computationally Reproducible Processing of Large-Scale Data". In: *Scientific Data* 9 (2022). DOI: 10.1038/s41597-022-01163-2.

[29] Sandra G. Hart and Lowell E. Staveland. "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research". In: *Human Mental Workload*. Ed. by Peter A. Hancock and Najmedin Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183. DOI: 10.1016/S0166-4115(08)62386-9.

[30] Francisco Iniesto et al. "Creating 'A Simple Conversation': Designing a Conversational User Interface to Improve the Experience of Accessing Support for Study". In: *ACM Transactions on Accessible Computing* 16.1 (Mar. 2023). DOI: 10.1145/3568166.

[31] Vahid Jalili et al. "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2020 update". In: *Nucleic Acids Research* 48.W1 (June 2020), W395–W402. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa434.

[32] Roosa Järvinen. "Prompting the Future: Visual and Interaction Design Guidelines for Hybrid Graphical–Conversational and Generative AI User Interfaces". MA thesis. Aalto University, 2025. URL: https://aaltodoc.aalto.fi/items/83da5e49-3304-4808-a776-ddc29a32b911.

[33] Ziwei Ji et al. "Survey of Hallucination in Natural Language Generation". In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38. DOI: 10.1145/3571730.

[34] Ivo Jimenez et al. "The Popper Convention: Making Reproducible Systems Evaluation Practical". In: *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2017, pp. 1561–1570. DOI: 10.1109/IPDPSW.2017.157.

[35] Project Jupyter et al. "Binder 2.0 - Reproducible, interactive, sharable environments for science at scale". In: *Proceedings of the 17th Python in Science Conference*. Ed. by Fatih Akici et al. 2018, pp. 113–120. DOI: 10.25080/Majora-4af1f417-011.

[36] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific containers for mobility of compute". In: *PLOS ONE* 12.5 (2017), e0177459. DOI: 10.1371/journal.pone.0177459.

[37] Xinyi Liu, Tim Rietz, and Alexander Maedche. "Conversational versus graphical user interfaces: the influence of rational decision style when individuals perform decision-making tasks repeatedly". In: *Universal Access in the Information Society* 24.2 (2025), pp. 1157–1172. DOI: 10.1007/s10209-024-01122-1.

[38] Yixin Liu and Jean-Bernard Martens. "Conversation-based hybrid UI for the repertory grid technique: A lab experiment into automation of qualitative surveys". In: *International Journal of Human-Computer Studies* 184 (2024), p. 103227. DOI: 10.1016/j.ijhcs.2024.103227. URL: https://www.sciencedirect.com/science/article/pii/S1071581924000119.

[39] Kashumi Madampe et al. “Addressing the Agony of Recruitment for Human-centric Computing Studies”. In: *SIGSOFT Softw. Eng. Notes* 50.2 (June 2025), pp. 23–26. ISSN: 0163-5948. DOI: 10.1145/3721890.3721898. URL: https://doi.org/10.1145/3721890.3721898.

[40] Ravi Madduri et al. “Reproducible Big Data Science: A Case Study in Continuous FAIRness”. In: *PLOS ONE* 14.4 (2019). DOI: 10.1371/journal.pone.0213013.

[41] Dirk Merkel. “Docker: lightweight Linux containers for consistent development and deployment”. In: *Linux Journal* 2014.239 (2014).

[42] National Academies of Sciences, Engineering, and Medicine. *Reproducibility and Replicability in Science*. Washington, DC: The National Academies Press, 2019. DOI: 10.17226/25303.

[43] Chaitra Niddodi et al. “IOSPReD: I/O Specialized Packaging of Reduced Datasets and Data-Intensive Applications for Efficient Reproducibility”. In: *IEEE Access* 11 (Jan. 2023), pp. 1718–1731. DOI: 10.1109/ACCESS.2022.3233787.

[44] Brian A. Nosek et al. “Promoting an open research culture”. In: *Science* 348.6242 (2015), pp. 1422–1425. DOI: 10.1126/science.aab2374.

[45] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022, pp. 27730–27744.

[46] Roger D. Peng. “Reproducible Research in Computational Science”. In: *Science* 334.6060 (2011), pp. 1226–1227. DOI: 10.1126/science.1213847.

[47] Hans E. Plesser. “Reproducibility vs. Replicability: A Brief History of a Confused Terminology”. In: *Frontiers in Neuroinformatics* 11 (2017), p. 76. DOI: 10.3389/fninf.2017.00076.

[48] S. Pröll and A. Rauber. “Data Citation in Practice: Managing Persistent Identification for Evolving Data”. In: *Proceedings of the 13th International Conference on Digital Preservation (iPRES)*. 2017.

[49] Shweta Purawat et al. “A Kepler Workflow Tool for Reproducible AMBER GPU Molecular Dynamics”. In: *Biophysical Journal* 112.12 (2017), pp. 2469–2474. ISSN: 0006-3495. DOI: 10.1016/j.bpj.2017.04.055. URL: https://www.sciencedirect.com/science/article/pii/S0006349517305520.

[50] Renato Ribeiro. “Conversational Generative AI Interface Design: Exploration of a Hybrid Graphical User Interface and Conversational User Interface for Interaction with ChatGPT”. MA thesis. Umeå University, 2024. URL: https://www.diva-portal.org/smash/get/diva2:1872051/FULLTEXT02.pdf.

[51] Nicola Rieke et al. “The future of digital health with federated learning”. In: *NPJ digital medicine* 3 (2020), p. 119. DOI: 10.1038/s41746-020-00323-1.

[52] Geir K. Sandve et al. “Ten Simple Rules for Reproducible Computational Research”. In: *PLoS Computational Biology* 9.10 (2013), e1003285. DOI: 10.1371/journal.pcbi.1003285.

[53] Ben Shneiderman. “Direct Manipulation: A Step Beyond Programming Languages”. In: *Computer* 16.8 (1983), pp. 57–69. DOI: 10.1109/MC.1983.1654471.

[54] Ben Shneiderman et al. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 6th ed. Pearson, 2016.

[55] Stian Soiland-Reyes et al. “Packaging research artefacts with RO-Crate”. In: *Data Science* 5.2 (Jan. 2022), pp. 97–138. DOI: 10.3233/ds-210053.

[56] Philip B. Stark. “Before reproducibility must come preproducibility”. In: *Nature* 557 (2018), p. 613. DOI: 10.1038/d41586-018-05256-0.

[57] Victoria Stodden et al. “Enhancing Reproducibility for Computational Methods”. In: *Science* 354.6317 (2016), pp. 1240–1241. DOI: 10.1126/science.aah6168.

[58] Victoria Stodden, Jennifer Seiler, and Zhaokun Ma. “An empirical analysis of journal policy effectiveness for computational reproducibility”. In: *Proceedings of the National Academy of Sciences* 115.11 (2018), pp. 2584–2589. DOI: 10.1073/pnas.1708290115.

[59] Ashish Vaswani et al. “Attention is All You Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[60] Sira Vegas, Cecilia Apa, and Natalia Juristo. “Crossover Designs in Software Engineering Experiments: Benefits and Perils”. In: *IEEE Transactions on Software Engineering* 42.2 (2016), pp. 120–135. DOI: 10.1109/TSE.2015.2467378.

[61] Jianwu Wang et al. “Big data provenance: Challenges, state of the art and opportunities”. In: *2015 IEEE International Conference on Big Data (Big Data)*. 2015, pp. 2509–2516. DOI: 10.1109/BigData.2015.7364047.

[62] Joseph Weizenbaum. “ELIZA – a computer program for the study of natural language communication between man and machine”. In: *Communications of the ACM* 9.1 (1966), pp. 36–45. DOI: 10.1145/365153.365168.

[63] Frank Wilcoxon. “Individual Comparisons by Ranking Methods”. In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83. DOI: 10.2307/3001968.

[64] Mark D. Wilkinson et al. “The FAIR Guiding Principles for Scientific Data Management and Stewardship”. In: *Scientific Data* 3.160018 (2016). DOI: 10.1038/sdata.2016.18.

[65] Claes Wohlin et al. *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer, 2012. DOI: 10.1007/978-3-642-29044-2.