

UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO JANDRE DE OLIVEIRA

Enabling Collaboration in Scientific Experiments

NITERÓI

2022

UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO JANDRE DE OLIVEIRA

Enabling Collaboration in Scientific Experiments

Dissertation presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. Area: Computer Science.

Advisor:

VANESSA BRAGANHOLO

Co-advisor:

BRUNA DIIRR

NITERÓI

2022

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

048e Oliveira, Eduardo Jandre de
Enabling Collaboration in Scientific Experiments / Eduardo
Jandre de Oliveira ; Vanessa Braganholo, orientadora ; Bruna
Diirr, coorientadora. Niterói, 2022.
78 f.

Dissertação (mestrado)-Universidade Federal Fluminense,
Niterói, 2022.

DOI: <http://dx.doi.org/10.22409/PGC.2022.m.11898360766>

1. Proveniência. 2. Colaboração. 3. Scripts. 4.
Experimentos 'in silico'. 5. Produção intelectual. I.
Braganholo, Vanessa, orientadora. II. Diirr, Bruna,
coorientadora. III. Universidade Federal Fluminense. Instituto
de Computação. IV. Título.

CDD -

EDUARDO JANDRE DE OLIVEIRA

Enabling Collaboration in Scientific Experiments

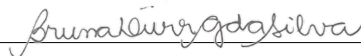
Dissertation presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. Area: Computer Science.

Approved in January, 2022.

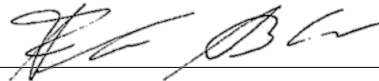
DISSERTATION DEFENSE COMMITTEE



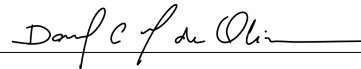
Profa. Vanessa Braganholo, D.Sc. – Advisor, UFF



Profa. Bruna Diirr, D.Sc. – Co-Advisor, UNIRIO



Prof. Fabrício Alves Barbosa da Silva, PhD. – Fiocruz



Prof. Daniel de Oliveira, D.Sc. – UFF

Niterói

2022

Acknowledgements

I want to thank my advisors Bruna Diirr and Vanessa Braganholo, for the opportunity, patience, valuable feedback, and support during this journey.

My parents Sebastião and Neiva, for investing and prioritizing my education, and for being a model of honorable and persistent people.

My wife Greiceane, and my daughter Manuela, for always being on my side, for all the comprehension, and for giving meaning to this effort.

Finally, I thank God for putting those people in my life, and also for giving me the health and conditions to get here.

Resumo

A ciência é uma atividade colaborativa. As descobertas que alcançamos hoje só são possíveis devido ao conhecimento acumulado construído por várias gerações de cientistas. Também é possível observar que a maneira como os cientistas fazem pesquisa mudou com os avanços tecnológicos, tornando-se cada vez mais baseada em computadores (experimentos *in virtuo* e *in silico*). Além disso, a colaboração na pesquisa científica aumentou com o tempo, com equipes (cada vez maiores) ao invés de indivíduos conduzindo as pesquisas científicas. Essas novas maneiras de fazer pesquisa exigem apoio de ferramentas adequadas para ajudar a atribuir o crédito correto aos cientistas que trabalharam no experimento; identificar colaborações implícitas (quando um cientista usa dados produzidos por outro experimento); e coletar e consolidar dados experimentais gerados por diferentes usuários e dispositivos. Sistemas de Gerência de Workflows e ferramentas baseadas em script (ambas com suporte a coleta de proveniência) têm sido maneiras populares de executar experimentos *in silico*. No entanto, essas ferramentas frequentemente negligenciam o aspecto de colaboração. Mesmo soluções que visam experimentos colaborativos nem sempre atendem às necessidades dos cientistas. A literatura mostra pesquisas que discutem assuntos relacionados a experimentos *in silico*. No entanto, elas se concentram na coleta de proveniência em suas aplicações, tratando assim a colaboração apenas como outra aplicação possível, ou se concentram nos Sistemas de Gerência de Workflows, apenas listando a colaboração como um possível desafio. Assim, como primeira etapa desta dissertação, levantamos as ferramentas e abordagens disponíveis que visam auxiliar os cientistas a conduzir experimentos *in silico* colaborativos. Particularmente, nos concentramos nos desafios relacionados à proveniência desses experimentos colaborativos. Elaboramos uma taxonomia com os aspectos da colaboração na pesquisa científica e discutimos cada um desses aspectos. Também identificamos lacunas na literatura que fornecem oportunidades futuras. Em seguida, trabalhamos para preencher uma das lacunas identificadas: a falta de apoio à colaboração em ferramentas baseadas em script com reconhecimento de proveniência. Para isso, propomos uma abordagem colaborativa para este tipo de ferramenta, que foi implementada no noWorkflow. Algumas das capacidades colaborativas que adicionamos ao noWorkflow foram: compartilhamento de ensaios executados entre cientistas, fornecendo assim um único banco de dados de proveniência que representa todo o experimento; criando o conceito de "Experimento" no banco de dados de proveniência, permitindo assim que os cientistas tenham múltiplos experimentos em um único banco de dados de proveniência; e enriquecendo a colaboração no contexto de diferentes experimentos. Apresentamos também os resultados de um estudo de caso que realizamos, e que ajuda a entender como a abordagem proposta aumenta a colaboração e a produtividade, além de enriquecer as informações de proveniência coletadas.

Palavras-chave: proveniência, colaboração, eScience, experimentos in silico.

Abstract

Science is a collaborative activity. The findings we achieve today are only possible due to accumulated knowledge constructed by several generations of scientists. In addition, the way scientists do science has changed with technological advances: it has increasingly become computer-based (*in virtuo* and *in silico* experiments). Moreover, collaboration in science has increased over time, with teams instead of individuals conducting more and more scientific research and teams growing in size. These new ways of doing research demand proper tooling support to help with attributing the correct credit to the scientists who worked on the experiment; identifying implicit collaborations (when a scientist uses data produced by another experiment); and collecting and consolidating experiment data generated by different users and devices. Provenance-aware Workflow Management Systems and Script-based Systems have been popular ways of running *in silico* experiments, but these tools often neglect the collaboration aspect. Even solutions that aim at collaborative experiments do not always address the collaborators' needs. The literature shows surveys discussing subjects related to *in silico* experiments. However, they either focus on provenance collection and applications, thus treating collaboration as just another possible application, or focus on Workflow Management Systems, only listing collaboration as a possible challenge. So as the first step of this dissertation we survey available tools and approaches that aim at aiding scientists to conduct collaborative *in silico* experiments. Particularly, we focus on challenges related to the provenance of these collaborative experiments. We devise a taxonomy with the aspects of collaboration in scientific research and discuss each of these aspects. We also identify literature gaps that provide future opportunities. We then work to fill in one of the identified gaps: lack of support of collaboration on provenance-aware Script-based systems tools. For that, we propose a collaborative approach to this kind of tool and implement it on top of noWorkflow. Some of the collaborative capabilities we add are: sharing executed trials among scientists, thus providing a single provenance database representing the whole experiment; creating the concept of "Experiment" in the provenance storage, thus allowing scientists to have multiple experiments in a single provenance storage; and enriching collaboration in the context of different experiments. We also present the results of a validation case study that we conducted, which helps to understand how the proposed approach increases collaboration and productivity, besides enriching the collected provenance information.

Keywords: provenance, collaboration, eScience, in silico experiments.

List of Figures

2.1	An abstract model of collaborative provenance nodes and dependencies using the extended Open Provenance Model [2]	8
2.2	Collaborative Provenance Model (CPM) [97]	9
2.3	Collaborative workflow composition provenance ontology [99]	10
2.4	ProvDB Conceptual Data Model [52]	11
2.5	Taxonomy of collaboration in scientific research	12
2.6	Experiments life cycle [51]	13
2.7	Timeline of selected publications	23
3.1	noWorkflow architecture [59].	28
3.2	noWorkflow SQLite database model.	29
3.3	View of the provenance information in noWorkflow. (1) Graph with the trial history showing two trials; (2) graph view of trial 1.1.1 – nodes are function activations, and arrows denote sequence of calls (solid arrows) and returns (dashed arrows); (3) Details of trial 1.1.1.	30
3.4	View of trial diff in noWorkflow. (2) graph view of trial 1.1.1 and trial 2.1.1 diff, considering function activations; (3) Details other differences between trial 1.1.1. and trial 2.1.1	31
3.5	Illustrates two scientist (Alice and Bob) trying to collaborate using noWorkflow.	32
4.1	Central node collaboration architecture.	37
4.2	Peer-to-peer collaboration architecture.	38
4.3	Experiment portal screen	43
4.4	Add experiment screen	44

4.5	Group management screen	44
4.6	Manage annotation screen	46
4.7	Add annotation screen	46
4.8	The new "view annotations" option on trial details. Using that option, the user can navigate to manage extended annotations within a trial.	47
4.9	noWorkflow SQLite database model after our implementation. The shadowed entities represent the tables added by our approach.	48
5.1	COVID deaths prediction provenance visualization	53
5.2	SQL query used to find experiments using a specific file	54
5.3	SQL query used to find scientists involved in experiments.	55
5.4	SQL query used to find implicit collaborations	56
A.1	Questions (1) and (2) summary of "Participant Characterization and Previous Experience" Section: (1) What is your education level? (2) How many people scientific experiments have you ever performed on computational environments?	68
A.2	Questions (3) and (6) summary of "Participant Characterization and Previous Experience" Section: (3) How long have you been running scientific experiments on computational environments? (6) How many people usually participate in your group of scientific research?	69
A.3	Question (4) summary of "Participant Characterization and Previous Experience" Section: (4) In which roles have you performed computational experiments? (multiple answers allowed)	69
A.4	Question (5) summary of "Participant Characterization and Previous Experience" Section: (5) What are your preferred/more often used tools to run experiments? (select up to 3 tools)	70
A.5	Collaboration difficulties.	71
A.6	Question (9) and (11) summary of "Collaboration Experience" Section: (9) Do you face any difficulties when performing joint research with other members of your research group? (11) Do you face any difficulties when continuing and/or reproducing the work of other researchers?	71

List of Tables

2.1	Summary of the collaborative provenance models	11
2.2	Aspects of collaboration in the surveyed approaches	15
4.1	Summary of implementation on top of noWorkflow	49
4.2	Summary of the collaborative provenance models including ou Extended noWorkflow version	50
4.3	Aspects of collaboration in the surveyed approaches including Extended noWorkflow	51
A.1	Questionnaire responses. Questions 1-8	72
A.2	Questionnaire responses. Questions 8-15	73

Acronyms

API	:	Application Programming Interface;
DSL	:	Domain Specific Language;
DVCS	:	Distributed Version Control Systems;
JSON	:	JavaScript Object notation;
HTTP	:	Hypertext Transfer protocol;
ORM	:	Object-Relational Mapper;
REST	:	Representational State transfer;
XML	:	Extensible Markup language;

Contents

1	Introduction	1
2	Provenance in Collaborative in Silico Scientific Research	6
2.1	Provenance Models	7
2.2	Collaboration in Scientific Research	12
2.2.1	Experiment Phases	14
2.2.2	Temporality	17
2.2.3	Concurrency Control	19
2.2.4	Sharing	20
2.2.5	Provenance Support	22
2.3	Discussion and Opportunities	22
3	The Script-Based noWorkflow System	27
3.1	noWorkflow Architecture and Features	28
3.1.1	Provenance Storage	28
3.1.2	Provenance Analysis	30
3.2	Lack of Collaboration Support	32
4	Collaborative approach	34
4.1	Proposed approach	35
4.1.1	Central Node Collaboration Architecture	36
4.1.2	Peer-to-peer Collaboration Architecture	37
4.1.3	Multiple Experiment Support	38

4.2	Approach Implementation	39
4.2.1	Synchronizing the Content Database	39
4.2.2	Synchronizing the SQLite Database	40
4.2.3	Support to Multiple Experiments	44
4.2.4	Annotation support	45
4.2.5	Implementation Summary	47
4.3	Approach Comparison	48
5	Case Study	52
6	Conclusion	58
	References	60
	Appendix A - Questionnaire	68
A.1	Participant Characterization and Previous Experience	69
A.2	Collaboration Experience	70
	Appendix B - Example of Bundle Serialized Object	74

Chapter 1

Introduction

Science, as we know, is a result of centuries of hard work. Several civilizations and people contributed to the accumulated knowledge we have today. Although it is common that a single "genius" receives credit for some advances, the truth is that, in general, these "geniuses" used information that was only possible due to the effort of other people to acquire and document it [75, 11]. Thus, science is a collaborative activity of humankind.

Scientific knowledge is built incrementally and cumulatively. To discover something new, scientists have to extensively study their fields to understand the current state of the art. Additionally, an important part of the scientific process is the communication of the work done and the outcomes reached, which allows the scientific community to analyze and review other scientist's research and the obtained results. This process is essential because it allows other people to double-check the ideas, find flaws, or reproduce the achieved results, besides enabling the use of acquired knowledge in future discoveries [13].

"Scientific collaboration can be defined as interaction taking place within a social context among two or more scientists that facilitates the sharing of meaning and completion of tasks with respect to a mutually shared, super-ordinate goal" [76]. Therefore, scientific collaboration occurs not only after the publication but especially in ongoing research. This collaboration can occur in different intensities, reflected in publications with multiple authors or acknowledgements to other researchers. Indeed, collaboration is often encouraged and even required by research funding agencies [76].

Wuchty, Jones, and Uzzi [95] analyze almost 20 million publications from the mid-50s to the early 21st century, and conclude that the production of publications by teams of collaborators has increased over time and that these teams have grown in size. Also, the

authors conclude that publications produced in teams usually receive more citations on average than publications made by a single author, even when self-citations are ignored [95].

At the same time, computer technology has advanced hugely. Computers have become cheaper and more accessible, and computer networks have spread all around the world. This movement produced two direct effects: (i) it allowed collaboration to occur not just between people nearby but also between people located all around the world; and (ii) it increased the number of scientific experiments conducted *in silico*.

In silico experiments are those that are totally executed on the computer. In such experiments, both the execution and observation environments are entirely composed of computational models [84]. They usually require much more support from data management tools and software engineering when compared to traditional experiments [84]. In this dissertation, we are particularly interested in this kind of experiment since they have been increasingly adopted by the scientific community and pose new challenges in terms of collaboration, such as: attributing the correct credit to the scientists who worked on the experiment; identifying implicit collaborations (when a scientist uses data produced by another experiment); and collecting and consolidating experiment data generated by different users and devices [77, 2].

In silico experiments typically demand more support from data management and software engineering tools when compared to other experiment classes (*in vivo*, *in vitro*, and *in virtuo*) [85]. Workflow Management Systems [3, 33, 96, 42] and Script-based systems [21, 48, 59] have been popular ways of running such experiments. Of course, this is a simplification and this classification may present variations between authors and can also be fine-grained depending on the subject of the analysis. Mölder F, Jablonski KP, Letcher B *et al.* [55] classified workflow systems in 5 categories: (1) workflow management systems that offers and demands users to use graphical user interfaces for composition and execution of workflows [33, 42] (this is the kind of tool we are classifying as Workflow Management Systems); (2) systems like Ruffus [38], where workflows are specified using a library of classes and functions for generic programming languages; (3) some systems like Snakemake [55] and Nextflow [27] where workflows are specified using a domain specific language (DSL) [55]; (4) a category which very similar to the third one but instead of specifying the workflow using a DSL it uses configuration file formats like YAML [10]; (5) systems based on generic workflow specification languages like CWL [6] and WDL [89]. These workflow specifications are system-independent and can be executed by multiple

tools, *e.g.* Cromwell [6], Toil [88]. In our classification, category (1) is considered Workflow Management Systems while categories (2), (3), (4), and (5) are considered Script-based systems, plus another category that is not well covered by Mölder F, Jablonski KP, Letcher B *et al* [55], in which we fit systems for generic programming languages that do not need a previous workflow definition, like noWorkflow [59]. Regardless of the classification we use for those systems, collaboration is still one of the challenges in the area [20, 34, 41].

The data related to *in silico* experiments are not limited to the results of the experiment but also include the logical sequence of performed activities; parameters used; intermediary results of activities; information about the execution environment; etc. [32]. It is common for these data to be collected and stored in a provenance database. Provenance is a broad concept that can be applied in many disciplines and is usually linked to the origin of an object or data. It can be seen as a set of metadata that describes not only the object or data itself but also the activities applied in its production process. Bringing the concept into scientific research, it refers to information on how the experiment was performed and how the research results were recorded [41]. This should also include records of how the collaboration was conducted.

Provenance gathering is a common feature in many tools [3, 21, 33, 41, 48, 59, 96, 4]. However, when focusing on collaborative experiments, two challenges emerge: (C1) how to collect provenance in a collaborative experiment (this comprises collecting provenance of actions of scientists that may be working in different parts of the experiment or different geographical locations and machines); and (C2) how provenance can be used to make collaboration easier in this environment.

As a first step of this dissertation, we map the state-of-the-art approaches and provenance-aware models that are available to conduct *in silico* collaborative experiments. We aim at investigating how they address challenges C1 and C2. To do so, we answer the following research questions: (R1) How do existing tools store and collect provenance in a collaborative experiment?; (R2) How do existing tools use provenance to make collaboration easier in scientific experiments?. The research questions R1 and R2 are respectively linked to challenges C1 and C2.

As a result of this survey we propose a taxonomy and use it to classify the existing tools and discuss opportunities based on the gaps we identified. These opportunities and gaps are used as a starting point to our collaborative approach to script-based experiments.

What we found in our survey is that although some authors (*e.g.* [99, 33, 29]) deal with provenance aware tools to the design and analysis of experiments collaboratively,

they are based on workflow management systems, ignoring the fact that many scientists use scripts in their experiments [66]. When looking at provenance aware script-based solutions, collaboration support is still quite limited. Another option could be the use of regular software development tools, such as version control systems, to collaborate and execute the experiment, but these tools typically do not address specific problems of scientific research such as provenance gathering. This survey is detailed on Chapter 2. An earlier version of it was also published in SIGMOD Record [44].

Although this survey showed that there is a gap in script-based solutions, we wanted to quantify the relevance of this gap to the scientific community, so we made a questionnaire with researchers. This questionnaire showed that the vast majority of respondents work in teams and prefer to use script-based solutions rather than workflows, even though the limitations we already found. The questionnaire also showed that 52% of the respondents face problems in collaboration with other scientists and 68% face problems to reproduce collaborators' work. Although the limited number of respondents, these data helped to show the importance of a provenance-aware script-based approach for collaborative *in silico* experiments. We detail this questionnaire and other insights we found in Appendix A.

To fill the gaps we identified in our survey, this work proposes a provenance aware approach that assists scientists in conducting *in silico* collaborative experiments. For this, we take advantage of existing research on conducting experiments that are materialized in the form of scripts. Among the existing approaches, we chose noWorkflow [59, 68], a tool that allows capturing the provenance of Python scripts. The choice was made because noWorkflow allows capturing provenance transparently, in different levels of granularity, and is Open Source¹. We extend noWorkflow by adding collaborative capabilities such as: sharing executed trials between scientists, thus providing a single provenance database representing the whole experiment; and creating the concept of "Experiment" in the provenance storage, thus allowing scientists to have multiple experiments in a single provenance storage. We provide further details of noWorkflow in Chapter 3.

To evaluate our approach, we conducted a case study. We built a simulated scenario for using our approach in a fictitious research institute that performs multiple collaborative research about COVID-19. The experiment aims at evaluating how our proposed approach could increase the productivity and fluidity of the conducted research. We contrast our approach in this scenario with the alternative of using the original version of noWorkflow

¹<https://github.com/gems-uff/noworkflow>

as a baseline. The results showed that our approach allowed scientists to answer questions about the experiment that were not possible when using noWorkflow's original version and also provided a considerable gain of productivity in collaborative scenarios.

The dissertation proceeds as follows: Chapter 2 presents our survey, including: an analysis of the existing provenance models that aim to precisely represent collaborative research; a proposed taxonomy to capture the aspects that may influence collaboration in the scientific research scenario; a discussion of publications and opportunities in the field. Chapter 3 presents a summary of noWorkflow and its gaps regarding collaboration. Chapter 4 presents our approach to enable collaboration in scientific experiments. It also brings the implementation details of that approach on noWorkflow. Chapter 5 details the case study that we conducted and results obtained. Finally, Chapter 6 presents some final considerations and establishes some possibilities of future work.

Chapter 2

Provenance in Collaborative *in Silico* Scientific Research

To answer the research questions (R1) How do existing tools store and collect provenance in a collaborative experiment?; (R2) How do existing tools use provenance to make collaboration easier in scientific experiments?; we made a snowballing [37] based survey. We evaluate 170 publications and select 20 approaches and 7 surveys. To be selected, an approach has to satisfy the following criteria: (i) has collaboration as a focus (*i.e.*, the problem to be solved or the subject of a survey); or (ii) has provenance as a focus while discussing collaboration features; and (iii) is in the context of *in silico* scientific experiments. The surveys are used to reinforce this work's motivation and as a benchmark. From the 20 selected approaches, 15 are tools for collaborative experiments, 2 are provenance-aware data models for collaborative experiments, and 3 approaches present both a tool and a provenance-aware data model for collaborative experiments.

Although there are some existing surveys on related topics [9, 20, 34, 41, 49, 71, 95] they do not cover all the aspects that we are interested in. Our survey differs from Lu and Zhang's work [49] and Belloum *et al.* [9] by bringing a more detailed and up-to-date view of the work in the area. Besides that, Belloum *et al.* discuss the challenges to support e-science collaborative experiments with a closer look at the experiment life cycle, but it only addresses the tools provided by the VL-e project. Wuchty *et al.* [95] aim to demonstrate that teams have been increasingly dominating the scientific research in the production of knowledge, without addressing available tools and research that helps the execution of this type of experiment. On the other hand, Davidson and Freire [20] and Gil *et al.* [34] focus on the challenges and opportunities existing in the Workflow Management Systems research, without detailing the available tools. Other publications focus on provenance collection and its applications, and collaboration merely appears as

one of the possible applications of provenance [41, 71]. As opposed to that, our survey focuses on provenance-related aspects of collaboration.

This chapter is organized as follows. On Section 2.1 we analyze the existing provenance models that aim to precisely represent collaborative research. Section 2.2 discusses some aspects of collaborative research and proposes a taxonomy to capture the aspects that may influence collaboration in the scientific research scenario. Section 2.3 discusses publications and opportunities in the field. We also classify the tools for collaborative experiments covered by the survey in this taxonomy.

2.1 Provenance Models

Provenance is a broad concept and can be seen from different perspectives. Ragan *et al.* [71] classify provenance in five types: *Data provenance* (the history of changes and movement of data); *Visualization provenance* (the history of graphic views and visualization states); *Interaction provenance* (the history of user interaction with a system); *Insight provenance* (the history of cognitive outcomes and information derived from the analysis process); and *Rationale provenance* (the history of reasoning and intentions behind decisions, hypotheses, and interactions) [71].

Collaboration brings additional challenges in collecting and storing provenance. The first challenge (C1) resides in how to collect provenance in a collaborative experiment. It involves collecting data, interaction, and visualization provenance from multiple devices since scientists collaborating could be working on different machines. Some initiatives capture provenance from multiple devices [22, 25, 92], but they usually focus on high-performance settings, where a single user executes an experiment on a distributed manner, with parts of the trial being executed in a cloud, cluster, or a grid. This is different from having several scientists executing different trials on different devices. Collecting this provenance could be useful in several situations, such as giving credit to those involved in the research [41], auditing the research, enabling the reproducibility of the experiment and providing relevant information that allows each member of a group to better understand the actions of other members in the context of a collaborative scientific experiment. Another challenge (C2) resides in how to use this provenance to make collaboration easier in a collaborative environment.

The first step to overcoming these challenges is providing a provenance model that can properly represent the research collaboration aspects. This model needs to represent

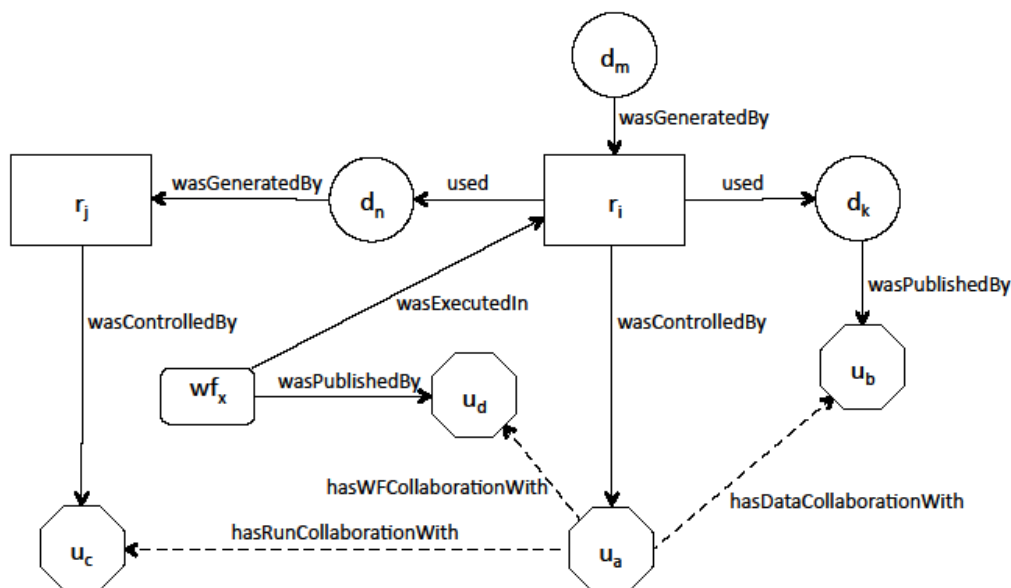


Figure 2.1: An abstract model of collaborative provenance nodes and dependencies using the extended Open Provenance Model [2]

four main aspects [49]: (i) Distribution (D) – Collaboration typically involves resources from multiple organizations; (ii) Heterogeneity (H) – Provenance produced by different workflows may have different formats. Even those that conform to the same schema may evolve during the experiment life cycle; (iii) Multilevel (M) – Experiments usually have complex tasks that are modeled hierarchically (*e.g.*, using sub-workflows, or by functions calling functions in a script). Although this is not a specificity of collaborative experiments, the provenance model should store this hierarchy; (iv) Collaboration (C) – The model must support new user iterations and collaboration standards, besides storing information about these collaborations. The term collaborative workflow has been used with multiple meanings in the literature. It is understood both as the *collaboration between workflows* or the *collaboration between workflow users* [49]. Collaboration between workflow users is the direct collaboration of users in the context of a scientific workflow. On the other hand, a collaboration between workflows refers to the indirect use of data produced by another workflow. This suggests an implicit collaboration, when collaboration occurs through the data published by another researcher.

Altintas *et al.* [1, 2] propose the provenance model shown in Figure 2.1, which is capable of capturing *implicit collaborations* within a scientific experiment. The model identifies workflows dependency from the relations between the dataflows input and output, and also helps to identify contributions from users who collaborate on a project based on records of past executions. The authors extend OPM (Open Provenance Model)

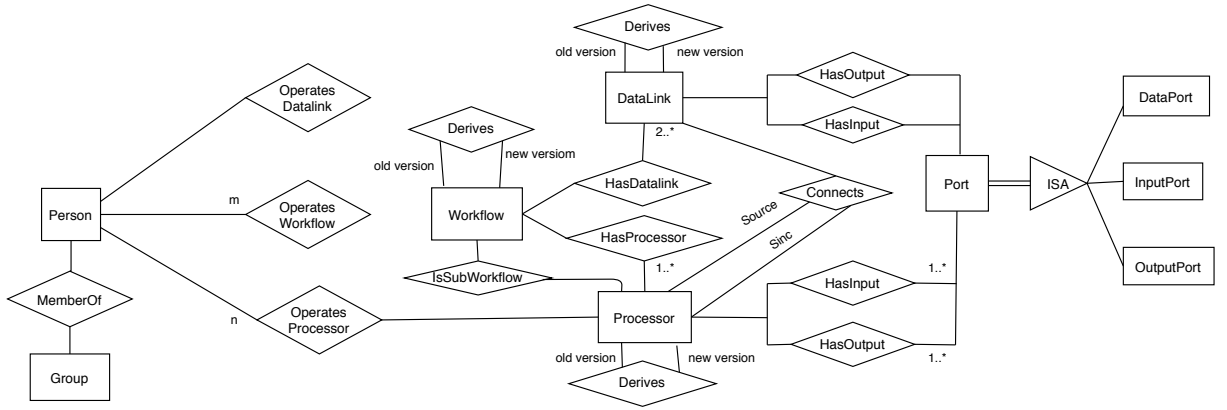


Figure 2.2: Collaborative Provenance Model (CPM) [97]

[56] to record user interactions when publishing data and workflows, which is essential for identifying the various types of user collaboration. This model explicitly represents *collaboration amongst users* (agents u_i in the figure) and which users were responsible for each run of the experiment (r_i in the figure). According to Ragan *et al.*'s classification [71], it captures *data* and *interaction* provenance. The approach also proposes a query language, which is an extension of the QLP (Query Language for Provenance) [7].

Missier *et al.* [54] propose a model that facilitates the sharing of provenance in collaborative environments. The model aims to provide end-to-end support for *implicit collaborations*. The approach treats sharing as an action from which provenance has to be preserved, *i.e.*, the focus is to register the provenance of the data sharing process. To do so, the model adds new information to provenance traces, *stitching* common parts of those traces. With this, the model can represent cases when scientists use data that was produced by another scientist's workflow, even when they come from heterogeneous workflow systems. This model can represent *data* and *interaction* provenance [71].

Zhang *et al.* [97], Confucius [99], and ProvDB [52] present provenance models and tools that track collaboration provenance. Zhang *et al.* [97] propose the Collaborative Provenance Model (CPM), which is an extension of PROV-DM (PROV Data Model) [57]. Figure 2.2 shows that the model explicitly represents *Person* and *Group* of *Person* (a collaborating group), besides versions of *Workflow*, *Processor*, and *Data Links*. It also captures which user operates which workflow version, process version, and data link version. The model captures *data* and *interaction* provenance [71].

Confucius [99] introduces a provenance ontology (Figure 2.3). The ontology aims at supporting the capture and record of scientific workflow composition and user interactions during the process of a collaborative workflow composition. The provenance is stored in

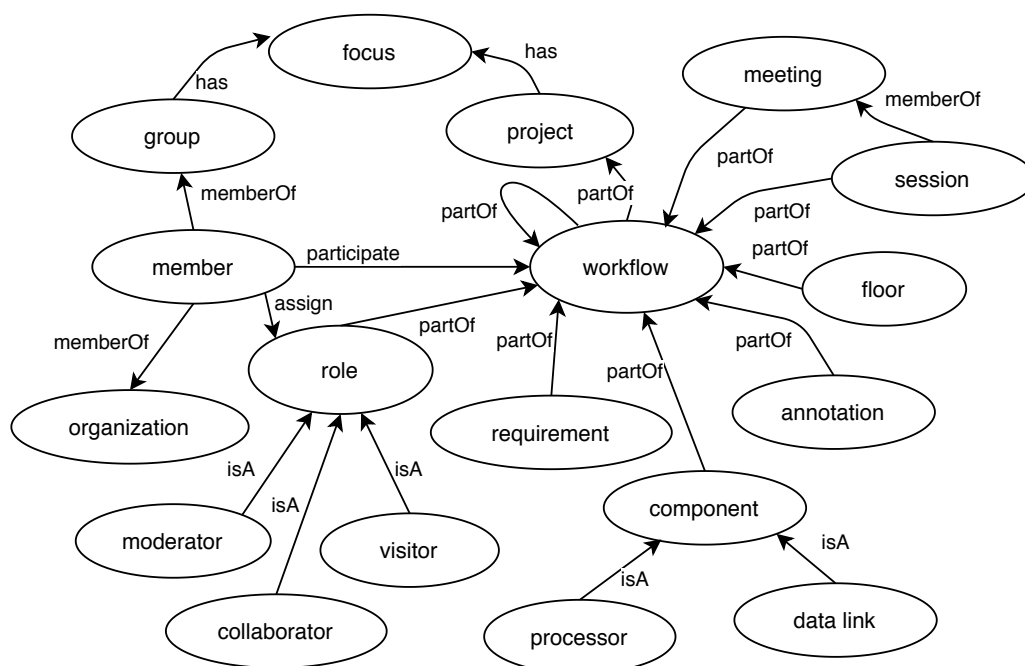


Figure 2.3: Collaborative workflow composition provenance ontology [99]

a provenance repository on the central node of Confucius. Note that the ontology can represent workflows and their components, and roles of people in the collaboration groups. As for Ragan *et al.*'s classification [71], this model can represent *data* and *interaction* provenance, besides the remaining types through *annotations*.

ProvDB [52] proposes a provenance model with a schema-later approach, providing a base schema that can be extended by arbitrary properties as key-value pairs (Figure 2.4). These values can be complex, such as a JSON document. Information to fill in the base schema is collected through Git and the built-in ingestors. Additional information can be added through custom ingestors or by user's annotations. When the user runs a command using ProvDB, the system verifies the registered ingestors and executes them. The ingestors can analyze the before- and after-state of the artifacts produced by the command to generate provenance information about the executed command. The model deals with *data* and *interaction* provenance [71] and can deal with all other types of provenance using the ingestors.

Table 2.1 summarizes how each model supports the collaboration aspects mentioned at the beginning of this section. All the models present limitations when representing some aspects of collaboration. Altintas *et al.* [1, 2] present a model that is capable of capturing user collaborations but lacks support for the other analyzed items. Confucius [99] and CPM [97] do not adequately treat the heterogeneity of collaboration, since they are not able to deal with different workflow formats. Confucius also does not deal with workflow

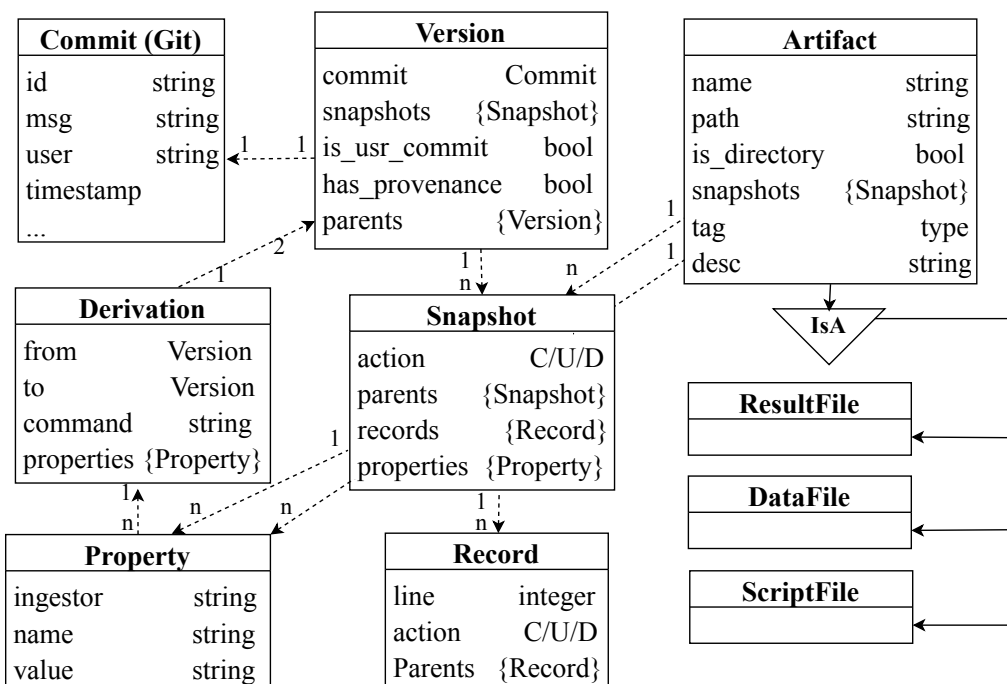


Figure 2.4: ProvDB Conceptual Data Model [52]

Table 2.1: Summary of the collaborative provenance models

Provenance Model	Provenance Types [71]	Aspects of Collaborative Research*			
		D	H	M	C
Altintas <i>et al.</i> [1, 2]	Data; Interaction	No	No	No	Yes
CPM [97]	Data; Interaction	Yes	Evolution Only	Yes	Yes
Missier <i>et al.</i> [54]	Data; Interaction	Yes	Different schema only	No	Yes
Confucius [50, 86, 96, 99]	All**	Yes	No	Yes	Yes
ProvDB [52]	All**	Yes**	Yes**	Yes**	Yes

* (D) Distribution (H) Heterogeneity (M) Multilevel (C) Collaboration

**Modeled as extended properties

evolution. Missier *et al.*'s model [54] present limitations in dealing with workflow evolution and representing the multilevel hierarchy. ProvDB [52] is the only model that provides support for all the analyzed aspects, but it does that making use of extended properties in a key-value schema. Regarding Ragan *et al.*'s [71] classification, only Confucius and ProvDB can capture all types of provenance, but they do that by using annotations or extended properties. This kind of schema could make things hard and inefficient to query. Another important aspect is that the models just provide a form of storing the information generated in collaborative research and do not necessarily provide a way of collecting them. We also notice that the models supported by a tool [52, 97, 99] can store some provenance on collaboration, but the tool may not fully capture it.

In this section, we show several provenance models that are able to store in part (or

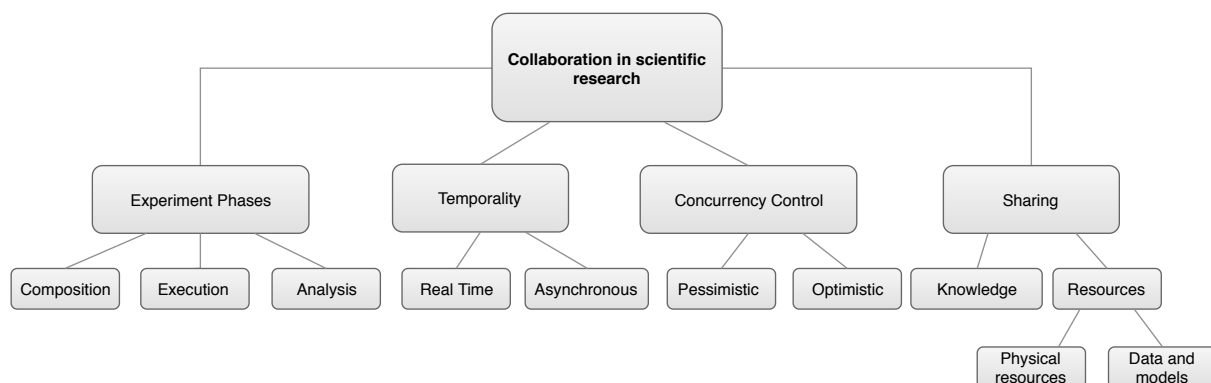


Figure 2.5: Taxonomy of collaboration in scientific research

in total) collaboration aspects of scientific experiments. However, in order to properly answer our two research questions, we need more insights. In the next section, we discuss how the existing approaches capture and use this information to foster collaboration.

2.2 Collaboration in Scientific Research

Scientific research is a complex activity per se, and collaboration in this environment becomes a challenging task. To better understand these challenges, we independently analyze the aspects that may influence collaboration in the scientific research scenario. We develop a taxonomy (Figure 2.5) by examining the 20 approaches we selected, capturing, and categorizing their similarities and differences. We then standardize and enrich the categorization based on other publications [52, 70, 73].

The first branch of the taxonomy is *Experiment Phases*, which is defined in different ways by different authors [9, 52]. In this survey, we use the classification proposed by Mattoso *et al.* [51] and illustrated on Figure 2.6, where scientific experiments go through three phases: *composition*, *execution*, and *analysis*. During *composition*, scientists structure and configure the entire experiment, establishing the logical sequence of activities, the type of input data to be provided, and the type of output data. During *execution*, scientists materialize the experiment, define the required input data to run the experiment, trigger its execution, and get the results to be analyzed. During *analysis*, scientists study the gathered data from prior phases [51] aiming at proving or refuting their hypothesis. Each of these experiment phases may involve different forms of collaboration, as discussed in Section 2.2.1. Provenance plays an important role in each phase, so it is important to keep track of all the user interaction and data transformations on a provenance database.

The second branch of the taxonomy regards the *temporal* aspect of collaboration.

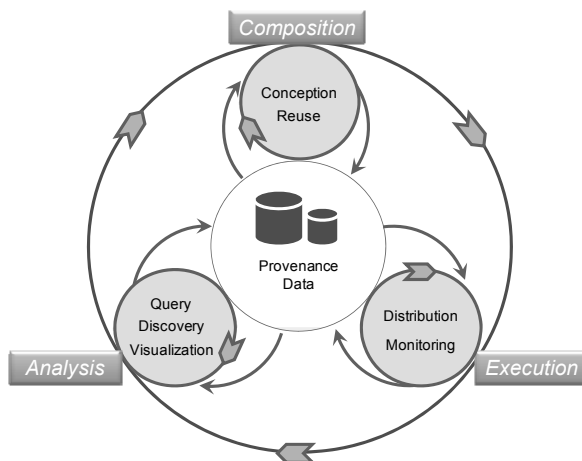


Figure 2.6: Experiments life cycle [51]

This aspect is related to the experience of time and the temporal organization of activities [73]. In a collaborative environment, some tasks need to be synchronized, while others can be done asynchronously. Section 2.2.2 analyzes if and how existing approaches allow collaborative tasks to occur in real-time or asynchronously.

The third branch is *concurrency control*, which has been extensively studied in the context of databases [72, 19, 30], operating systems [82], and software development [79, 78, 61]. Although the conduction of scientific experiments has its peculiarities, the taxonomy uses ideas that govern version control systems once the problems that may arise when accessing a resource during an experiment resembles the ones that are dealt with by such systems. There are two main concurrency control policies to allow simultaneous work on version control systems: optimistic and pessimistic policy [70]. In pessimistic policies, the artifact that needs to be accessed by several users is restricted to be changed by a single user at a time (*i.e.*, the artifact is locked to a specific user and is only released when the interaction is finished). In optimistic policies, artifacts can be updated in parallel, and users need to merge the changes when conflicts occur. Each of these policies has advantages and disadvantages, and the choice of the most appropriate policy depends on the concurrency frequency, as well as the effort required to merge the artifacts [70]. Section 2.2.3 discusses how existing approaches deal with concurrency control.

The fourth branch of the taxonomy regards the *sharing* of conceived ideas as well as results and experiments. This allows other researchers to develop new research using these ideas [13]. Although this process is practically mandatory in research, there is a considerable variation in what is shared, which may facilitate or hinder the research reuse. There is indeed some initiatives like the "FAIR Guiding Principles for scientific data management

and stewardship" [93] that provides guidelines to increase the Findability, Accessibility, Interoperability, and Reuse of datasets. FAIR establishes that other important aspect is that not only humans should have the capacity to interact with the shared data, but also the computational systems should have the ability to find, access, interoperate, and reuse data. Some forms of sharing within research would be knowledge sharing, as in publications; data and models sharing, such as sharing a database obtained after some research; and physical resources sharing, such as what happens in the case of institutions sharing a supercomputer. For these different types of sharing (in particular, knowledge, data, or models) to succeed, provenance data is crucial. Without it, the shared information comes out of context and may be useless. Section 2.2.4 evaluates which of these sharing forms the existing approaches are prepared to deal with, and how this occurs.

Note that all branches of this taxonomy are connected to challenges C1 and C2. They need to be taken into consideration both when collecting provenance (C1) and using this provenance to make collaboration easier (C2). Note also that all branches of the taxonomy are related to data and interaction provenance [71].

Table 2.2 presents the selected approaches and classifies them according to our taxonomy. This classification considers the aspects addressed in each approach and not the solution maturity of a specific aspect. Thus, two solutions can be equivalently classified, but this does not mean they have the same robustness level. We also evaluated if these tools collect provenance and, whenever possible, classify which type of provenance these tools support. On the next subsections, we detail each of the taxonomy branches and how the surveyed approaches fit them, besides briefly discussing the provenance support of those tools.

2.2.1 Experiment Phases

Most of the approaches tackle collaboration in the composition phase, while the execution and analysis phases have been receiving less attention.

Composition. This phase has two sub-phases: *conception* and *reuse* [51]. *Conception* aims at producing a high-level representation of the scientific experiment protocol, which is afterward refined and instantiated as a concrete implementation [51] in the form of a workflow or script. *Reuse* consists of retrieving an existing component and adapting it to a new purpose [51].

Some proposals support the *conception* sub-phase [33, 29, 97, 99, 42, 58, 52, 17].

Table 2.2: Aspects of collaboration in the surveyed approaches

Approach	Aspects of collaboration				
	Experiment Phase	Temporality	Concurrency Control	Sharing	Provenance Support
Confucius [50, 86, 96, 99]	Composition	Asynchronous; Real Time	Pessimistic	Data and models	Data; Interaction
myExperiment [35, 36, 23]	Composition and Analysis	Asynchronous	N/A	Data and models; Knowledge	Yes**
CAMERA [5, 81]	Composition and Analysis	Asynchronous	N/A	Data and models; Knowledge	Yes**
e-ScienceNet [16, 14, 15]	Composition	Asynchronous	N/A	Data and models; Knowledge	No
Collaborative PL-Science [62]	Composition and Analysis	Asynchronous	N/A	Data and models; Knowledge	No
Ellkvist <i>et al.</i> [29]	Composition	Real Time	Optimistic	Data and models	Data
VisTrails [33]	Composition	Asynchronous	Optimistic	N/A	Data
NoCoV [90]	Analysis	Asynchronous; Real Time	N/A	N/A	No
RASA [53]	Execution	Asynchronous	N/A	Physical resources	No
Wood, Wright, and Brodlie [94]	Analysis	Real Time	N/A	N/A	No
ViroLab [12]	Composition	Asynchronous	N/A	Data and models	Yes*
J. Zhang <i>et al.</i> [97]	Composition	Real Time	Pessimistic	Data and models	Data; Interaction
Mostaen <i>et al.</i> [58]	Composition	N/A	Pessimistic	N/A	No
ProvDB [52]	Composition	Asynchronous	Optimistic	Data and models	Data; Interaction
Dataverse [46]	Composition and Analysis	Asynchronous	N/A	Data and models	Yes**
OpenML [87]	Composition and Analysis	Asynchronous	N/A	Data and models	No
CoCalc [17]	Composition and Analysis	Asynchronous; Real Time	Optimistic	Data and models; Knowledge	Data; Interaction
Sumatra [21]	Analysis	Real Time	N/A	Data and models	Data

*No details are provided to correctly classify which provenance types are collected

**Stores data collected by other tools

VisTrails [33] is a provenance-aware Workflow Management System that implements support for the collaborative composition of the workflow. Ellkvist *et al.* [29] and Zhang *et al.* [97] introduce VisTrails extensions that unleash real-time collaboration on the composition phase of the experiment. Confucius [99] extends Taverna [42] to allow the collaborative composition of workflows by using a client-server architecture that communicates using a service-oriented architecture and XML messages. Mostaeen *et al.* [58] propose a fine-grained lock scheme that aims to increase efficiency in workflow conception by reducing the waiting time for lock release. ProvdB [52] uses Git to allow the user to collaborate on experiment conception. It also enriches the information collected using ingestors. CoCalc is a virtual workspace for calculations, research, collaboration, and for authoring documents [17], which provides a web portal where scientists can share files with multiple collaborators. This includes Jupyter notebooks, where multiple scientists can simultaneously edit scripts in real-time.

Regarding the *reuse* sub-phase, many of the selected publications focus on the sharing aspect, thus allowing scientists to share a component, a workflow, or a dataset with their peers. That is the case of CAMERA [5], e-ScienceNet [16], myExperiment [35], OpenML [87], Dataverse [46], Collaborative PL-Science [62] and ViroLab [12]. ViroLab [12] provides a way for sharing script components of a workflow. The remaining approaches focus on experiments represented as workflows.

Execution. RASA [53] is the only solution that addresses collaboration in the execution phase of the experiment. RASA is a framework that coordinates the use of scientific instruments, being able to dynamically adapt workflows during the experiment execution according to the needs of the scientists and the equipment.

Analysis. The analysis phase has three sub-phases: *query*, *visualization*, and *discovery* [51]. During *Query*, scientists can relate data and extract information of both the experiment results and provenance data. *Visualization* simplifies the analysis of large volumes of raw data. Data is often projected in graphs or maps to simplify the identification of patterns and the reasoning over the data. During *discovery*, scientists evaluate query results and visual data to draw conclusions about the entire experiment, aiming at checking if the hypothesis is likely to be correct or if it should be refuted. For this, scientists must analyze the experiment as a whole, including all the executions of the experiment (trials) [59].

OpenML [87], CAMERA [5] and myExperiment [35] provide *query* support. They offer a mechanism for sharing not just the workflow components but also other data, such

as results and provenance datasets. The myExperiment platform also allows scientists to interact with each other and discuss the shared results. These approaches support the *discovery* sub-phase since they provide a mechanism to analyze and discuss the experiment as a whole. Although not described in the paper [21], Sumatra provides some support to collaboration [80]. It allows different users to share the same provenance database and provides some query features to support the *query* sub-phase.

NoCoV [90] and Wood, Wright, and Brodlie [94] support the *visualization* sub-phase. NoCoV (Notification-service-based Collaborative Visualization) uses a client-server architecture to provide mechanisms for the collaborative visualization of experiment data. The pipeline controller (server) is responsible for synchronizing the clients' visualization, and multiple clients can connect to it simultaneously. The clients could be a pipeline editor (which can update the visualization pipeline) or a parameter control client (which can only adjust visualization parameters). Wood, Wright, and Brodlie [94] propose a collaborative approach on top of IRIS Explorer [31] that allows multiple scientists to collaboratively interact over a visualization.

CoCalc [17] supports the *query*, *discovery*, and *visualization* sub-phases. It allows scientists to query the results of the experiment and its history, besides other data. Scientists can also visualize the results using Jupyter notebooks and libraries, such as matplotlib. They can also use chat rooms to discuss the experiment and reason about it.

Dataverse [46] focuses on creating an infrastructure to share datasets related to scientific publications. It provides the data to be used in the *query*, *discovery*, and *visualization* sub-phases, although it does not explicitly deal with them.

2.2.2 Temporality

Starting with the approaches that implement *asynchronous* interactions, CAMERA [5], myExperiment [35], e-ScienceNet [14], Collaborative PL-Science [62], ViroLab [12], Dataverse [46] and OpenML [87] provide solutions focused on the sharing of data and components, where a scientist can publish workflows, components or datasets. These published artifacts become available for other scientists to reuse them *asynchronously*. On VisTrails [33], each version of the workflow is treated as a node in a version tree. Nodes are never modified or deleted (each modification generates a new node in the tree). To collaboratively compose a workflow, scientists can asynchronously work in their local copy of the workflow and synchronize it with another scientist's copy when needed. However, if two scientists modify the same workflow before synchronizing it, this generates multiple

disjoint versions, which can be problematic since the changes could be complementary. When this occurs, the scientist should re-implement part of the workflow. ProvDB [52] is a client-server application that uses Git to support version management tasks as well as distributed and decentralized management of individual repositories. Each user makes the necessary modifications to her local repository and, asynchronously, synchronizes them using Git.

We have also identified several proposals that provide *real-time* collaboration. Ellkvist *et al.* [29] implement a solution based on a client/server architecture, where the server is a MySQL database, and the client is a modified version of VisTrails that consists of a mechanism to unleash real-time collaboration during workflow composition. The server is used as a shared database to synchronize the versions among the scientists. When one scientist makes a modification, it is saved on the shared database and the other clients are automatically notified to update their local versions. Although implemented in VisTrails, the authors argue that their solution could be implemented in other provenance-aware Workflow Management Systems. Zhang *et al.* [97] also implement a plugin to VisTrails, which allows any changes made by one scientist to be immediately reflected on all other collaborators' screens. The approach communicates with VisTrails through third-party packages and the VisTrails API. It utilizes Git to provide a new version tree over the existing VisTrails History View. Wood, Wright, and Brodlie [94] present a real-time approach based on a client-server architecture, which allows scientists to visualize an experiment collaboratively. Users can share and alter visualization parameters and visualization pipelines so they can see other users' changes in real-time. Participants may also disconnect single modules from their group to allow periods of independent work on a subset of the pipeline while remaining in contact with the rest of the session. Sumatra [21] provides a way of sharing the provenance database in real-time. The information is shared as soon as it is collected. However, the solution still has several limitations and, in some scenarios, even data loss is possible.

Three solutions work in both real-time and asynchronous scenarios. Confucius [99] provides a solution inspired by a protocol of human communication called Robert's Rules of Order, which is a set of rules created by Henry M. Robert in 1876 to run effective and orderly meetings with maximum fairness to all members [43]. Confucius implements that with a locking strategy that controls which scientist has the right to interact at a given time in a real-time collaboration session. Confucius also maintains a database on the central node that is used for storing provenance of collaboration and workflow evolution, which allows asynchronous collaboration. NoCoV [90] is implemented in a service-oriented

architecture that uses notification Web services to synchronize clients and server. When someone alters the visualization pipeline, the pipeline controller notifies other clients, so everyone sees the same visualization in real-time. To transmit information between the pipeline controller and the client, it uses skML [28], an XML-based dataflow description language. NoCoV uses the stateful Web Services provided by GlobusToolkit 4 (GT4) [83]. Using this stateful feature, the state of the pipeline is persisted and users can retrieve the saved pipeline to continue the work of other users, thus achieving asynchronous collaboration. CoCalc [17] provides a solution based on a web portal where scientists can simultaneously compose scripts in real-time. All changes are immediately synchronized with others. It saves files and data in its cloud infrastructure, so scientists can leave the session and rejoin when needed (allowing asynchronous work).

2.2.3 Concurrency Control

All approaches providing a mechanism for concurrency control focus on the composition phase of the scientific experiment.

Starting with the approaches that implement the *pessimistic* policy for concurrency control, Confucius' authors [99] treat the concurrency control problem as they would treat it in a face-to-face activity. A central node is needed for the collaboration to occur. A group is registered on this node, and the person responsible for registering the group is automatically assigned as the group moderator. The moderator is responsible for shift control, which is the definition of which group member is allowed to change the workflow at a given time. There is an algorithm for automatically granting and releasing the right to the shift, but the moderator can intervene by taking the right to the shift. Confucius also considers that workflow development can last for long periods in an asynchronous form and, in this scenario, workflow level locking may not be appropriate. Therefore, Confucius blocks smaller building blocks. Thus, several scientists can change the same workflow at the same time. Confucius establishes that the smallest building blocks are tasks and data channels, that in Taverna are called processors and data links, respectively. Confucius introduces the concept of synchronization area "that represents a conceptual area in a shared scientific workflow, which allows only one collaborator to work on it at a given time" [99]. When the user starts to modify a data link, the synchronization area is the data link itself. When the user locks a processor, the synchronization area is the processor and all the fan-out data links of the processor. Zhang *et al.* [97] also implement a pessimistic collaboration protocol based on Robert's Rules of Order. The protocol is

fully described in [45, 98]. Mostaeen *et al.* [58] analyze the existing locking schemes in terms of concurrency control on the composition of workflows. The approach presents a pessimistic strategy of fine-grain locking in scientific workflows. The lock is done for a single user but at the attribute level, while other approaches use turns or module level locking. The main benefit here is to reduce the waiting time for a lock since smaller portions of the workflow are locked for each modification.

Only four approaches implement the *optimistic* policy for concurrency control. Ellkvist *et al.* [29] and VisTrails [33] present an optimistic lock approach that creates different branches in the version tree in the case of simultaneous changes. Although VisTrails presents a mechanism for merging, it merges two version trees of different files and not two branches of the same version tree. If the scientists want to keep both of the changes, they will have to use the diff functionality to better understand what has changed and to replay the changes manually. VisTrails also has a functionality called "analogy" that could help on the process: given two versions of a workflow, VisTrails can automatically detect their differences and apply those differences to another workflow version. Ellkvist *et al.*'s proposal [29] is built on top of VisTrails, and although it adds support to real-time collaboration, it uses the same concurrency control approach of VisTrails. ProvDB [52] also works on the idea of immutable versions, in which any update will result in a new version. In Cocalc [17], the whole experiment environment is cloud-based. All changes are made directly in the cloud and synchronized with the online scientists' browser – there is no lock.

2.2.4 Sharing

Most of the approaches providing sharing features allow the sharing of *data and models*. That is the case of e-ScienceNet [16], ViroLab [12], myExperiment [36], CAMERA [5], Dataverse [46], OpenML [87], ProvDB [52], Zhang *et al.* [97], Ellkvist *et al.* [29], Confucius [99], Collaborative PL-Science [62], CoCalc [17] and Sumatra [21]. ProvDB [52], Zhang *et al.* [97], Ellkvist *et al.* [29], and Confucius [99] work with a centralized database for the experiment, which stores the provenance collected from the collaborative experiment and makes this information available to the involved scientists. ViroLab [12] addresses the issue of sharing code blocks for reuse. The approach also mentions the persistence and sharing of provenance but does not provide details on what kind of provenance information is stored and shared. Sumatra [21] provides a way of sharing a provenance database between multiple scientists.

Roure, Globe, and Stevens [24] argue that one of the barriers of workflow reuse is on how the knowledge about the workflow could be transmitted to potential users. That challenge can be minimized by the distribution of other documentation data in addition to the workflow definition. Most of the approaches try to increase collaboration by adding the possibility of sharing *knowledge*. That is the case of e-ScienceNet [16], myExperiment [36], CAMERA [5], Dataverse [46], CoCalc [17], and Collaborative PL-Science [62]. Pereira *et al.* [62] propose the Collaborative PL-Science, an extension of PL-Science [18]. It aims to facilitate the reuse of components in the construction of scientific workflows, thus combining models and knowledge sharing. The idea is that adding information that helps to understand published artifacts facilitates reuse. The approach uses ontologies to enrich the information of shared objects. CoCalc [17] allows the sharing of a great variety of files, including scripts in multiple programming languages. It also allows the sharing of documentation that can help scientists to better understand what has been made on the experiment and help them to better use the shared data and scripts. e-ScienceNet [16] is another approach that allows both the sharing of data and models and also knowledge. It differs from other approaches because it presents a peer-to-peer solution for sharing the experiment results and models without the dependency of a central server.

Some publications explore the creation of portals for sharing data and reusable components in research, where it is common to share scientific workflows. Goble and Roure [36] propose myExperiment, a social network for scientists focused on workflow-related issues. It allows the sharing of the workflow itself as well as other metadata, such as provenance logs, besides enabling researchers to interact using the tool, commenting, and discussing the shared resources. CAMERA [5] also focuses on the sharing of scientific workflows and provenance logs. The tool works exclusively with Kepler [4] workflows and allows the execution of the experiments within the portal. OpenML [87] is focused on the machine learning community and provides a portal to share datasets, algorithm implementations, and workflows. It also presents a Web API, which allows users to interact with the portal in a programmatic form, and ways of sharing scientific tasks and receiving other scientists' collaboration. Dataverse [46] provides a Web infrastructure to share datasets related to scientific publications. The main idea is that sharing the datasets may increase the reproducibility of experiments, and, as a counterpart to the authors, it may increase the number of citations of the related publications [46].

RASA [53] is the only approach that focuses on sharing physical resources. The approach provides a framework for coordinating the use of scientific instruments. The idea is to provide a mechanism to dynamically modify workflows depending on the needs

of the requester scientist and the particularities of the equipment, and also the knowledge of the equipment operator.

2.2.5 Provenance Support

As seen in Table 2.2, many of the tools do not collect provenance. Although ViroLab [12] provides some provenance support, it does not provide details on what is stored. DataVerse [46], CAMERA [5], and myExperiment [24] provide support for storing and sharing provenance data collected by other tools. CoCalc [17] collects interaction provenance through the log of the activities executed by scientists, but this unstructured information is hard to query. VisTrails [33], Ellkvist *et al.* [29], and Sumatra [21] can capture data provenance from multiple users in their local stations and consolidate them in a single database, but those databases do not properly represent collaboration aspects of the research covered by Section 2.1, thus collaboration provenance is not included. Zhang *et al.* [97], Confucius [50, 86, 96, 99], and ProvDB [52] provide data and interaction provenance support, and use the collaboration-aware provenance models described in Section 2.1. The models proposed by Confucius and ProvDB need extended properties to represent some collaboration aspects, but the tools proposed by those papers are not able to capture these properties. Thus, there is a difference between the provenance types represented by the models and those supported by the tools.

2.3 Discussion and Opportunities

Figure 2.7 shows a timeline that helps understand how research has progressed in this field. Some of the publications are highly related and represent the evolution of the same research. In such cases, we treat them in a consolidated manner, thus linking these publications in the figure and handling them as a single approach. This topic has received much attention in recent years, but there are still some gaps to be further explored. In this section, we classify the selected approaches, answer the research questions introduced in Chapter 1, discuss the gaps that still exist, and present opportunities derived from those gaps.

In the Introduction, we highlight two challenges (C1 - how to collect provenance in a collaborative experiment; and C2 - how provenance can be used to make collaboration easier in this environment) and two research questions related to those challenges (R1 - How do existing tools store and collect provenance in a collaborative experiment?;

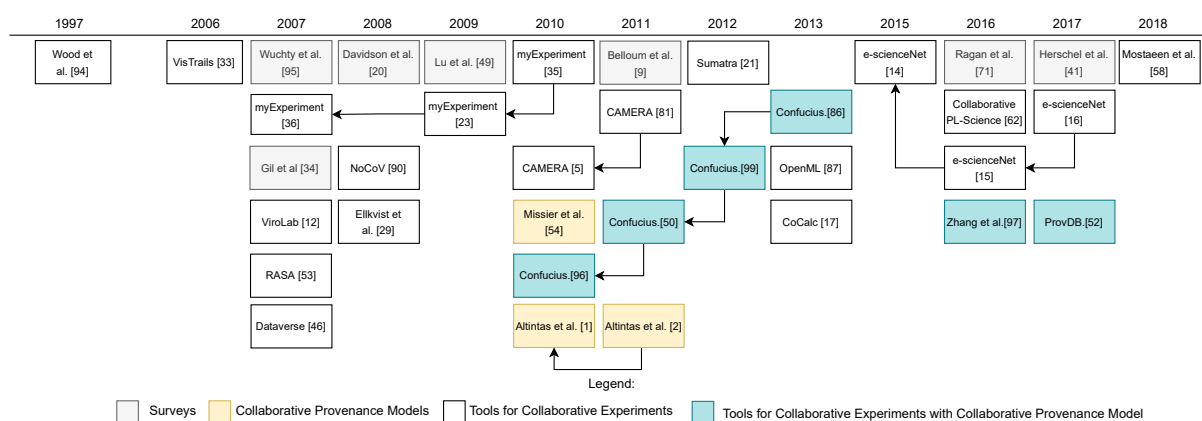


Figure 2.7: Timeline of selected publications

and R2 - how do existing tools use provenance to make collaboration easier in scientific experiments?). We answer them now.

R1: How do existing tools store and collect provenance in a collaborative experiment? To answer this question, we analyzed the available models for storing provenance in collaborative environments. Although significant progress has been made with those models, all of them present limitations (they do not deal with different workflow formats, or do not deal with workflow evolution). Models that can represent all the aspects we analyzed do so by using extended properties, which makes them difficult to query.

Regarding the available tools and how they collect provenance: Some tools (Dataverse [46], CAMERA [5], and myExperiment [24]) just provide storage for provenance, but do not collect it. Other tools (VisTrails [33], Ellkvist *et al.* [29], and Sumatra [21]) provide a way of consolidating the provenance collected from different users but lack support for other collaboration aspects. Finally, a few tools (Zhang *et al.* [97], Confucius [50, 86, 96, 99], and ProvDB [52]) use collaborative aware provenance models but still present some limitations.

R2: How do existing tools use provenance to make collaboration easier in scientific experiments? We conclude that the surveyed approaches fail to use the collected provenance to support the collaboration. Although Confucius [99], Zhang *et al.* [97], and ProvDB [52] are capable of collecting provenance of the collaboration process, they do not propose forms of using that valuable data to increase the efficiency and awareness of the process.

As illustrated in Table 2.2, most of the approaches support the composition phase of the experiment life cycle (especially the conception sub-phase). However, they are

mostly based on Workflow Management Systems and ignore the fact that many scientists use scripts in their experiments [67]. The only approaches that support experiments represented as scripts are ViroLab [12], Sumatra [21], CoCalc [17], and ProvDB [52]. However, ViroLab only addresses the reuse sub-phase of the experiment composition. Sumatra fully delegates the script composition to Git and presents several limitations for the shared provenance storage, such as a possible data loss depending on the network connection. Despite being quite complete, CoCalc [17] demands the scientist to be online in order to work, and that she works on the browser, which can be a tough change in the workspace, tools, and IDEs that the scientist is used to. It is possible to run applications from the CoCalc portal, but this is not the same as running them from the scientist's machine. It also presents several limitations on free accounts. Another point worth mentioning is that it does not properly capture the provenance of the experiment. It presents features like "time travel" and "log" that let users see the history of the files and activity on the project, but it is very high level and may not be enough to guarantee the reproducibility of the experiment, for example. ProvDB uses Git to handle version management and a provenance ingestor framework to capture other provenance data, but it is highly specialized in data science problems and is not well prepared for a general-purpose experiment.

Although versioning tools handle several collaborative needs of script building, they are software development tools that do not address specific problems in scientific research. These tools will not provide provenance capture and analysis support by default. Provenance is not just related to the obtained results but also the input data, intermediate results, etc. Trying to deal with this complexity without the proper tooling support could take much effort from the scientists and steal the energy that should be spent on research. Although ProvDB considers these challenges, it depends on the scientist being able to access an external tool (Git), a specific OS (UNIX), and demands the creation of ingestors to capture some provenance aspects. ProvDB is also focused on a specialized type of experiment (data science analysis), and does not address awareness during collaboration. Thus, *we must investigate and design provenance-aware tools that can handle composition, execution, and analyses of generic script-based experiments collaboratively, increasing the awareness of users during the process at the same time.*

The execution phase also lacks support. We could find only one approach that supports collaboration in this phase of the experiment life cycle. RASA [53] supports the execution phase by controlling access to physical resources such as equipment. Providing

provenance-aware support of the execution phase is crucial in collaborative experiments, since without it, important aspects of the collaboration may be lost. In fact, for reproducibility purposes, it is crucial to know which user executed each part of the experiment, where and under which conditions. Thus, the *support for the collaborative execution of scientific experiments needs more investigation.*

Some approaches support the analysis phase of experiments. Most of them allow scientists to comment on the experiment structure or results. Some approaches [12, 33, 52, 99] provide provenance gathering of the collaborative experiment that could help the analysis of the experiment. However, they do not provide a clear way to collaborate throughout the analysis, so they were classified without this phase of the life cycle in Table 2.2.

Temporality is well explored, with several approaches supporting asynchronous or real-time interactions. However, some features could be improved. When conducting an experiment in groups, it is important to know what happened in the experiment while scientists were offline, who did what, and in which part of the experiment (interaction provenance). It is also important to know if there is anyone online and in which part of the experiment they are working at. Although some tools let the users query for some of that information, it would be desirable that such information would be automatically shown to users, depending on the context of the experiment. Thus, *an interesting issue to examine will be ways of increasing the awareness of the scientists about the actions of their collaborators.*

As for concurrency control, most of the approaches use a pessimistic locking scheme. Pessimistic locking may work well in real-time scenarios, but it can be quite troublesome for asynchronous collaboration. VisTrails [33] and Ellkvist *et al.* [29] are the only solutions that work with an optimistic locking scheme, but they do not implement a merging mechanism capable of merging two workflow branches. Although VisTrails diff and analogy functionalities could help to merge two branches, they impose some additional steps for such a task and lack some basic merge functionalities like conflict resolution. Thus, *we need tools that work with optimistic locking and provide complete merge support in the composition of workflows.*

Also, in a collaborative environment, some collaboration tasks may perform better if treated with a pessimistic locking policy while others will benefit from an optimistic approach [70]. In experiments with files that are difficult to merge, scientists could opt to work with a pessimistic policy, while in others they may prefer to work with an optimistic

one. Existing tools only implement one of the policies, so if scientists want to use this tool, they are forced to use the implemented policy. Scientists must have the flexibility needed to interact in a way that is more appropriate to the use case in hand. Thus, ***tools that allow scientists to choose the more appropriate lock policy are needed.***

Sharing is well covered in the literature with a wide range of available solutions. Solutions address centralized sharing as well as peer-to-peer sharing, besides providing mechanisms for commenting and enriching the shared artifacts, making them easier to use. We believe that, in this aspect, there is no clear gap in the available tools.

We end up finding that none of the available tools are capable of using provenance to make collaboration easier in scientific experiments (related to R2). So, ***there is a need to investigate how to use the captured provenance to make collaboration easier in scientific research.***

Chapter 3

The Script-Based noWorkflow System

With the findings we discussed in Chapter 2, we decided to tackle the gap of support for collaboration in provenance-aware script-based approaches. We aim to take advantage of the progress already done in Script-based systems instead of building an entire Script-Based system from scratch. In particular, we were looking at tools that do not demand the scientist to previously specify the experiment in a workflow format. To do so, we evaluated well-documented open-source tools that we could build on top of and unleash the collaborative benefits. We ended up with two possibilities: noWorkflow [59] and Sumatra [21]. We chose noWorkflow because, when compared to Sumatra [21], it allows capturing provenance in a finer grain and presents a more elaborate visualization and search layers that facilitate the perception of the research evolution and access to the collected data. Before taking a deeper look at our approach and its implementation is very important to have some background knowledge about noWorkflow, the script-based system that we decide to extend on our implementation.

noWorkflow [59, 69, 64, 65, 63, 68] is a self-contained tool that is capable of collecting, storing, and visualizing provenance from Python scripts. noWorkflow works transparently – it analyzes the script execution, capturing and storing provenance information that can later be used in the analysis phase of the experiment. For that, assuming the experiment is written in a Python file named *script.py*, instead of running *python script.py*, scientists run *now run script.py*.

In this chapter we provide further details of noWorkflow functionalities and implementation: Section 3.1 presents the noWorkflow Architecture and Features, focusing on the Provenance Storage and Provenance Analysis modules, since these are the modified modules by our extension; Section 3.2 details noWorkflow’s gaps regarding collaboration.

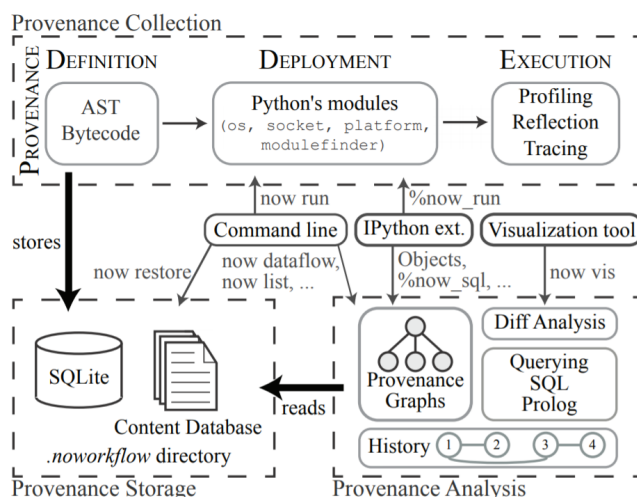


Figure 3.1: noWorkflow architecture [59].

3.1 noWorkflow Architecture and Features

The noWorkflow architecture can be divided into three modules (as shown in Figure 3.1): provenance capture, provenance storage, and provenance analysis. The provenance capture module is responsible for executing the Python script and also for collecting its provenance. The provenance storage module is responsible for storing both input and output files and capturing provenance (executed functions, parameters, variable values, etc.). The provenance analysis module is responsible for providing querying interfaces to the provenance storage. It also provides pre-built graphic visualizations that help scientists to better understand the experiment and its provenance. We propose extensions that add new collaboration features in the provenance storage and provenance analysis modules. Thus, in this section, we provide details of these modules.

3.1.1 Provenance Storage

In the provenance storage, data is stored in a folder called ".noWorkflow", which is automatically created by noWorkflow in the directory where the script execution was invoked (using *now run script.py*). Each script execution on that directory will correspond to a *trial* that is stored in the provenance storage. The information collected by noWorkflow is spread across an SQLite database and a content database.

The content database stores all files used during the execution of the script. This includes the script itself, versions of the used Python libraries and the input/output files of the experiment (as well as intermediate files). noWorkflow is able to capture these data

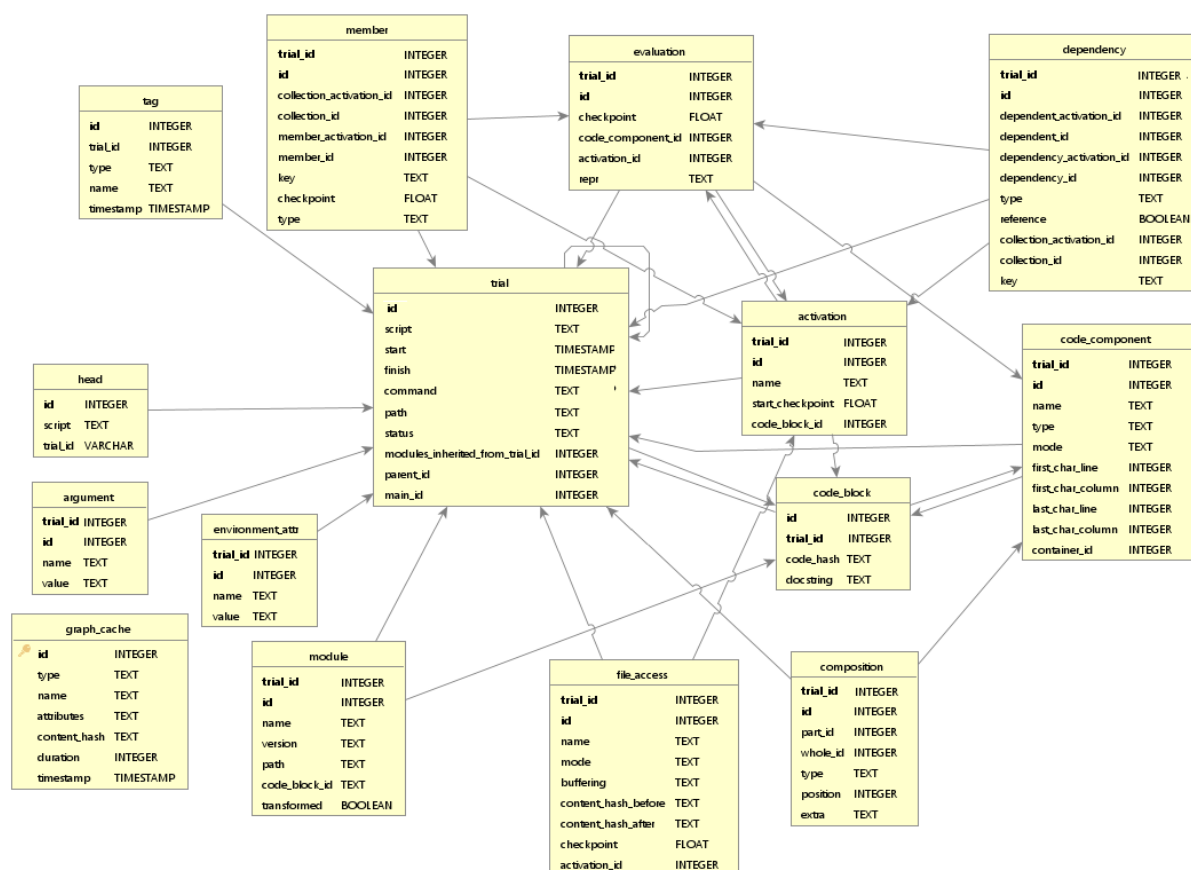


Figure 3.2: noWorkflow SQLite database model.

without the need to perform any changes on the scientist script, it is done by overriding the system file reads and writes function [59].

To store these files, noWorkflow first generates the SHA1 hash code of the file content. Then, the first two characters of the SHA1 hash code are used as the folder name, and the remaining characters are used as the file name. For example, a file with SHA1 hash code *2aa6c35c94fcfb415dbe95f408b9ce91ee846ed* will be stored in the folder *./noWorkflow/2a* with file name *ae6c35c94fcfb415dbe95f408b9ce91ee846ed*. Using SHA1 hash codes to identify the files prevents noWorkflow from storing the same file multiple times and makes it simple to identify multiple trials that depend on/use the same files. Additionally, placing the files in folders according to their hash prefix prevents OS limitations on the number of files that can be stored in a single folder.

The SQLite database stores information regarding the execution of the script, such as environment variables, execution parameters, function calls, and other information, depending on the level of granularity chosen by the user in the provenance capture. This database contains a trial table with an auto-increment identifier and several other tables that are related using foreign keys. The content database and the SQLite database are

linked together by the file hash code saved on the SQLite database. The full database model is illustrated in Figure 3.2.

Besides being a tool made to run Python scripts, noWorkflow’s source code is also written in the Python programming language. To interact with the SQLite database it uses SQLAlchemy [60]. SQLAlchemy is an open-source SQL toolkit and object-relational mapper (ORM) that serves to abstract the interactions with the database. Instead of writing SQL queries, the programmer uses the library commands to insert, update and delete entities of the database, and even to control the database schema. This abstraction layer makes the process of interacting with the database simpler, and also makes it easier to switch between database providers since the programmer does not need to deal with SQL variations between database providers.

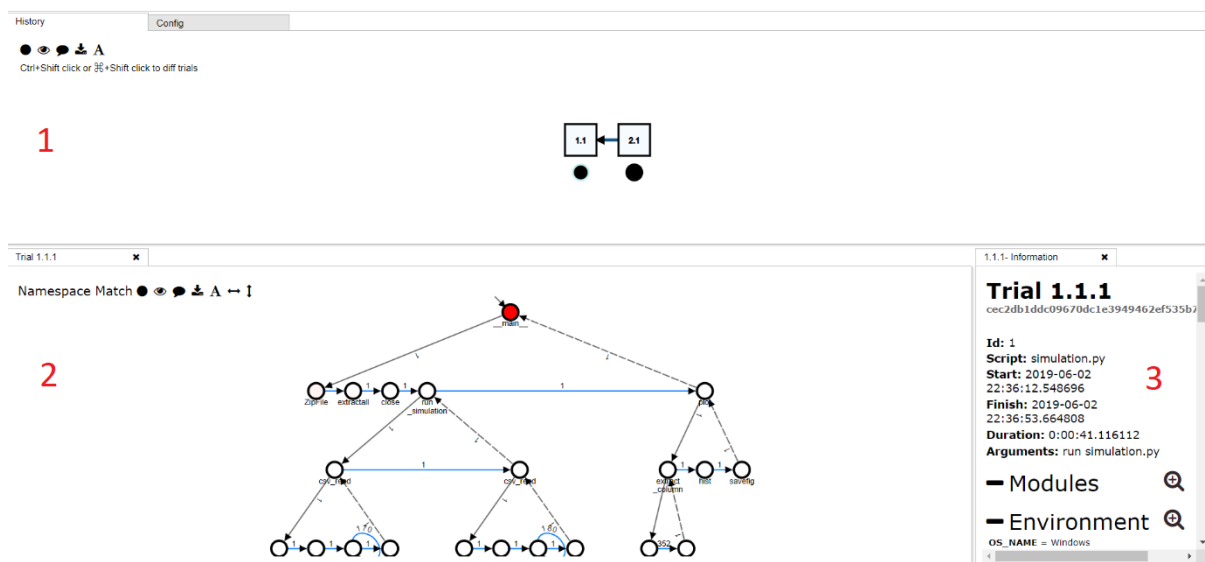


Figure 3.3: View of the provenance information in noWorkflow. (1) Graph with the trial history showing two trials; (2) graph view of trial 1.1.1 – nodes are function activations, and arrows denote sequence of calls (solid arrows) and returns (dashed arrows); (3) Details of trial 1.1.1.

3.1.2 Provenance Analysis

The provenance analysis module provides several querying options: command line queries, Prolog queries, and SQL queries (which need to be run directly in the SQLite database). noWorkflow also has a visualization tool (shown in Figure 3.3) activated by the command *now vis*. The *now vis* command starts a web interface that allows viewing the collected provenance, comparing two trials, inspecting accessed files etc. It is a quick way of checking dependencies, function activations, parameters, etc. This visualization tool is

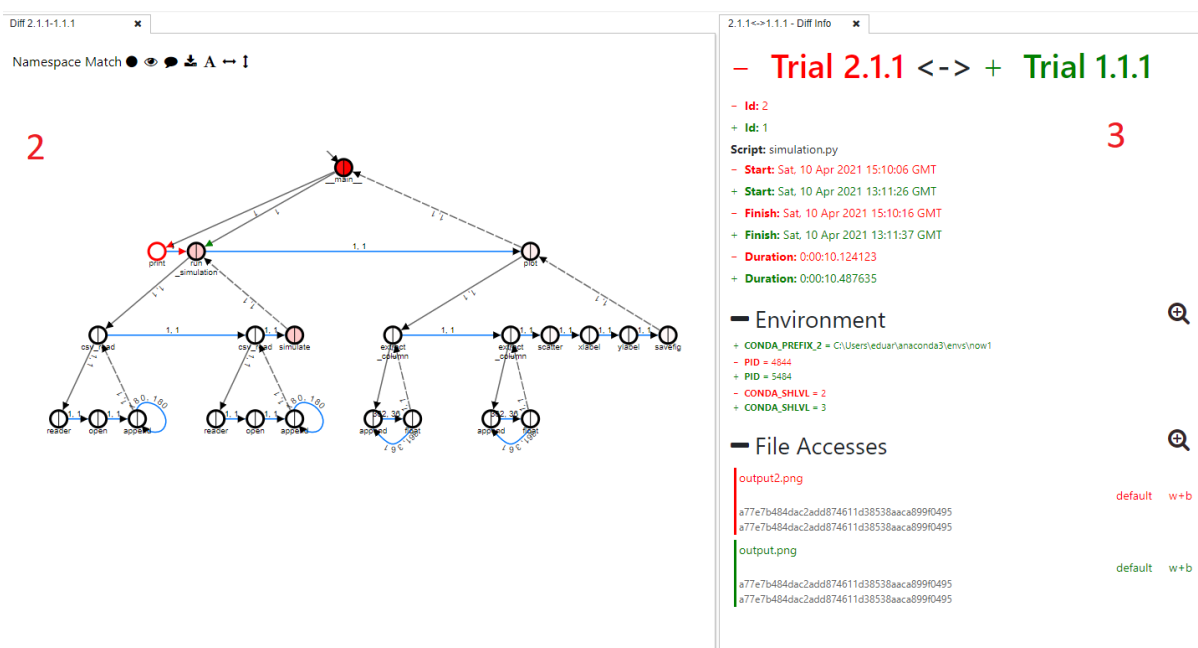


Figure 3.4: View of trial diff in noWorkflow. (2) graph view of trial 1.1.1 and trial 2.1.1 diff, considering function activations; (3) Details other differences between trial 1.1.1. and trial 2.1.1

part of the noWorkflow source code, and it is built using the Python framework Flask [39].

Panel (1) of Figure 3.3 shows a graph with the trial history. Squares represent a script version (1.1, for instance), while black circles below them represent an execution of that script version (a trial). In the Figure, there is one execution of the script version 1.1, and this is labelled as trial 1.1.1 (the trial label is not shown in Panel (1)). The trial history graph shows that the scientist modified the script version 1.1, creating the script version 2.1. An arrow from square 2.1 to square 1.1 represents the evolution of the script code. The scientist then executed this 2.1 script version (the black circle below the 2.1 square represents this execution as trial 2.1.1). All of this history is captured automatically and transparently by noWorkflow, without the need of using a version control system.

Panels (2) and (3) show details of the trial that is selected in Panel (1). The blue circle around the trial 1.1.1 on Panel (1) means that this is the selected trial detailed in Panels (2) and (3)). Panel (2) shows a graph that represents the execution of trial 1.1.1. Each node in this graph corresponds to the execution of a function in the script (which noWorkflow calls *function activation*). Solid black arrows represent the start of activations, blue arrows represent a sequence of calls within activations, and dashed arrows represent returns, the node colors represent their duration, in a scale where red represents the slowest function activations, and white, the fastest ones[68]. Finally, Panel (3) presents details of trial 1.1.1, such as execution duration, module dependencies, arguments, environment variables, etc.

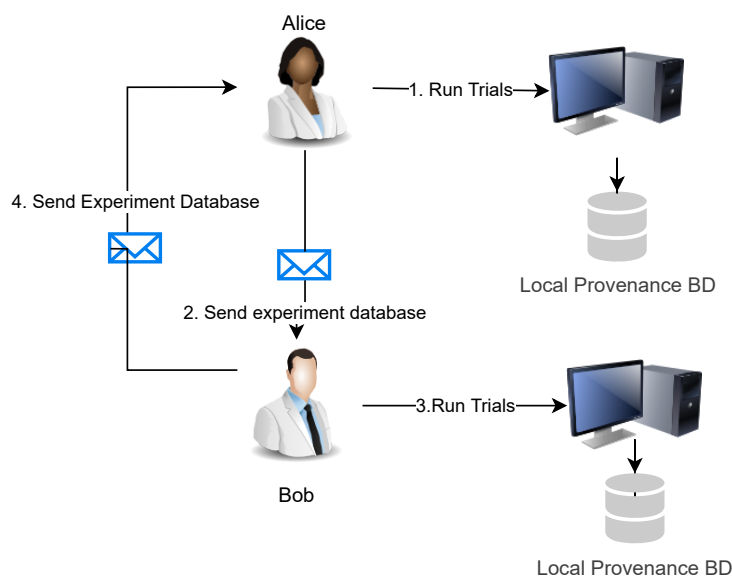


Figure 3.5: Illustrates two scientist (Alice and Bob) trying to collaborate using noWorkflow.

It is also possible to compare trials visually, for that the user must select two trials in Panel (1) instead of one. When this happens, Panel (2) and Panel(3) show a comparison between these trials can also compare the graphs of two trials, as illustrated in Figure 3.4. In this case, Panel (2) shows that the only difference in function activation it is the inclusion of the node "print" on Trial 2.1.1. Note that Panel (3) now only shows the trial information that is different in the two experiments. We can observe that the experiments were started and finished in a different time, there is also a small variation on experiment duration, the process id is different between the experiments, and the trial 2.1.1 generated a file called "output2.png" while trial 1.1.1 generated a file called "output.png". It is also possible to see that only the file name changed since the hash of those two files is the same.

3.2 Lack of Collaboration Support

noWorkflow was designed for single users. Consequently, collaboration is not supported by the tool itself, and scientists must do some workarounds to work collaboratively when using noWorkflow. As an example, suppose that scientists Alice and Bob are working collaboratively on an experiment using noWorkflow *as is*. As illustrated by Figure 3.5, Alice writes the script and runs the first trial. noWorkflow automatically captures provenance data of the executed trial. Then, she compresses the *provenance storage* (the .noWorkflow folder) and sends it to Bob by e-mail along with the script of the experiment. Bob then

uncompresses the folder, makes some changes to the script and runs it. He then does the same – compresses the folder and sends it back to Alice.

Although the strategy of using email (or another external tool) to share the experiment and the provenance database allows them to work collaboratively, it adds the burden of compressing and sending files back and forth, but, most importantly, it prevents Alice and Bob from working in parallel, which is unthinkable in the present days. To address this problem, Alice could save the provenance storage in a shared network folder, but that would imply in users having to be in the same network, and, most importantly, since noWorkflow was not designed to work this way, several concurrency problems may occur.

With this example is possible to visualize that collaboration between people involved in the experiment must be integrated with the Script-based System, and using external and unconnected tools to collaborate will typically result in work limitations and overhead to the researchers. In the next chapter, we propose a solution to this problem.

Chapter 4

Collaborative approach

With the growth of *in silico* experiments, the problem of managing this kind of experiment has received some attention from researchers. As we identified in Chapter 2, one of them resides in the lack of support for collaboration in provenance-aware script-based approaches. Some authors propose provenance models to capture the collaboration in this environment. Missier *et al.* [54] and Altintas *et al.* [1, 2] present provenance models that are capable of representing collaborative experiments. Although this is an important subject, scientists still need tools that can collect this data and present it properly to scientists involved in the research.

Workflow management systems provide some collaborative features, like Confucius [99] and VisTrails [33], however, they are workflow-based solutions, which requires the experiment to be described in a specific format – they do not support script-based experiments. While script-based systems provide very few features to support collaboration, scripts are a trendy way of representing *in silico* experiments [66] and thus, supporting them is a must for the scientific community.

When we evaluate script management, it is natural to consider some software engineering tools like versioning systems. Although versioning systems (*e.g.* Git [78] and Mercurial [61]) deal with several collaborative aspects of script construction, they are aimed at software development and concerned only with the development phase of script composition. Specific problems of scientific research, such as the analysis phase, reproducibility of the experiment, and the consequent need for provenance collection in the execution phase of the experiment [41], cannot be solved using only versioning tools.

Besides that, the more traditional provenance-aware script-based solutions [59, 48] are focused only on collecting provenance and do not deal with collaboration aspects. Even

solutions that address collaborative aspects of the experiment, such as ProvDB [52] and Sumatra [21], still have some limitations. Despite having a feature that allows multiple users to use the tool, Sumatra does not synchronize the data files of the experiment when collaborating, and the researcher is in charge of synchronizing this data. Sumatra also requires that the server be accessible during the execution of the experiment with the risk of losing information if the server is not reachable. ProvDB, on the other hand, has limitations such as: it depends on an external tool (Git); it works only on a specific operating system (UNIX); it focuses on a specific type of experiment (data science analysis); finally, it demands the development of ingestors to capture provenance data.

To fill in this gap and support the execution of collaborative experiments, we propose an approach that allows aggregating, synchronizing, and analyzing the provenance information of an experiment conducted collaboratively by several scientists on a Script-based system. To achieve that, our proposed approach consolidates the provenance database of the various scientists who are involved in the experiment.

This chapter details this proposed approach and its implementation, and proceeds as follows: Section 4.1 presents our conceptual approach, detached from any Script-based system; Section 4.2 details the implementation of that approach on top of noWorkflow. Finally, in Section 4.3 we use the provenance model classification and taxonomy presented on Chapter 2 to classify our approach and compare it with existing tools.

4.1 Proposed approach

Our approach aims to design a provenance-aware Script-based approach ready for collaboration. In our view for scientists to have all the benefits that provenance can provide, the Script-based system must simultaneously provide the provenance data to all the scientists involved. Scientists should be able to work in parallel and the system should be able to store and consolidate the provenance data in single storage. The Script-based system should help larger groups to organize their experiments and be able to make "cross experiments" queries during the analysis phase.

In order to provide flexibility to our users, we propose the possibility of having two types of architectures: a "Central Node Collaboration Architecture" and a "Peer-to-peer Collaboration Architecture". In the "Central Node Collaboration Architecture", there is a server working as the single source of truth for the provenance data, with the benefit of a scientist only needing be aware of the "provenance server". This eliminates any need to

communicate with other machines, but it implies in an infrastructure cost. On the "Peer-to-peer Collaboration Architecture", on the other hand, there is no need for setup or infrastructure cost, but the collaborators in the experiment must be able to communicate between them (across a network) in order to share provenance data.

In Section 4.1.1 we cover the details of the "Central Node Collaboration Architecture", while in Section 4.1.2 we cover the "Peer-to-peer Collaboration Architecture". Finally, in Section 4.1.3 we cover the support to multiple experiments and discuss its importance.

4.1.1 Central Node Collaboration Architecture

In our approach, we use concepts of DVCS (Distributed Version Control Systems) [61, 78], which are already consolidated in Software Engineering. The idea is that each scientist has his/her own provenance storage containing the information of the experiment. This repository is then synchronized with a central repository, where it would be available to other interested parties. To achieve these goals, we propose two new operations (*pull* and *push*) to allow the scientists to send and receive data from another provenance storage. Both operations receive an *address* as a parameter, indicating on which provenance storage the operation should be made.

Figure 4.1 illustrates how the proposed approach works. In this scenario, two scientists are working together on the same experiment using a central node as the provenance repository of the experiment. When one scientist executes the *pull* operation, her machine asks the server for the list of trials that are stored on the server. Then, it compares this list with the local provenance database, decides which trials have to be sent to the central node, and sends them out. The *push* operation has a very similar behavior but, in this case, the scientist machine discovers which trials on the server are not on the local storage and then retrieves them from the server.

This approach allows the scientists to work offline without access to the central node. When back online, scientists can share their work (using the *push* operation) and obtain the progress made by others (using the *pull* operation). This has the additional advantage of centralizing the provenance data of the whole experiment into a single provenance database, which could be very helpful for the scientists working on the experiment. Some of the advantages reside in facilitating scientists to reproduce the trials executed by others, attributing the correct credit for those involved, and making it easier for third parties to understand and reproduce the experiment.

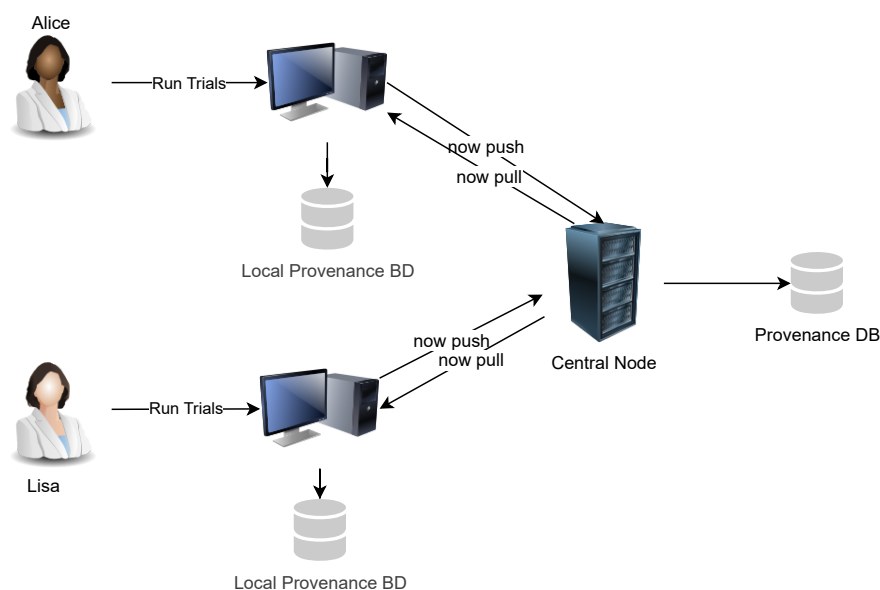


Figure 4.1: Central node collaboration architecture.

To illustrate a use case where these operations are used, we can return to the example given in Chapter 3, but now considering that Alice and Lisa are working collaboratively on an experiment using our proposed approach. Alice writes the script and runs the first trial. The Script-based tool automatically captures provenance data of the executed trial. Then, she uses the *push* command to send provenance information to the central node server. Lisa then uses the *pull* command to receive experiment data, changes the script and runs it. Lisa then does the same – uses the *push* command to send information to the central node where it will be available for Alice and other scientists. This will produce a single provenance database regarding the whole experiment, with several benefits in terms of analysis of provenance data and experiment reproducibility. Another important aspect is the capability of working in parallel and still being able to consolidate the provenance into a single provenance database, which is not possible in some tools [21].

4.1.2 Peer-to-peer Collaboration Architecture

Although we see much value in having a central node to store provenance data, we also understand that this might not be possible in some scenarios. So, in our approach, we also support a peer-to-peer model, allowing scientists to share data without the need of a central infrastructure. This model also uses the *push* and *pull* operations that are used for the central node approach.

As shown in Figure 4.2, in this scenario, each scientist has their own provenance database, but there is no central provenance storage.. If Alice wants to share or retrieve

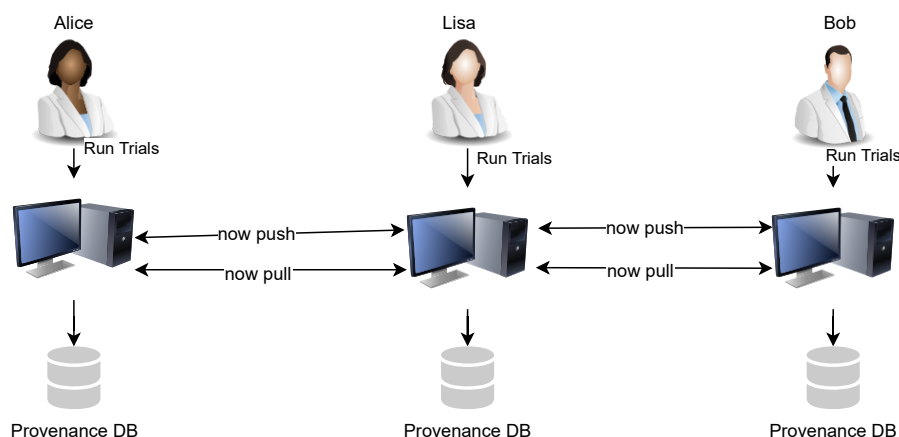


Figure 4.2: Peer-to-peer collaboration architecture.

provenance data from Lisa, she connects directly with Lisa using the pull and push commands. And after that, Lisa can connect with Bob and transmit the received data.

This approach has the clear advantage of not needing a dedicated infrastructure previously setup. However, the biggest disadvantage of this approach is the lack of a central provenance storage always available working as the ground truth.

4.1.3 Multiple Experiment Support

We also propose the support for multiple experiments. This is relevant since most existing tools that capture provenance of scripts allows only for a single experiment (with several trials). New experiments are stored in different provenance databases. The support of multiple experiments, allied with the central node approach, allows a scientific institution or a research group to concentrate provenance data of multiple experiments into a single provenance storage.

This approach has several uses in increasing collaboration, such as scientists could identify similar experiments, *i.e.*, experiments that use the same dataset or the same algorithms; scientists could search the database to identify software pieces that they can reuse; and also identify implicit collaborations (when one experiment uses another experiment output as input, for example, creating a collaboration even though the scientists involved do not meet or talk to each other).

4.2 Approach Implementation

In this section we cover the implementation of our approach. As mentioned in chapter 3, we took advantage of the progress already done in the field and choose noWorkflow [59, 68] to extend and implement our proposed approach. Since noWorkflow already has an embedded HTTP application that is used in the visualization tool (Figure 3.3), we decide to build on top of that application and build Restful APIs [74] to extend it. To make it easier to switch between the central node and the peer-to-peer operation models, we decide to use the same code base on the server and the client machines, as well as the same database structure.

To implement the *push* and *pull* operations, we build Restful endpoints on the noWorkflow HTTP application to return/receive experiment data. We then add new commands to the noWorkflow command-line interface to interact with the APIs and update the local provenance store. Since we use Rest APIs, it was a natural decision to use an URL as the *address* parameter, which indicates the address of the provenance database.

Although noWorkflow already captures and stores really detailed provenance information, and with the multiple experiment support it can represent other aspects of the experiment, it is not possible to predict all the possible use cases and storage needs. So we also add the possibility of the scientist adding annotations to the experiment. These annotations can hold miscellaneous data.

As mentioned in Chapter 3, noWorkflow provenance storage is composed of an SQLite database, which stores structured provenance data; and a content database, which stores file contents. Thus, we split the problem in synchronizing these two databases. The synchronization algorithms are explained in the next subsections. In the Section 4.2.1 we cover the implementation of the content database synchronization; In Section 4.2.2 we explain how we do the SQLite database synchronization; In Section 4.2.3 we detail how we implemented the multiple experiment support described in Section 4.1.3; In Section 4.2.4 we detail the annotation support we added. Finally, in Section 4.2.5 we summarize our contributions to noWorkflow.

4.2.1 Synchronizing the Content Database

Starting with the content database, we build three API endpoints: *list files*, *get file content* and *store files*. To implement the *push* operation, our noWorkflow client application first calls the *list files* endpoint and retrieves the hash of all files existing in the server. Then,

it compares this list with the local storage, identifying which files exist in the local storage that do not exist in the server. Since noWorkflow uses the hash of the file content as an identifier, this operation is straightforward. With the list of files that have to be sent to the server, the noWorkflow client application calls the *store file* endpoint sending the files contents to be stored. In Algorithm 1 we illustrate the algorithm used for the push command.

Algoritmo 1 Push command

Input: `url` #Url where the noWorkflow instance receiving data is running
`textbf`#Export Files

```

serverFilesHash ← httpGet(url +' /collab/files')
localFilesHash ← localProvanence.GetFilesHash
filesToExport ← [for id in localFilesHash where id not in serverFilesHash]
fileData ← compress(localProvanence.GetFileData(filesToExport))
httpPost(fileData, url +' /collab/files')

```

#Export Trial data

```

serverFilesHash ← httpGet(url +' /collab/files')
localFilesHash ← localProvanence.GetFilesHash
filesToExport ← [for id in localFilesHash where id not in serverFilesHash]
fileData ← compress(localProvanence.GetFileData(filesToExport))
httpPost(fileData, url +' /collab/files')

```

To implement the *pull* operation, our noWorkflow client application also calls the *list files* endpoint and retrieves all files existing in the server. However, instead of determining "missing" files on the server, it now looks for files in the server that do not exist in the local provenance storage. With the list of files that need to be retrieved, it calls the *get file content* endpoint to get files contents and then stores them in the local database. In Algorithm 2 we illustrate the algorithm used for the pull command.

Considering scientific experiments nowadays, we understand that those files could be large and that this could be a problem. To mitigate that, all endpoints use gzip [26] compression to minimize network traffic. We understand that, in many cases, this may not be enough. Thus, improving this in future versions of our tool is on our radar as future work.

4.2.2 Synchronizing the SQLite Database

To synchronize the SQLite database, the logic is similar but the implementation is more complex. The noWorkflow SQLite database consists of a relational database with many

Algoritmo 2 Pull command

Input: `url` #Url where the noWorkflow instance transmitting data is running
#Import Files

```
serverFilesHash ← httpGet(url + ' /collab/files')
localFilesHash ← localProvanence.GetFilesHash
filesToImport ← [for id in serverFilesHash where id not in localFilesHash]
fileData ← httpGet(filesToImport, url + ' /collab/files')
localProvanence.storeFiles(uncompress(fileData))
```

#Import Trial data

```
serverTrialsList ← httpGet(url + ' /collab/trialsids')
localTrialList ← localProvanence.GetTrialsIds
trialsToImport ← [for id in serverTrialsList where id not in localTrialList]
bundle ← httpGet(trialsToImport, url + ' /collab/bundle')
trialData ← deserialize(uncompress(bundle))
localProvanence.StoreTrials(trialData)
```

entities and relationships whose referential integrity must be preserved. To help implement the commands, we also build three API endpoints: *list trials*, *retrieve trial data* and *store trial data*. Considering the database is "trial oriented", we use the trial entity as the starting point of the synchronization process.

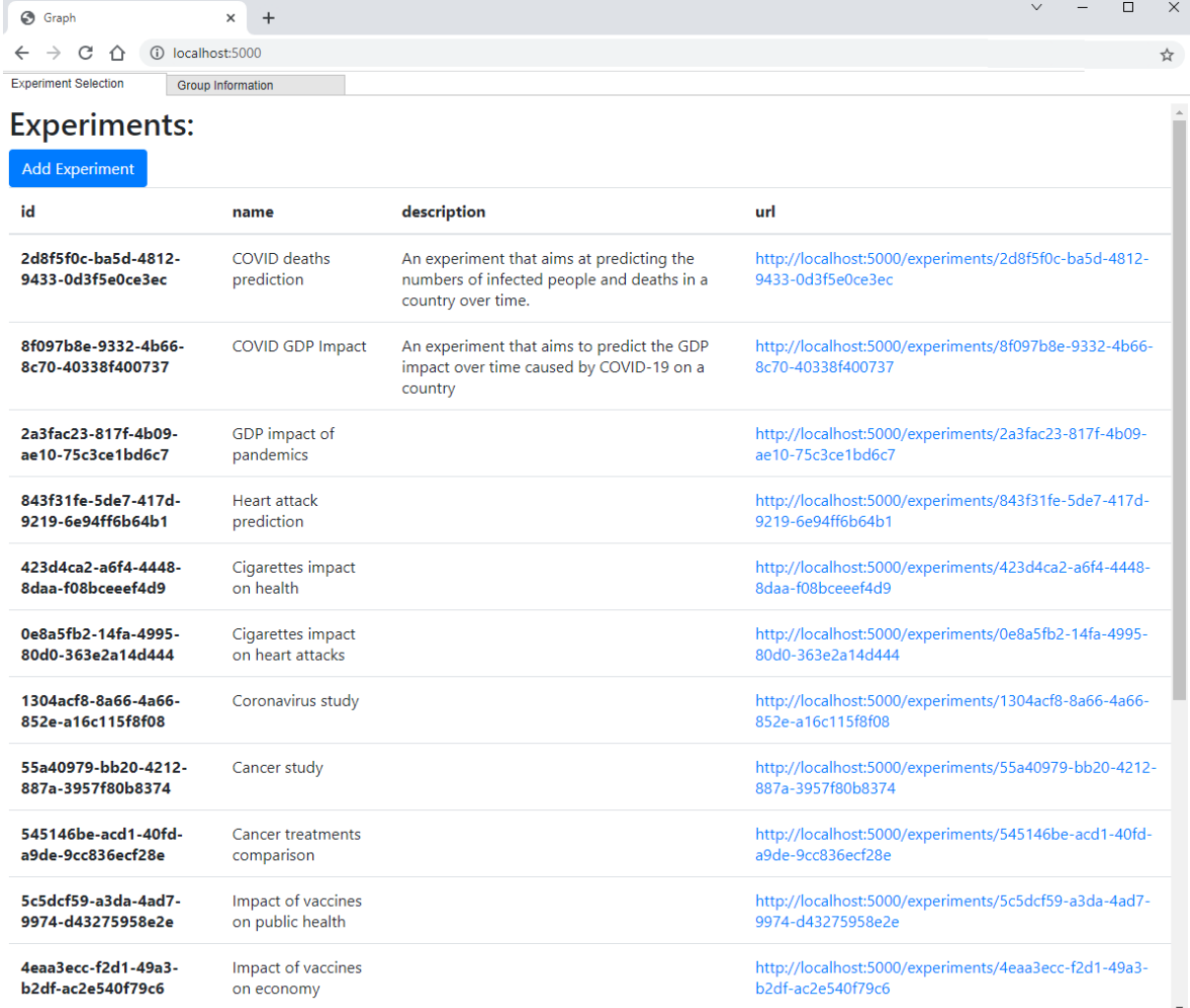
The process for implementing the *push* operation is similar to the one described for the content database. We determine which trials must be sent to the server and use the *store trial data* to save information on the server. Since noWorkflow uses an auto-increment integer as a trial identifier, it is not easy to identify if a trial on the server is the same as one in the local storage. One approach could be to compare all trial data (including child entities data), but this could be very time-consuming. Another critical issue is that the trial table has a self-reference to identify the "trial parent", which could demand a prohibitive comparison effort on synchronization. To deal with that, we have modified the trial table, renaming the trial id to "sequence key" and creating another column as the trial id where we store an UUID (Universally Unique Identifier) [47]. We also modified all other tables with foreign key references to the trial table in order to correctly deal with the new trial id format. Since noWorkflow uses SQLAlchemy as mentioned in Chapter 3, this modification does not involve SQL commands. Instead, it requires modification on the Python objects that maps those SQL tables. In addition, we updated the noWorkflow *run* command to generate this UUID when it is storing the trial information on the database. This identifier is unique globally, and we use it on the *list trials* endpoint and on the noWorkflow client application to infer trials that must be sent to the server.

To send trial information to and from the server, we choose to use the JSON (JavaScript Object Notation) [8] format. The noWorkflow source code already has Python objects to represent entities, but it is a non-serializable representation of the database tables that is used by SQLAchemy in database operations. Thus, to be able to transfer data between client and server applications we created a serializable object representation to each table that needs to be synchronized that we called a "lightweight object".

The synchronization works as follows. The entities are retrieved from the database using SQLAchemy in its model object representation. Then, they are converted into its lightweight representation. We also created the concept of "bundle", which is an object containing lists for every entity being synchronized. We then send a serialized "bundle" object, using a very standard serialization process to transform it into the JSON format and send it in the body of the HTTP request. A sample of the bundle serialized object is available on Appendix B. When the server receives JSON content, it deserializes it, converts it into the database object model representation, and then stores it on the server provenance storage.

All the tables of the noWorkflow's original database version (illustrated on Figure 3.2) are synchronized, except for the "graph_cache" and the "tag" tables. The "graph_cache" is not synchronized for two reasons: it does not have a direct relationship with the trial table, making almost impossible to identify what has to be synchronized. Additionally, this table is used for improving the visualization performance and its data is temporary. The "tag" table was not synchronized because of a design decision. Tags in noWorkflow are divided into three dot-separated parts that work in a hierarchical fashion. From left to right, the first part is a sequence that identifies trials with the same content hash of the script that is being executed (same source code), the second part identifies trials with the same command line (same parameters), and the last part is a sequence to differentiate trials with same code and same command. Since in our collaboration extension the scientists can work offline simultaneously, they can end up with the same tag codes for different trials, which can generate conflicts when synchronizing. So, we decide to not synchronize the tag table. Upon import, each trial receives a new tag, which is not related to the tag in the source database. The new tag is generated by comparing the hash of the script content and the command executed with the trials in the destination database. Thus, instead of using the tags to uniquely identify a trial in multiple databases, the trial id must be used.

For the *pull* operation, the noWorkflow client application also calls the *list trials* end-



The screenshot shows a web browser window with the address bar at localhost:5000. The page title is 'Graph' and the URL is 'localhost:5000'. Below the browser window, there are two tabs: 'Experiment Selection' and 'Group Information'. The main heading is 'Experiments:' followed by a blue 'Add Experiment' button. Below this is a table with the following data:

id	name	description	url
2d8f5f0c-ba5d-4812-9433-0d3f5e0ce3ec	COVID deaths prediction	An experiment that aims at predicting the numbers of infected people and deaths in a country over time.	http://localhost:5000/experiments/2d8f5f0c-ba5d-4812-9433-0d3f5e0ce3ec
8f097b8e-9332-4b66-8c70-40338f400737	COVID GDP Impact	An experiment that aims to predict the GDP impact over time caused by COVID-19 on a country	http://localhost:5000/experiments/8f097b8e-9332-4b66-8c70-40338f400737
2a3fac23-817f-4b09-ae10-75c3ce1bd6c7	GDP impact of pandemics		http://localhost:5000/experiments/2a3fac23-817f-4b09-ae10-75c3ce1bd6c7
843f31fe-5de7-417d-9219-6e94ff6b64b1	Heart attack prediction		http://localhost:5000/experiments/843f31fe-5de7-417d-9219-6e94ff6b64b1
423d4ca2-a6f4-4448-8daa-f08bceef4d9	Cigarettes impact on health		http://localhost:5000/experiments/423d4ca2-a6f4-4448-8daa-f08bceef4d9
0e8a5fb2-14fa-4995-80d0-363e2a14d444	Cigarettes impact on heart attacks		http://localhost:5000/experiments/0e8a5fb2-14fa-4995-80d0-363e2a14d444
1304acf8-8a66-4a66-852e-a16c115f8f08	Coronavirus study		http://localhost:5000/experiments/1304acf8-8a66-4a66-852e-a16c115f8f08
55a40979-bb20-4212-887a-3957f80b8374	Cancer study		http://localhost:5000/experiments/55a40979-bb20-4212-887a-3957f80b8374
545146be-acd1-40fd-a9de-9cc836ecf28e	Cancer treatments comparison		http://localhost:5000/experiments/545146be-acd1-40fd-a9de-9cc836ecf28e
5c5dcf59-a3da-4ad7-9974-d43275958e2e	Impact of vaccines on public health		http://localhost:5000/experiments/5c5dcf59-a3da-4ad7-9974-d43275958e2e
4eaa3ecc-f2d1-49a3-b2df-ac2e540f79c6	Impact of vaccines on economy		http://localhost:5000/experiments/4eaa3ecc-f2d1-49a3-b2df-ac2e540f79c6

Figure 4.3: Experiment portal screen

point and retrieves all trials existing in the server. But now, it obtains which trials must be retrieved. Then it calls the *retrieve trial data* endpoint to get the trials information and then stores it in the local provenance storage. The processes for comparing trials, serialization, and deserialization are analogous to the ones we explained for the *push* operation.

Since we have the same source code for the noWorkflow client application and the server, the implementation of the peer-to-peer model uses the same mechanism already explained. In the peer-to-peer model, the scientist just runs the HTTP application embedded on her machine, and it can assume a server role. The only prerequisite is that the other scientist is allowed to reach this machine using the HTTP protocol.

Figure 4.4: Add experiment screen

id	name	Members	Actions
440fe7b8-72f1-4adf-8dcf-f807153f41b8	Economic Impact Study Group	Alice Lisa Peter Regina Bob	Add User Delete Group
02f0db7c-6f04-4202-b68f-c793b4df9a88	Death Prediction Study Group	Carlos Monica Paul	Add User Delete Group

Figure 4.5: Group management screen

4.2.3 Support to Multiple Experiments

We also extended noWorkflow to create the concept of multiple experiments using a single provenance storage. We see this as a very important improvement that will allow scientists to easily compare experiments, understand "hidden" relationships between them, commonly used data, etc. To implement this, we have created a table on the SQLite database representing the experiment entity, and added a foreign key on the trial table referring to the experiment.

In addition, we have created a new screen (illustrated in Figure 4.3), which allows scientists to list existing experiments and its descriptions. We made this screen to be the home screen of the noWorkflow HTTP application. It is also possible to navigate to the experiment's provenance details (same screen illustrated in Figure 3.3) from that screen by clicking on the URL link of the experiment.

To illustrate the process of creating an experiment, suppose that scientists Alice and Bob want to start a new experiment that they will conduct together. For simplicity, we are considering the central node approach here, although an analogous process could be applied for the peer to peer architecture.

To create a new experiment, Bob clicks on the "Add Experiment" button at the home screen and fills in the name and description of the experiment (screen illustrated in Figure

4.4). After confirming the operation Bob is redirected again to the home screen, where the new experiment is now listed. Notice that at this moment the experiment does not exist on Bob's machine – it only exists on the server. To create the experiment on his machine, Bob has two options. The first option is to copy the URL available in the experiment list at the home screen and then use it on a *pull* command. In this case, noWorkflow will download the empty repository to his machine and he can start working on it. In second option, since there is no strong link between the local repository and the remote one, he can also start to work on the experiment in the same way he used to do in the previous version of noWorkflow, using a command such as *now run experiment.py*, and then use the URL only when he wants to sync his work with Alice. Since Bob has already created the experiment, when Alice enters the portal home screen she can see the experiment and use the *pull* command, using the experiment URL as a parameter, to retrieve Bob's progress at the experiment. After doing the work she needs, she can also use the *push* command, informing the same URL to make her work available to Bob.

The multiple experiment feature can also help with the network traffic problem since users will often benefit from a file already being on the server because another user in another experiment has already stored the information there.

As mentioned in Chapter 2, collaboration typically involves resources from multiple organizations. To allow our approach to support this aspect and be able to manage not only multiple experiments in the same organization but also into multiple ones, we create the concept of "groups" in the noWorkflow database. These groups could be used to represent organizations or even different research groups within the same organization. The user can navigate to the "Group Information" tab on the home screen, where it is possible to create and delete groups and members to a group. This screen is illustrated in Figure 4.5.

4.2.4 Annotation support

Additionally, we add annotation support to noWorkflow. For that, we modified the SQLite database creating a new table called "extendedAnnotation" where the created annotations would be stored. Besides the annotation itself, this table stores: a description for that annotation; the "annotation level" (annotations can be done in different levels, in the experiment level or in the trial level); the "related trial" or "related experiment", depending on the annotation level; the "annotation format" (text, JSON, XML, etc.), making it easier to interpret; and the "annotation type", used to classify the annotation regarding

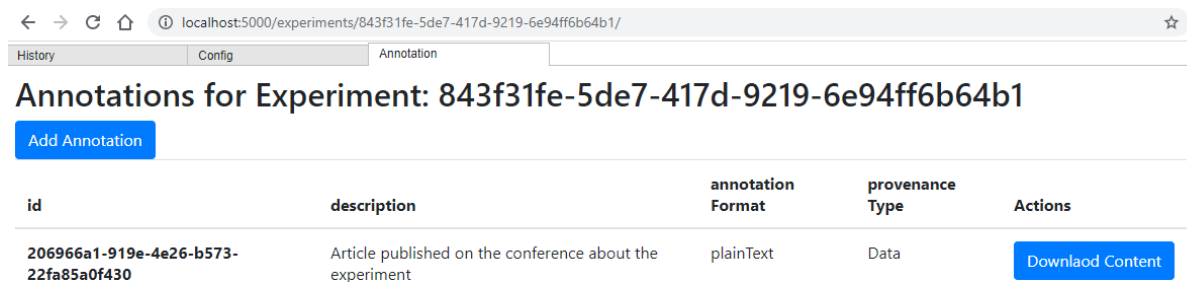


Figure 4.6: Manage annotation screen

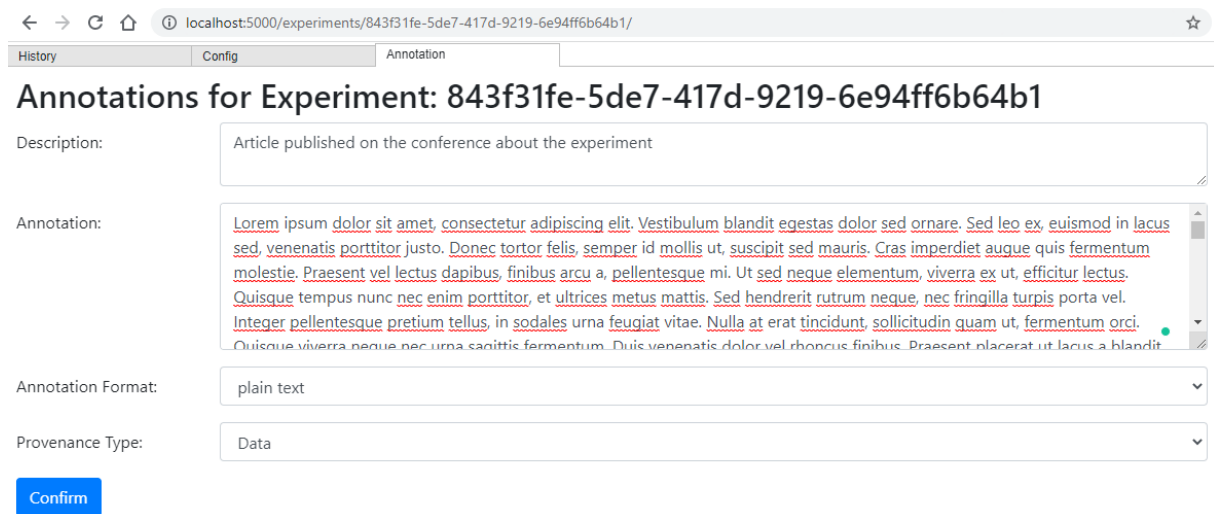


Figure 4.7: Add annotation screen

the provenance type (Data, Interaction, Visualization, Insight, or Rationale), making it easier to query and extract the value of the recorded information.

There are two ways of adding an annotation, one of them is using a REST API, the other one is using the user interface. In the case of the user interface, we added a new tab, the "Annotation" tab, on the experiment visualization screen where the scientist can manage the annotations of an experiment. This tab is illustrated in Figure 4.6. Here the scientist can add new annotations, see annotations already made and download the annotation content. When the user clicks on add annotation a new screen is shown (Figure 4.7), where the scientist can input relevant data for the new annotation.

Notice that annotations can also be made on the trial level. To manage annotations on the trial level, the scientist has to click on a particular trial, and on the trial details she has to click on the "view annotations" option as shown in Figure 4.8. Then, the user will navigate to the same screen illustrated in Figure 4.6, but now on the trial level.

As previously mentioned, it is not possible to predict all the possible uses for anno-

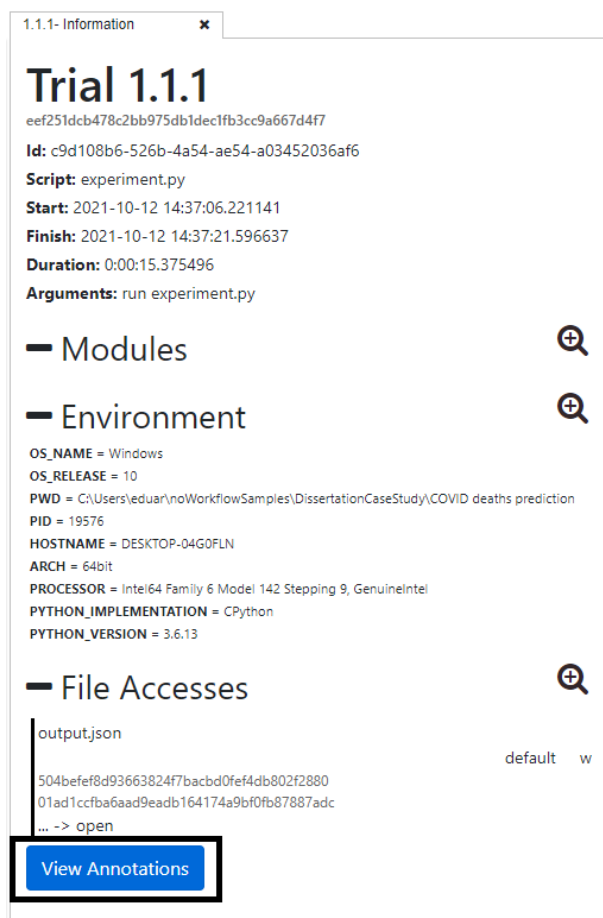


Figure 4.8: The new "view annotations" option on trial details. Using that option, the user can navigate to manage extended annotations within a trial.

tations, but one really interesting use we envision is the possibility to store provenance from other tools and formats. noWorkflow only captures provenance for python scripts. Imagine that as a part of the experiment, there is an execution of an external command in another tool, that also captures provenance information. Now it would be possible to take that information and store it as an annotation on the same provenance storage, and since the annotations can be added programmatically through the REST APIs, this process can be fully automated.

4.2.5 Implementation Summary

On Figure 4.9, we illustrated the SQLite database model after our implementation. The updates were: the "experiment" table that allows us to have multiple experiments in a single provenance database; the modification of the trial id column to a UUID, making it easier to identify trials globally; tables "user", "group", and "memberOfGroup" that allows to represent the user and also the possibility to group users; the "extendedAnno-

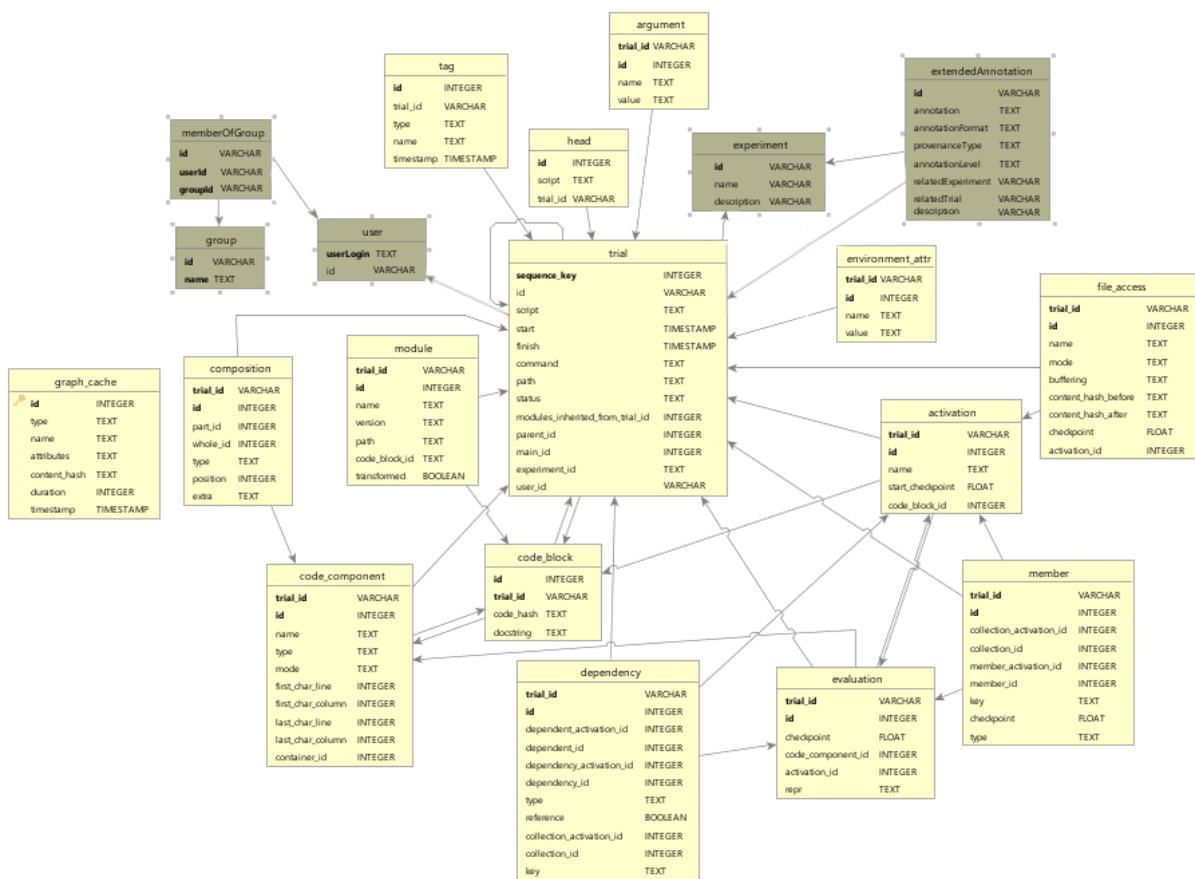


Figure 4.9: noWorkflow SQLite database model after our implementation. The shadowed entities represent the tables added by our approach.

tation" table that makes it possible to store annotations in any format (e.g., plain text, JSON, XML, base64 images, etc) related to an experiment or a trial.

Table 4.1 displays a summary of our implementation on top of noWorkflow.

4.3 Approach Comparison

To compare our approach with existing collaborative provenance models and Script based systems, we use the provenance model classification (Table 4.2) proposed on Chapter 2 and the taxonomy proposed on the same chapter.

Regarding the provenance model classification, the classification of our noWorkflow extended version is illustrated on Table 4.2. In this Table, we repeat the models classified on Chapter 2 to easier the comparison. Regarding the provenance model, our approach is capable of capturing the data and interaction provenance naturally with its schema. It also opens up the possibility to register visualization, insight, and rationale provenance

Table 4.1: Summary of implementation on top of noWorkflow

Implementation	Description
Rest APIs	We have built Rest APIs that serves as the basis for our implementation. These APIs also open up a new possibility of integration with noWorkflow and the provenance storage, and can have diverse uses.
Push command	The push command identifies which data needs to be sent (either to the server in the central node approach or to another scientist in the peer-to-peer model) and sends the necessary data.
Pull command	The pull command identifies which data needs to be received (either from the server in the central node approach or from another scientist in the peer-to-peer model) and downloads the necessary data.
Multi experiment support	We have created the concept of "experiment" in the provenance storage, thus allowing to store data from several experiments in a single database and, from this, to compare these data and, for example, identify implicit collaborations.
Annotation support	It opens the possibility to the user to annotate the experiments or trials, being possible to include a textual description and even more structured complementary information. For now, there is a limitation that those annotations are not synchronized with the Pull and Push commands and the recommendation is that they should be made directly on the central node.

through its extended annotation table. Those extended properties could also be classified regarding its provenance type making it easier for the future use of these data. It is also naturally capable of supporting the distribution, multilevel, and collaboration aspects. Regarding the heterogeneity, the trial structure allows to easily store the evolution of the experiment, and the annotation capability allows scientists to register supplementary provenance data almost in any schema collected by other tools.

We also classify our approach as a tool on the collaboration taxonomy proposed on Chapter 2. On Table 4.3, we repeat the table presented on that chapter but now including our approach for comparison purposes. Regarding the experiment phases, our approach is able to address collaboration on the Composition and Analysis phases. In

terms of temporality, it uses an asynchronous approach, where each scientist works independently and chooses when to share information. It uses an Optimistic concurrency control approach, similar to the VisTrails [33] implementation explained on Chapter 2. Each execution of the experiment is treated as a trial. Trials are never modified or deleted (each modification generates a new trial). It also shares the limitation of VisTrails: if two scientists modify the same experiment before synchronizing it, this generates multiple disjoint versions, which can be problematic in terms of composition since the changes could be complementary. When this occurs, the scientist should "merge" the script manually. On the sharing aspect, it is possible to share data, models, and also knowledge through the annotation of the experiment or trials. Finally, on the provenance support, although the provenance model is able to capture all provenance types, the Visualization, Insight, and Rationale provenance needs to be registered manually by the users in the form of annotations.

In the next chapter, we present a case study that uses our extended version of noWorkflow to show its benefits for a group of scientists.

Table 4.2: Summary of the collaborative provenance models including our Extended noWorkflow version

Provenance Model	Provenance Types [71]	Aspects of Collaboration			
		D	H	M	C
Extended noWorkflow	All***	Yes	Yes**	Yes	Yes
Altintas <i>et al.</i> [1, 2]	Data; Interaction	No	No	No	Yes
CPM [97]	Data; Interaction	Yes	Evolution Only	Yes	Yes
Missier <i>et al.</i> [54]	Data; Interaction	Yes	Different schema only	No	Yes
Confucius [50, 86, 96, 99]	All**	Yes	No	Yes	Yes
ProvDB [52]	All**	Yes**	Yes**	Yes**	Yes

* (D) Distribution (H) Heterogeneity (M) Multilevel (C) Collaboration

**Modeled as extended properties

***Visualization, Insight, and Rationale provenance possible through extended properties

Table 4.3: Aspects of collaboration in the surveyed approaches including Extended noWorkflow

Approach	Aspects of collaboration				
	Experiment Phase	Temporality	Concurrency Control	Sharing	Provenance Support
Extended noWorkflow	Composition and Analysis	Asynchronous	Optimistic	Data, models and Knowledge	Data; Interaction
Confucius [50, 86, 96, 99]	Composition	Asynchronous; Real Time	Pessimistic	Data and models	Data; Interaction
myExperiment [35, 36, 23]	Composition and Analysis	Asynchronous	N/A	Data and models; Knowledge	Yes**
CAMERA [5, 81]	Composition and Analysis	Asynchronous	N/A	Data and models; Knowledge	Yes**
e-ScienceNet [16, 14, 15]	Composition	Asynchronous	N/A	Data and models; Knowledge	No
Collaborative PL-Science [62]	Composition and Analysis	Asynchronous	N/A	Data and models; Knowledge	No
Ellkvist <i>et al.</i> [29]	Composition	Real Time	Optimistic	Data and models	Data
VisTrails [33]	Composition	Asynchronous	Optimistic	N/A	Data
NoCoV [90]	Analysis	Asynchronous; Real Time	N/A	N/A	No
RASA [53]	Execution	Asynchronous	N/A	Physical resources	No
Wood, Wright, and Brodlie [94]	Analysis	Real Time	N/A	N/A	No
ViroLab [12]	Composition	Asynchronous	N/A	Data and models	Yes*
J. Zhang <i>et al.</i> [97]	Composition	Real Time	Pessimistic	Data and models	Data; Interaction
Mostaen <i>et al.</i> [58]	Composition	N/A	Pessimistic	N/A	No
ProvDB [52]	Composition	Asynchronous	Optimistic	Data and models	Data; Interaction
Dataverse [46]	Composition and Analysis	Asynchronous	N/A	Data and models	Yes**
OpenML [87]	Composition and Analysis	Asynchronous	N/A	Data and models	No
CoCalc [17]	Composition and Analysis	Asynchronous; Real Time	Optimistic	Data and models; Knowledge	Data; Interaction
Sumatra [21]	Analysis	Real Time	N/A	Data and models	Data

*No details are provided to correctly classify which provenance types are collected

**Stores data collected by other tools

Chapter 5

Case Study

To illustrate our approach, we build a simulated experiment that aims at (a) showing how the described approach can capture information and iterations not captured before and (b) how this data can be used to produce insights about the experiment.

In our case study, we consider a fictitious research institute called "Diseases Research Institute". This institute has several research groups working in several experiments and uses the central node collaboration architecture proposed on Chapter 4 to centralize the provenance data of those experiments. The experiments cited here are also fictitious, so our focus is on the interactions between the researchers and not on the actual experiment code or results.

With COVID-19 being a serious problem impacting the whole world, the Diseases Research Institute conducted multiple types of research on that subject. Among those experiments, one aims at predicting the numbers of infected people and deaths in a country over time. We call this experiment *COVID deaths prediction*. Despite being a fictitious experiment, methods of predicting deaths from COVID-19 are a real object of research [91, 40] and that is why we chose this example. *COVID deaths prediction* is conducted by several scientists, but since COVID implied in several lockdown practices, many of that scientists started to work remotely. Conducting this experiment collaboratively in a serialized manner with many scientists in different locations is nearly impossible, so much work is done in parallel. But since they are using our proposed approach, they could consolidate provenance information of the work done in a single provenance storage. This experiment has 5 scientists involved, and 28 trials summing up the trials of each scientist. The scientists worked in parallel and used the *push* and *pull* commands to send and retrieve information from the central node server. At the end of the process,

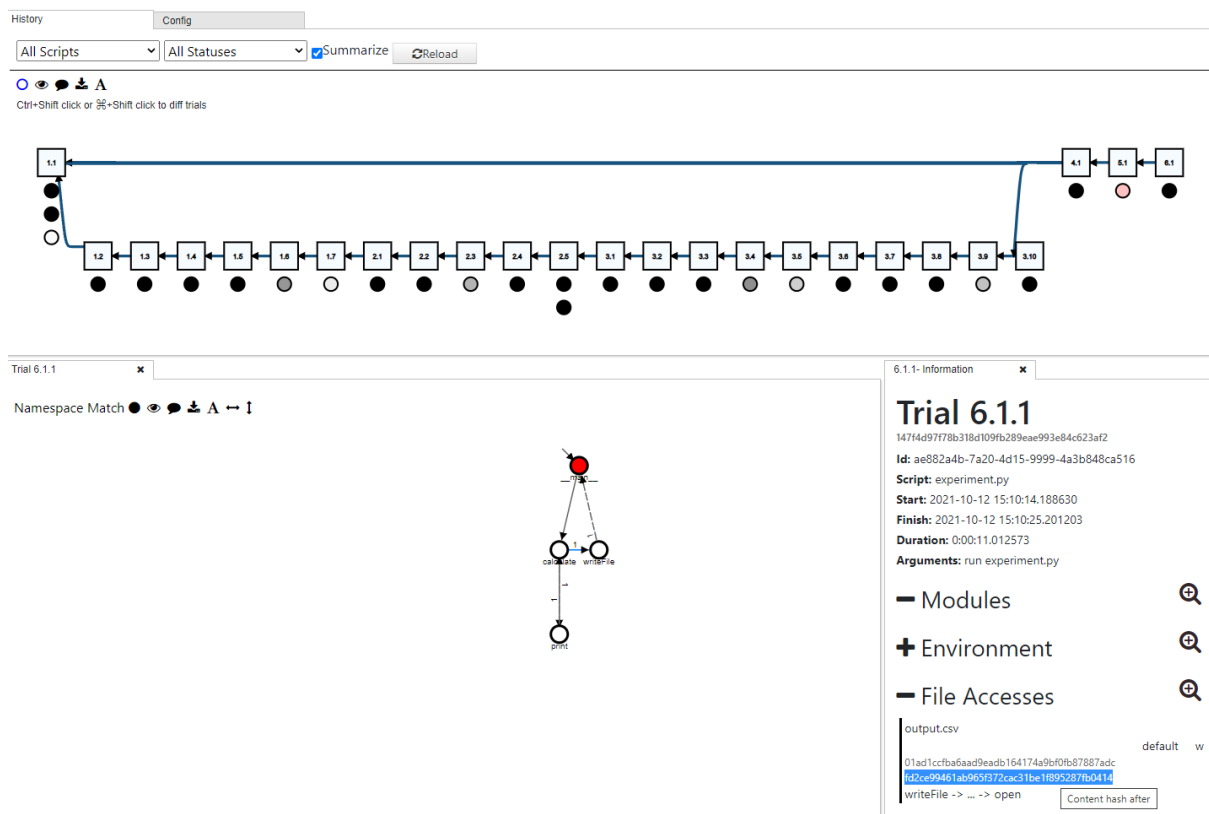
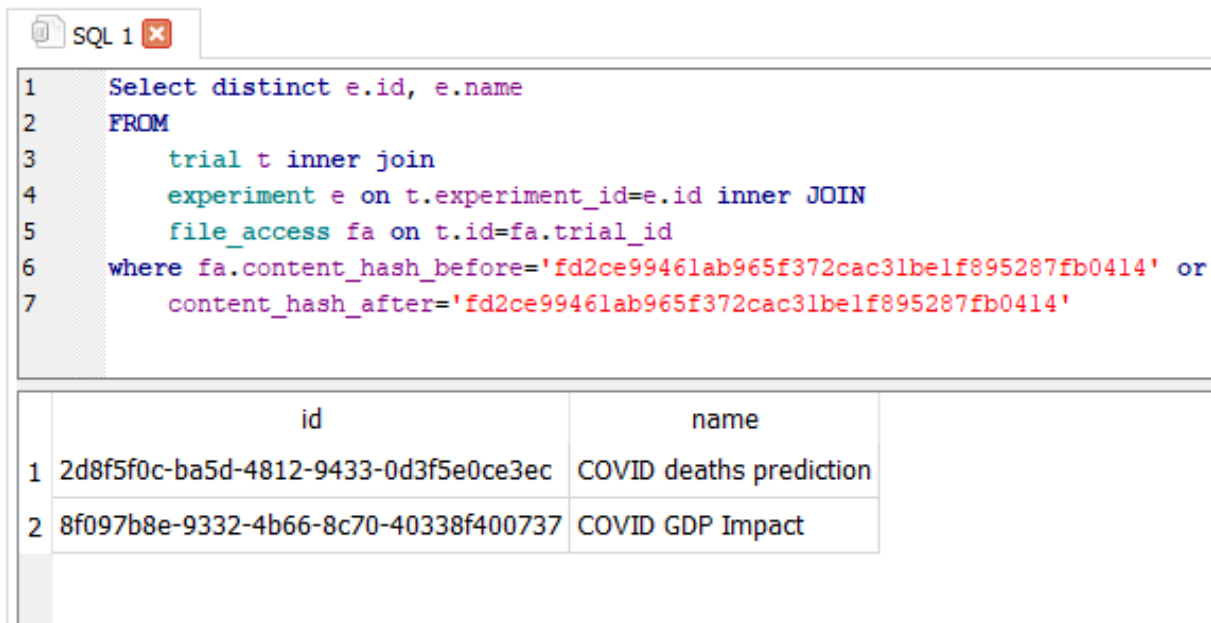


Figure 5.1: COVID deaths prediction provenance visualization

the provenance data of all of the trials are available at Diseases Research Institute’s provenance storage and could be queried not only by the researchers involved but also by anyone authorized by the Institute.

The results of the *COVID deaths prediction* experiment were published, and its provenance data was accessible to other researchers on the Institute, who used the results to conduct other experiments. However, the "Diseases Research Institute" ends up finding out that there is one implementation problem (bug) in the script code used by *COVID deaths prediction*. First of all, to avoid that the wrong data could be used in other experiments, the scientists added annotations both at the experiment and at the problematic trial reporting the problem and warning users of the problem. Now the Institute needs to discover all other experiments that were already built on top of *COVID deaths prediction* and could have their results contaminated by the *COVID deaths prediction* experiment problems.

Since researchers use our noWorkflow collaborative extension to run their experiments, the Diseases Research Institute has a centralized database containing interactions inside an experiment and the implicit collaborations between the experiments. To identify which experiments are impacted by the *COVID deaths prediction*, the user entered the



```

1  Select distinct e.id, e.name
2  FROM
3      trial t inner join
4      experiment e on t.experiment_id=e.id inner JOIN
5      file_access fa on t.id=fa.trial_id
6  where fa.content_hash_before='fd2ce99461ab965f372cac31be1f895287fb0414' or
7      content_hash_after='fd2ce99461ab965f372cac31be1f895287fb0414'

```

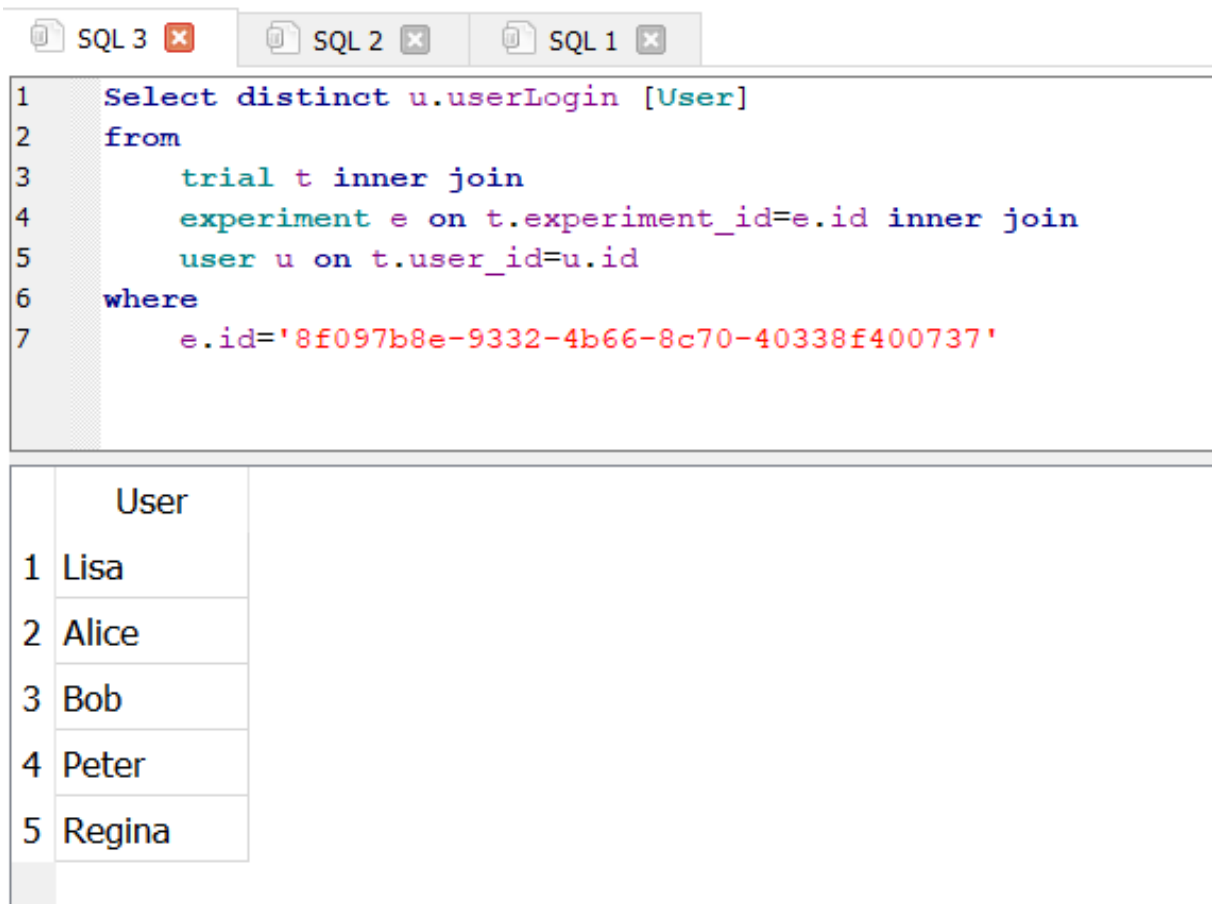
	id	name
1	2d8f5f0c-ba5d-4812-9433-0d3f5e0ce3ec	COVID deaths prediction
2	8f097b8e-9332-4b66-8c70-40338f400737	COVID GDP Impact

Figure 5.2: SQL query used to find experiments using a specific file

experiment portal (illustrated in Figure 4.3), navigated to *COVID deaths prediction* (first line on the experiment list shown in Figure 4.3), and inspected the outputs of its last trial (trial 6.1.1 on Figure 5.1). They find out that the experiment generates a file called "output.csv" with hash code *fd2ce99461ab965f372cac31be1f895287fb0414*. Then, the user used the SQL query illustrated in Figure 5.2 to query the provenance database and find out which experiments used its output.

The Diseases Research Institute ends up finding out that the *COVID GDP Impact* experiment was impacted by *COVID deaths prediction* results. The *COVID GDP Impact* is an experiment that aims to predict the GDP impact over time caused by COVID-19 on a country. The researchers of the *COVID GDP Impact* experiment must be notified, so the Institute uses the query illustrated in Figure 5.3 to find people involved in the experiment and notify them. Since the GDP experiment was not publicly published yet, they could use the new results of *COVID deaths prediction* and correct their experiment.

Another issue is that the Diseases Research Institute has a strict policy for credit attribution, thus demanding that even implicit collaborators should receive credit and be properly registered. To comply with that, the researchers responsible for the *COVID GDP Impact* experiment used the query illustrated in Figure 5.4 to find experiments that implicit collaborates with it. The query identifies that the *COVID deaths prediction* experiment is related to the *COVID GDP Impact*, as expected, but also that the the *GDP Impact of pandemics* experiment is related. With the experiment ids, they can use the query in Figure 5.3, just adjusting the experiment id parameter, to identify collaborators



```

1  Select distinct u.userLogin [User]
2  from
3      trial t inner join
4      experiment e on t.experiment_id=e.id inner join
5      user u on t.user_id=u.id
6  where
7      e.id='8f097b8e-9332-4b66-8c70-40338f400737'

```

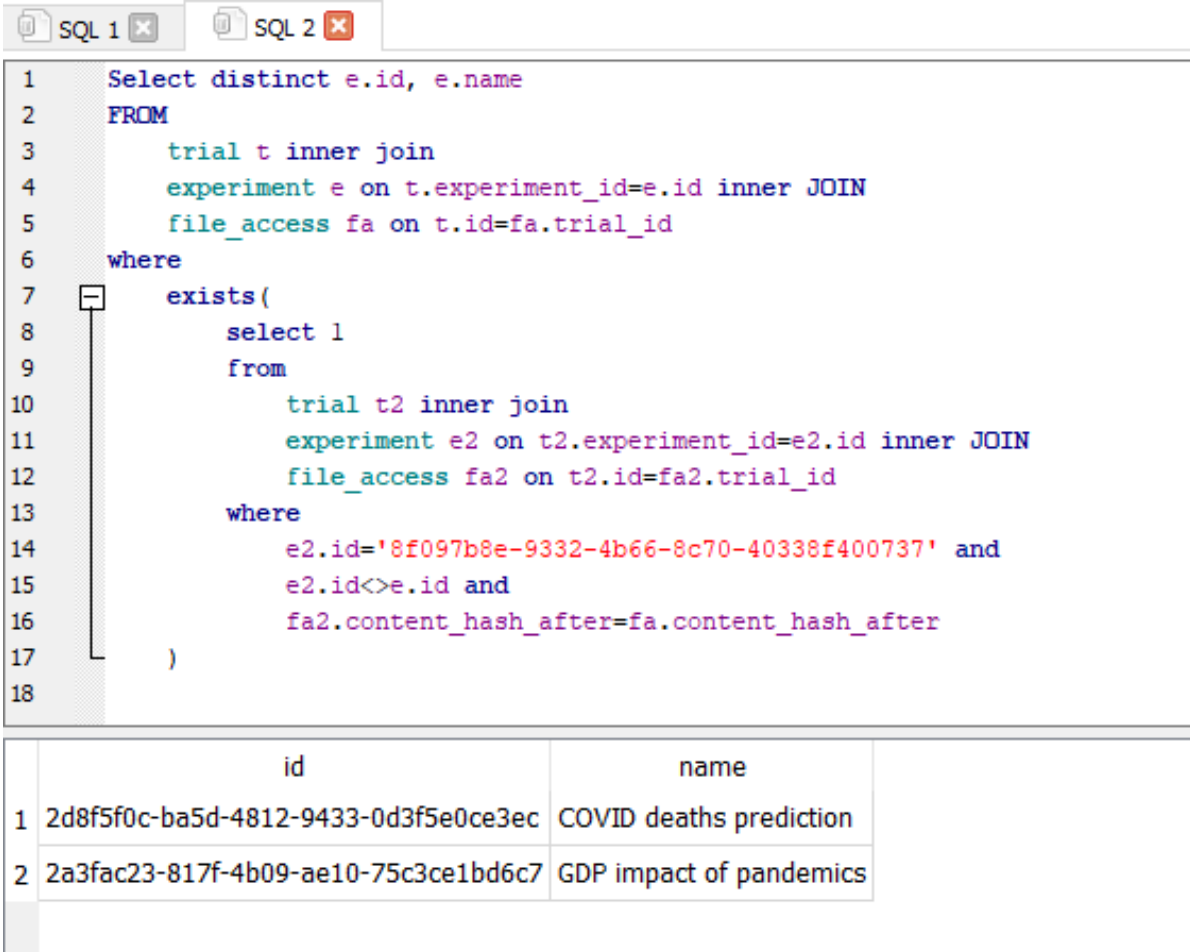
	User
1	Lisa
2	Alice
3	Bob
4	Peter
5	Regina

Figure 5.3: SQL query used to find scientists involved in experiments.

and give them the proper credit.

Using the noWorkflow standard version as a baseline, the first challenge will be even to conduct the *COVID deaths prediction* experiment with 5 scientists. Since noWorkflow standard version does not address collaboration and provenance consolidation between researchers, they will have to have the discipline to work synchronously, creating a manual "lock" policy between them and controlling who has the right to work in the experiment at a given time. They will also have to send the noWorkflow provenance database to the next one working on the experiment when they finish what they are doing. This could be done by e-mail for example. Notice that all these controls will have to be made by the scientists themselves without any support from the system.

The second challenge will be, after the sharing of the experiment data in the Diseases Research Institute, to find which other experiments could be contaminated by the bug on the *COVID deaths prediction* experiment. This could be done by broadcasting a message to the researchers on the institute communicating the flaw and that the published data is invalid. However, if every time such an event occurs in the Institute a broadcast



```

1  Select distinct e.id, e.name
2  FROM
3      trial t inner join
4      experiment e on t.experiment_id=e.id inner JOIN
5      file_access fa on t.id=fa.trial_id
6  where
7      exists(
8          select 1
9          from
10             trial t2 inner join
11             experiment e2 on t2.experiment_id=e2.id inner JOIN
12             file_access fa2 on t2.id=fa2.trial_id
13         where
14             e2.id='8f097b8e-9332-4b66-8c70-40338f400737' and
15             e2.id<>e.id and
16             fa2.content_hash_after=fa.content_hash_after
17     )
18

```

	id	name
1	2d8f5f0c-ba5d-4812-9433-0d3f5e0ce3ec	COVID deaths prediction
2	2a3fac23-817f-4b09-ae10-75c3ce1bd6c7	GDP impact of pandemics

Figure 5.4: SQL query used to find implicit collaborations

communication is sent, people will probably start to ignore the messages, treating them as spam. And assuming that a scientist works, on average, in 5 experiments simultaneously, even if a scientist takes a look at the communication and decides to take action on it to see if she was impacted, each scientist would have to query 5 provenance databases to get the answer.

The third challenge will be to discover implicit collaboration with other experiments in order to give the correct credit attribution. In that case, the user will have to list all the files used by the experiment and also contact all scientists in the research group to query their databases and see which trials generated those files. This process will become so complex and laborious that may cause scientists to trust their memory only, with a huge possibility of missing someone, which would lead to incomplete information.

All this process is very error-prone and also will demand a lot of extra work, besides introducing a lot of limits on how scientists conduct their work, definitely impacting their productivity. Compared to our suggested approach, it is clear that our approach will

have a huge benefit on researchers' productivity, saving a lot of time for scientists and also bringing much more reliable provenance data.

Chapter 6

Conclusion

Collaboration is part of science and scientific research. Although this has been a fact since centuries ago, we can see that it is still a growing movement nowadays. One of the catalysts of this has been the advances in computer technology and networks – the same advances that helped to enable and popularize *in silico* experiments. Although this is very positive for science, it brings several challenges. To better understand the challenges and evaluate the literature on the subject, we made a survey on collaboration in *in silico* scientific research. In this survey, we map the provenance models and available tools to identify the state-of-art of research on collaborative experiments conducted *in silico*. We also propose a taxonomy and use it to classify the existing tools and discuss opportunities based on the gaps we identified. This survey was published in SIGMOD RECORD [44].

This survey provided information on relevant gaps that we could work on. One of our particular interest was the lack of support to collaboration on Script-based systems. To validate the relevance of this gap we decided to do a questionnaire with the scientific community and analyze if the scientists on their daily routines face difficulties dealing with it. The questionnaire results corroborate the gap identified by the survey so we decided to work on a collaborative approach to *in silico* experiments based on scripts.

Aiming to take advantage of the progress already done in Script-based systems, we implemented our approach on top of noWorkflow. However, the approach itself is generic and could also be implemented in any other Script-based system. We also classify both our collaboration-aware provenance model and our tool based on the taxonomy proposed in Chapter 2

In addition, we conducted a case study using a hypothetical but very feasible situation. Our case study confirmed our hypothesis that the approach will make it easier to

collaborate and that a single provenance database for multiple experiments could help scientists and research institutes in several ways. The main benefits are: to easily allow collaboration in this kind of experiment; to have a single provenance storage that stores information about all experiments and all trials of a single experiment, even those run by different scientists; to find the hidden collaboration between scientists and experiments; and more reproducible and auditable experiments.

Although we argue that our proposal provides significant improvements in the current scenario, we understand there are some limitations. First, our questionnaire has a limited number of respondents, although it corroborates claims of related literature [44, 95, 41]. Second, *in silico* experiments can produce many data, and transferring these data over the network and even storing data in a distributed manner could be a problem in some big data scenarios. Third, the implementation of access control management is still pending. Fourth, although the possibility of annotating trials and experiments opens new possibilities, we are using it as a wildcard to store some provenance types – the automatic capture of visualization, insight, and rationale provenance should be studied and addressed. Fifth, our approach does not consider the heterogeneity of the research group and focuses more on scientists with good computer science skills – this could be a problem in some research groups. Finally, we use a fictitious case study – although it illustrates a very feasible scenario, we understand that a real collaboration scenario could be very helpful in identifying additional improvement possibilities for our approach.

As future work, we envision conducting a broad questionnaire with more scientists and communities that could lead to other insights. We also plan to propose better file storage and data transfer strategies and implement access control management features since many pieces of research could involve secrecy. Another possibility we want to explore is to integrate our approach with distributed provenance collection tools, covering experiments that run on a distributed infrastructure and are conducted by multiple scientists at the same time.

We also plan to work on improving the awareness of scientists involved in the experiment/research group. The awareness of these scientists could be improved through a more proactive behavior, providing information about other scientists' trials or experiments instead of being passive, and just pulling the information when requested or giving the collaborative information when queried. Finally, we plan to conduct a validation experiment with a real use case scenario.

References

- [1] ALTINTAS, I.; ANAND, M. K.; CRAWL, D.; BOWERS, S.; BELLOUM, A.; MISSIER, P.; LUDÄSCHER, B.; GOBLE, C. A.; SLOOT, P. M. A. Understanding collaborative studies through interoperable workflow provenance. In *Provenance and Annotation of Data and Processes* (2010), D. L. McGuinness, J. R. Michaelis, and L. Moreau, Eds., Springer Berlin Heidelberg, pp. 42–58.
- [2] ALTINTAS, I.; ANAND, M. K.; VUONG, T. N.; BOWERS, S.; LUDÄSCHER, B.; SLOOT, P. M. A. A data model for analyzing user collaborations in workflow-driven e-science. *International Journal of Computers and Their Applications* 18 (2011), 160–179.
- [3] ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; JONES, M.; LUDASCHER, B.; MOCK, S. Kepler: an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management* (2004), pp. 423–424.
- [4] ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; JONES, M.; LUDASCHER, B.; MOCK, S. Kepler: an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management* (2004), pp. 423–424.
- [5] ALTINTAS, I.; LIN, A. W.; CHEN, J.; CHURAS, C.; GUJRAL, M.; SUN, S.; LI, W.; MANANSALA, R.; SEDOVA, M.; GRETHE, J. S.; ELLISMAN, M. Camera 2.0: A data-centric metagenomics community infrastructure driven by scientific workflows. In *World Congress on Services* (2010), pp. 352–359.
- [6] AMSTUTZ, P.; CRUSOE, M. R.; TIJANIĆ, N.; CHAPMAN, B.; CHILTON, J.; HEUER, M.; KARTASHOV, A.; LEEHR, D.; MÉNAGER, H.; NEDELJKOVICH, M., ET AL. Common workflow language, v1. 0.
- [7] ANAND, M. K.; BOWERS, S.; MCPHILLIPS, T.; LUDÄSCHER, B. Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. In *Scientific and Statistical Database Management*, M. Winslett, Ed., Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 237–254.
- [8] BASSETT, L. *Introduction to JavaScript object notation: a to-the-point guide to JSON*. " O'Reilly Media, Inc.", 2015.
- [9] BELLOUM, A.; INDA, M. A.; VASUNIN, D.; KORKHOV, V.; ZHAO, Z.; RAUWERDA, H.; BREIT, T. M.; BUBAK, M.; HERTZBERGER, L. O. Collaborative e-science experiments and scientific workflows. *IEEE Internet Computing* 15, 4 (July 2011), 39–47.
- [10] BEN-KIKI, O.; EVANS, C.; INGERSON, B. Yaml ain't markup language (yaml™) version 1.1. *Working Draft 2008-05 11* (2009).

-
- [11] BENNIS, W. G.; BIEDERMAN, P. W. *Organizing genius: The secrets of creative collaboration*. Basic Books, 2007.
- [12] BUBAK, M.; GUBALA, T.; KASZTELNIK, M.; MALAWSKI, M.; NOWAKOWSKI, P.; SLOOT, P. Collaborative virtual laboratory for e-health. In *Expanding the Knowledge Economy: Issues, Applications, Case Studies, eChallenges* (2007), pp. 537–544.
- [13] CALDWELL, R.; LINDBERG, D. Participants in science behave scientifically. *Understanding Science*. (2018). Available at https://undsci.berkeley.edu/article/0_0_0/whatisscience_09.
- [14] CLASSE, T.; BRAGA, R.; CAMPOS, F.; DAVID, J. M. N. A semantic peer to peer network to support e-science. In *IEEE International Conference on e-Science* (2015), pp. 503–512.
- [15] CLASSE, T.; BRAGA, R.; DAVID, J. M. N.; CAMPOS, F.; ARAÚJO, M. A.; STRÖELE, V. A collaborative approach to support e-science activities. In *IEEE International Conference on Computer Supported Cooperative Work in Design* (2016), IEEE, pp. 20–25.
- [16] CLASSE, T.; BRAGA, R.; DAVID, J. M. N.; CAMPOS, F.; ARBEX, W. A distributed infrastructure to support scientific experiments. *Journal of Grid Computing* 15, 4 (2017), 475–500.
- [17] Cocalc user manual documentation. <https://doc.cocalc.com/contents.html>, 2013. Accessed: 2019-12-05.
- [18] COSTA, G. C. B.; BRAGA, R.; DAVID, J. M. N.; CAMPOS, F. A scientific software product line for the bioinformatics domain. *Journal of Biomedical Informatics* 56 (2015), 239–264.
- [19] DATE, C. J. *An introduction to database systems*. Pearson/Addison Wesley, Boston, 2004.
- [20] DAVIDSON, S. B.; FREIRE, J. Provenance and scientific workflows: Challenges and opportunities. In *ACM Special Interest Group on Management of Data* (2008), ACM, pp. 1345–1350.
- [21] DAVISON, A. Automated capture of experiment context for easier reproducibility in computational research. *Computing in Science & Engineering* 14, 4 (2012), 48–56.
- [22] DE OLIVEIRA, D.; OGASAWARA, E.; BAIÃO, F.; MATTOSO, M. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *International Conference on Cloud Computing* (Washington, DC, USA, 2010), pp. 378–385.
- [23] DE ROURE, D.; GOBLE, C.; STEVENS, R. The design and realisation of the myexperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems* 25, 5 (2009), 561–567.
- [24] DE ROURE, D.; GOBLE, C.; STEVENS, R. The design and realisation of the myexperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems* 25, 5 (2009), 561–567.

- [25] DEELMAN, E.; SINGH, G.; SU, M.-H.; BLYTHE, J.; GIL, Y.; KESSELMAN, C.; MEHTA, G.; VAHI, K.; BERRIMAN, G. B.; GOOD, J.; LAITY, A.; JACOB, J. C.; KATZ, D. S. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal* 13, 3 (2005), 219–237.
- [26] DEUTSCH, P. Rfc1952: Gzip file format specification version 4.3, 1996.
- [27] DI TOMMASO, P.; CHATZOU, M.; FLODEN, E. W.; BARJA, P. P.; PALUMBO, E.; NOTREDAME, C. Nextflow enables reproducible computational workflows. *Nature biotechnology* 35, 4 (2017), 316–319.
- [28] DUCE, D. A.; SAGAR, M. S. skML a markup language for distributed collaborative visualization. In *Theory and Practice of Computer Graphics* (2005), pp. 171–178.
- [29] ELLKVIST, T.; KOOP, D.; ANDERSON, E. W.; FREIRE, J.; SILVA, C. Using provenance to support real-time collaborative design of workflows. In *International Workshop on Provenance and Annotation (IPAW)*. Springer, 2008, pp. 266–279.
- [30] ELMASRI, R.; NAVATHE, S. *Fundamentals of database systems*, 6 ed. Addison-Wesley, Apr. 2010.
- [31] FOULSER, D. IRIS Explorer: a framework for investigation. *SIGGRAPH Computer Graphics* 29, 2 (1995), 13–16.
- [32] FREIRE, J.; KOOP, D.; SANTOS, E.; SILVA, C. T. Provenance for computational tasks: A survey. *Computing in Science & Engineering* 10, 3 (2008), 11–21.
- [33] FREIRE, J.; SILVA, C. T.; CALLAHAN, S. P.; SANTOS, E.; SCHEIDEGGER, C. E.; VO, H. T. Managing rapidly-evolving scientific workflows. In *Provenance and Annotation of Data*, L. Moreau and I. Foster, Eds., Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 10–18.
- [34] GIL, Y.; DEELMAN, E.; ELLISMAN, M.; FAHRINGER, T.; FOX, G.; GANNON, D.; GOBLE, C.; LIVNY, M.; MOREAU, L.; MYERS, J. Examining the challenges of scientific workflows. *Computer* 40, 12 (2007), 24–32.
- [35] GOBLE, C. A.; BHAGAT, J.; ALEKSEJEVS, S.; CRUICKSHANK, D.; MICHAELIDES, D.; NEWMAN, D.; BORKUM, M.; BECHHOFFER, S.; ROOS, M.; LI, P.; DE ROURE, D. myexperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research* 38, Web Server Issue (2010), 677–682.
- [36] GOBLE, C. A.; ROURE, D. C. D. myexperiment: Social networking for workflow-using e-scientists. In *Workshop on Workflows in Support of Large-Scale Science* (2007), ACM, pp. 1–2.
- [37] GOODMAN, L. A. Snowball sampling. *The Annals of Mathematical Statistics* 32, 1 (1961), 148–170.
- [38] GOODSTADT, L. Ruffus: a lightweight python library for computational pipelines. *Bioinformatics* 26, 21 (2010), 2778–2779.
- [39] GRINBERG, M. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

- [40] HAMZAH, F. B.; LAU, C.; NAZRI, H.; LIGOT, D. V.; LEE, G.; TAN, C. L.; SHAIB, M.; ZAIDON, U. H. B.; ABDULLAH, A.; CHUNG, M. H., ET AL. Coronatracker: worldwide covid-19 outbreak data analysis and prediction. *Bull World Health Organ* 1, 32 (2020), 1–32.
- [41] HERSCHEL, M.; DIESTELKÄMPER, R.; LAHMAR, H. B. A survey on provenance: What for? what form? what from? *The VLDB Journal* 26, 6 (2017), 881–906.
- [42] HULL, D.; WOLSTENCROFT, K.; STEVENS, R.; GOBLE, C.; POCOCK, M. R.; LI, P.; OINN, T. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 34, 2 (2006), 729–732.
- [43] III, H. M. R.; HONEMANN, D. H.; BALCH, T. J.; SEABOLD, D. E.; GERBER, S. *Robert’s rules of order newly revised*, 11 ed. PublicAffairs, 2011.
- [44] JANDRE, E.; DIIRR, B.; BRAGANHOLO, V. Provenance in collaborative in silico scientific research: a survey. *ACM SIGMOD Record* 49, 2 (2020), 36–51.
- [45] JIA ZHANG; CHANG, C.; JEN-YAO CHUNG. Mediating electronic meetings. In *International Computer Software and Applications Conference* (2003), pp. 216–221.
- [46] KING, G. An introduction to the dataverse network as an infrastructure for data sharing, 2007.
- [47] LEACH, P.; MEALLING, M.; SALZ, R. Rfc 4122: A universally unique identifier (uuid) urn namespace, 2005.
- [48] LERNER, B.; BOOSE, E. Rdatatracker: collecting provenance in an interactive scripting environment. In *USENIX Workshop on the Theory and Practice of Provenance (TaPP)* (2014).
- [49] LU, S.; ZHANG, J. Collaborative scientific workflows. In *IEEE International Conference on Web Services* (2009), IEEE, pp. 527–534.
- [50] LU, S.; ZHANG, J. Collaborative scientific workflows supporting collaborative science. *International Journal of Business Process Integration and Management* (2011), 185.
- [51] MATTOSO, M.; WERNER, C.; TRAVASSOS, G. H.; BRAGANHOLO, V.; OGASAWARA, E.; OLIVEIRA, D.; CRUZ, S.; MARTINHO, W.; MURTA, L. Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management* 5, 1 (2010), 79–92.
- [52] MIAO, H.; CHAVAN, A.; DESHPANDE, A. ProvdB: Lifecycle management of collaborative analysis workflows. In *Workshop on Human-In-the-Loop Data Analytics (HILDA)* (New York, NY, USA, 2017), ACM, pp. 7:1–7:6.
- [53] MILLER, T.; MCBURNEY, P.; MCGINNIS, J.; STATHIS, K. First-class protocols for agent-based coordination of scientific instruments. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (2007), pp. 41–46.

- [54] MISSIER, P.; LUDASCHER, B.; BOWERS, S.; DEY, S.; SARKAR, A.; SHRESTHA, B.; ALTINTAS, I.; ANAND, M.; GOBLE, C. Linking multiple workflow provenance traces for interoperable collaborative science. In *Workshop on Workflows in Support of Large-Scale Science* (2010), pp. 1–8.
- [55] MÖLDER, F.; JABLONSKI, K. P.; LETCHER, B.; HALL, M. B.; TOMKINS-TINCH, C. H.; SOCHAT, V.; FORSTER, J.; LEE, S.; TWARDZIOK, S. O.; KANITZ, A., ET AL. Sustainable data analysis with snakemake. *F1000Research* 10 (2021).
- [56] MOREAU, L.; CLIFFORD, B.; FREIRE, J.; FUTRELLE, J.; GIL, Y.; GROTH, P.; KWASNIKOWSKA, N.; MILES, S.; MISSIER, P.; MYERS, J.; PLALE, B.; SIMMHAN, Y.; STEPHAN, E.; DEN BUSSCHE, J. V. The open provenance model core specification (v1.1). *Future Generation Computer Systems* 27, 6 (2011), 743–756.
- [57] MOREAU, L.; MISSIER, P.; BELHAJJAME, K.; B'FAR, R.; CHENEY, J.; COPPENS, S.; CRESSWELL, S.; GIL, Y.; GROTH, P.; KLYNE, G.; LEBO, T.; MCCUSKER, J.; MILES, S.; MYERS, J.; SAHOO, S.; TILMES, C. PROV-DM: The PROV data model. W3C Recommendation. *W3C Recommendation* (2013). Available at <http://www.w3.org/TR/2013/REC-prov-dm-20130430/>.
- [58] MOSTAEEN, G.; ROY, B.; ROY, C. K.; SCHNEIDER, K. A. Fine-grained attribute level locking scheme for collaborative scientific workflow development. In *IEEE International Conference on Services Computing* (2018), pp. 273–277.
- [59] MURTA, L.; BRAGANHOLO, V.; CHIRIGATI, F.; KOOP, D.; FREIRE, J. noWorkflow: Capturing and Analyzing Provenance of Scripts. In *International Workshop on Provenance Annotation (IPAW)* (2014), Springer International Publishing, pp. 1–12.
- [60] MYERS, J.; COPELAND, R. *Essential SQLAlchemy: Mapping Python to Databases*. " O'Reilly Media, Inc.", 2015.
- [61] O'SULLIVAN, B. *Mercurial: The Definitive Guide*. " O'Reilly Media, Inc.", 2009.
- [62] PEREIRA, A. F.; DAVID, J. M. N.; BRAGA, R.; CAMPOS, F. An architecture to enhance collaboration in scientific software product line. In *International Conference on System Sciences* (2016), IEEE, pp. 338–347.
- [63] PIMENTEL, J. F.; DEY, S.; MCPHILLIPS, T.; BELHAJJAME, K.; KOOP, D.; MURTA, L.; BRAGANHOLO, V.; LUDÄSCHER, B. Yin & yang: Demonstrating complementary provenance from noworkflow & yesworkflow. In *Provenance and Annotation of Data and Processes* (Cham, 2016), M. Mattoso and B. Glavic, Eds., Springer International Publishing, pp. 161–165.
- [64] PIMENTEL, J. F.; FREIRE, J.; BRAGANHOLO, V.; MURTA, L. Tracking and analyzing the evolution of provenance from scripts. In *Provenance and Annotation of Data and Processes* (Cham, 2016), M. Mattoso and B. Glavic, Eds., Springer International Publishing, pp. 16–28.
- [65] PIMENTEL, J. F.; FREIRE, J.; MURTA, L.; BRAGANHOLO, V. Fine-grained provenance collection over scripts through program slicing. In *Provenance and Annotation of Data and Processes* (Cham, 2016), M. Mattoso and B. Glavic, Eds., Springer International Publishing, pp. 199–203.

- [66] PIMENTEL, J. F.; FREIRE, J.; MURTA, L.; BRAGANHOLO, V. A survey on collecting, managing, and analyzing provenance from scripts. *ACM Computing Surveys* 52, 3 (2019), 47:1–47:38.
- [67] PIMENTEL, J. F.; FREIRE, J.; MURTA, L.; BRAGANHOLO, V. A survey on collecting, managing, and analyzing provenance from scripts. *ACM Computing Surveys* 52, 3 (2019), 47:1–47:38.
- [68] PIMENTEL, J. F.; MURTA, L.; BRAGANHOLO, V.; FREIRE, J. noworkflow: a tool for collecting, analyzing, and managing provenance from python scripts. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1841–1844.
- [69] PIMENTEL, J. F. N.; BRAGANHOLO, V.; MURTA, L.; FREIRE, J. Collecting and analyzing provenance on interactive notebooks: When ipython meets noworkflow. In *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15)* (Edinburgh, Scotland, July 2015), USENIX Association.
- [70] PRUDÊNCIO, J.; MURTA, L.; WERNER, C.; CEPÊDA, R. To lock, or not to lock: That is the question. *Journal of Systems and Software* 85, 2 (2012), 277–289.
- [71] RAGAN, E. D.; ENDERT, A.; SANYAL, J.; CHEN, J. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 31–40.
- [72] RAMAKRISHNAN, R.; GEHRKE, J. *Database management systems*, third edition ed. McGraw-Hill, New York, 2003.
- [73] REDDY, M. C.; DOURISH, P.; PRATT, W. Temporality in medical work: Time also matters. *Computer Supported Cooperative Work* 15, 1 (2006), 29–53.
- [74] RICHARDSON, L.; AMUNDSEN, M.; AMUNDSEN, M.; RUBY, S. *RESTful Web APIs: Services for a Changing World*. " O'Reilly Media, Inc.", 2013.
- [75] SAWYER, K. *Group genius: The creative power of collaboration*. Basic books, 2017.
- [76] SONNENWALD, D. H. Scientific collaboration. *Annual review of information science and technology* 41, 1 (2007), 643–681.
- [77] SONNENWALD, D. H. Scientific collaboration. *Annual review of information science and technology* 41, 1 (2007), 643–681.
- [78] SPINELLIS, D. Git. *IEEE software* 29, 3 (2012), 100–101.
- [79] Apache subversion. <https://subversion.apache.org/>. Accessed: 2019-04-23.
- [80] Sumatra 0.7.0 documentation. https://pythonhosted.org/Sumatra/record_stores.html. Accessed: 2019-12-03.
- [81] SUN, S.; CHEN, J.; LI, W.; ALTINTAS, I.; LIN, A.; PELTIER, S.; STOCKS, K.; ALLEN, E. E.; ELLISMAN, M.; GRETHE, J.; WOOLEY, J. Community cyberinfrastructure for advanced microbial ecology research and analysis: the CAMERA resource. *Nucleic Acids Research* 39 (2011), D546–551.

- [82] TANENBAUM, A. S. *Modern operating systems*, 3 edition ed. Prentice Hall, Upper Saddle River, N.J, Dec. 2007.
- [83] Gt4 globus toolkit web site. <http://toolkit.globus.org/toolkit/>. Accessed: 2019-04-23.
- [84] TRAVASSOS, G. H.; BARROS, M. O. Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering. In *Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering* (2003), pp. 117–130.
- [85] TRAVASSOS, G. H.; BARROS, M. O. Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering. In *Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering* (2003), pp. 117–130.
- [86] VALI, S.; SREERAMA, S. Multi-user tool for scientific work flow composition. *International Journal of Computer Trends & Technology* 4 (2013).
- [87] VAN RIJN, J. N.; BISCHL, B.; TORGO, L.; GAO, B.; UMAASHANKAR, V.; FISCHER, S.; WINTER, P.; WISWEDEL, B.; BERTHOLD, M. R.; VANSCHOREN, J. Openml: A collaborative science platform. In *Joint european conference on machine learning and knowledge discovery in databases* (2013), Springer, pp. 645–649.
- [88] VIVIAN, J.; RAO, A. A.; NOTHAFT, F. A.; KETCHUM, C.; ARMSTRONG, J.; NOVAK, A.; PFEIL, J.; NARKIZIAN, J.; DERAN, A. D.; MUSSELMAN-BROWN, A., ET AL. Toil enables reproducible, open source, big biomedical data analyses. *Nature biotechnology* 35, 4 (2017), 314–316.
- [89] VOSS, K.; VAN DER AUWERA, G.; GENTRY, J. Full-stack genomics pipelining with gatk4+ wdl+ cromwell. *F1000Research* 6 (2017).
- [90] WANG, H.; BRODLIE, K. W.; HANDLEY, J. W.; WOOD, J. D. Service-oriented approach to collaborative visualization. *Concurrency and Computation: Practice and Experience* 20, 11 (2008), 1289–1301.
- [91] WEINBERGER, D. M.; CHEN, J.; COHEN, T.; CRAWFORD, F. W.; MOSTASHARI, F.; OLSON, D.; PITZER, V. E.; REICH, N. G.; RUSSI, M.; SIMONSEN, L., ET AL. Estimation of excess deaths associated with the covid-19 pandemic in the united states, march to may 2020. *JAMA Internal Medicine* 180, 10 (2020), 1336–1344.
- [92] WILDE, M.; FOSTER, I.; ISKRA, K.; BECKMAN, P.; ZHANG, Z.; ESPINOSA, A.; HATEGAN, M.; CLIFFORD, B.; RAICU, I. Parallel scripting for applications at the petascale and beyond. *Computer* 42, 11 (2009), 50–60.
- [93] WILKINSON, M. D.; DUMONTIER, M.; AALBERSBERG, I. J.; APPLETON, G.; AXTON, M.; BAAK, A.; BLOMBERG, N.; BOITEN, J.-W.; DA SILVA SANTOS, L. B.; BOURNE, P. E., ET AL. The fair guiding principles for scientific data management and stewardship. *Scientific data* 3, 1 (2016), 1–9.
- [94] WOOD, J.; WRIGHT, H.; BRODLIE, K. Collaborative visualization. In *Conference on Visualization* (1997), IEEE Computer Society Press, pp. 253–259.

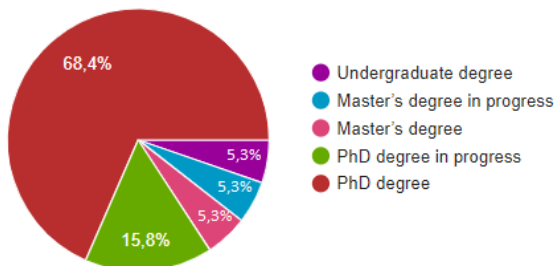
-
- [95] WUCHTY, S.; JONES, B. F.; UZZI, B. The increasing dominance of teams in production of knowledge. *Science* 316, 5827 (2007), 1036–1039.
- [96] ZHANG, J. Co-taverna: A tool supporting collaborative scientific workflows. In *IEEE International Conference on Services Computing* (2010), pp. 41–48.
- [97] ZHANG, J.; BAO, Q.; DUAN, X.; LU, S.; XUE, L.; SHI, R.; TANG, P. Collaborative scientific workflow composition as a service: An infrastructure supporting collaborative data analytics workflow design and management. In *IEEE International Conference on Collaboration and Internet Computing* (2016), pp. 219–228.
- [98] ZHANG, J.; CHANG, C. K.; VOAS, J. A uniform meta-model for mediating formal electronic conferences. In *International Computer Software and Applications Conference* (2004), IEEE, pp. 376–381.
- [99] ZHANG, J.; KUC, D.; LU, S. Confucius: A tool supporting collaborative scientific workflow composition. *IEEE Transactions on Services Computing* 7, 1 (2012).

APPENDIX A - Questionnaire

To better understand the importance of collaboration on the day-by-day of scientists and how the existing approaches could be improved, we conducted a questionnaire with members of the scientific community. The questionnaire was sent to the following institutions: LNCC, Fiocruz, USP, CENPES, New York University (NYU), University of Pennsylvania (UPenn), University of Amsterdam (UVA), Newcastle University, SouthHampton University. It was also sent to researchers of the DATAONE Project. The survey was conducted 100% online, and all responses were anonymous. We end up having 19 respondents, which give us an initial idea of their reality and the problems they face. The questionnaire has a total of 15 questions: 6 closed-questions and 9 discursive questions, some questions were not mandatory and could be left blank by the responder. Table A.1 and Table A.2 present all the questions and the obtained responses.

The questionnaire was split into two sections: Participant Characterization and Previous Experience (6 questions), and Collaboration Experience (9 questions). On Section A.1 and Section A.2 of this appendix we take a deeper on each questionnaire section and look at the insights that those questions bring.

What is your educational level?



How many scientific experiments have you ever performed on computational environments?

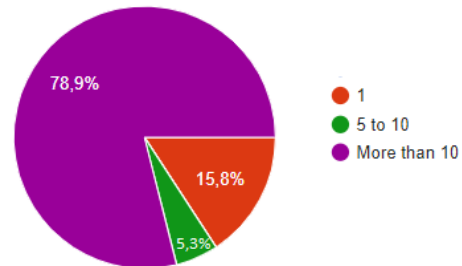


Figure A.1: Questions (1) and (2) summary of "Participant Characterization and Previous Experience" Section: (1) What is your education level? (2) How many people scientific experiments have you ever performed on computational environments?

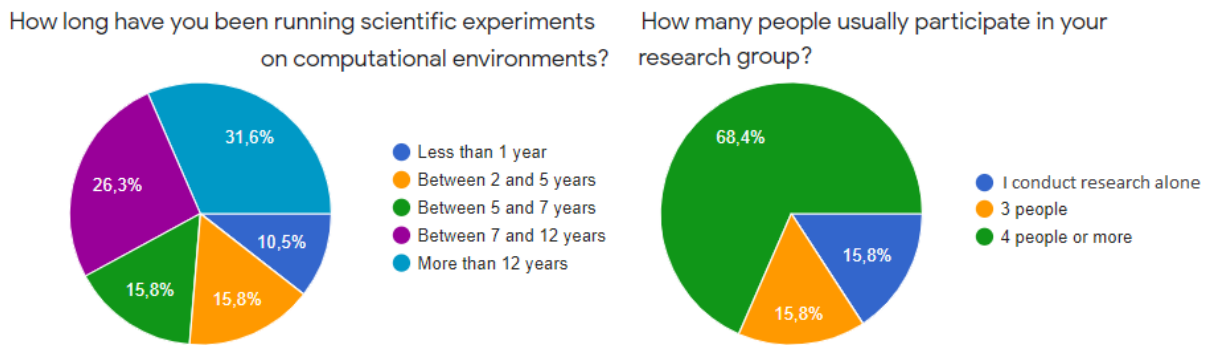


Figure A.2: Questions (3) and (6) summary of "Participant Characterization and Previous Experience" Section: (3) How long have you been running scientific experiments on computational environments? (6) How many people usually participate in your group of scientific research?

A.1 Participant Characterization and Previous Experience

In the first section, the objective is to determine the respondents' profile, how experienced they are, which tools they use, and if collaboration is part of their routine activities. Figure A.1, Figure A.2 and Figure A.3 summarizes the profile of the participants, their academic experience and how long they have been performing collaborative experiments. We can conclude, that in general, the respondents are very experienced researchers, since 68.4% has a PhD degree, 78.9% performed more than 10 experiments on computational environments, and 31.6% have been performing experiments for more than 2 years.

We also tried to understand if the respondents usually conduct experiments in groups and the size of those groups. As shown by the question "How many people usually par-

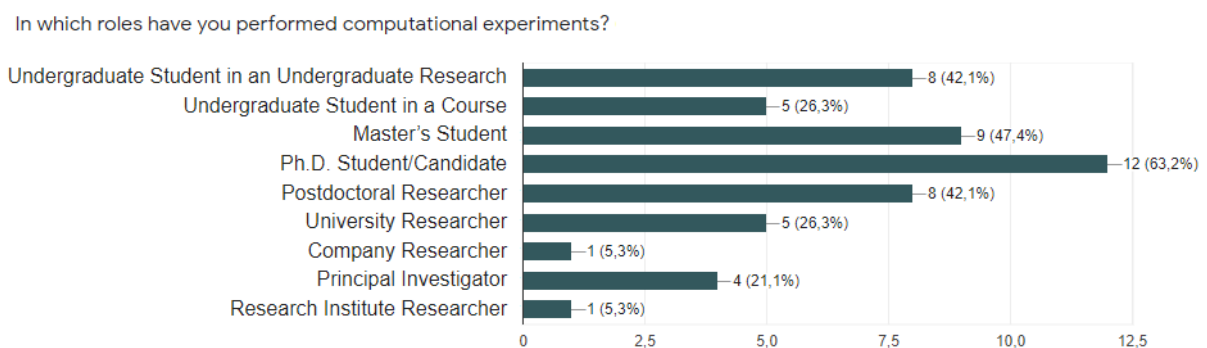


Figure A.3: Question (4) summary of "Participant Characterization and Previous Experience" Section: (4) In which roles have you performed computational experiments? (multiple answers allowed)

What are your preferred/more often used tools to run experiments? (select up to 3 tools)

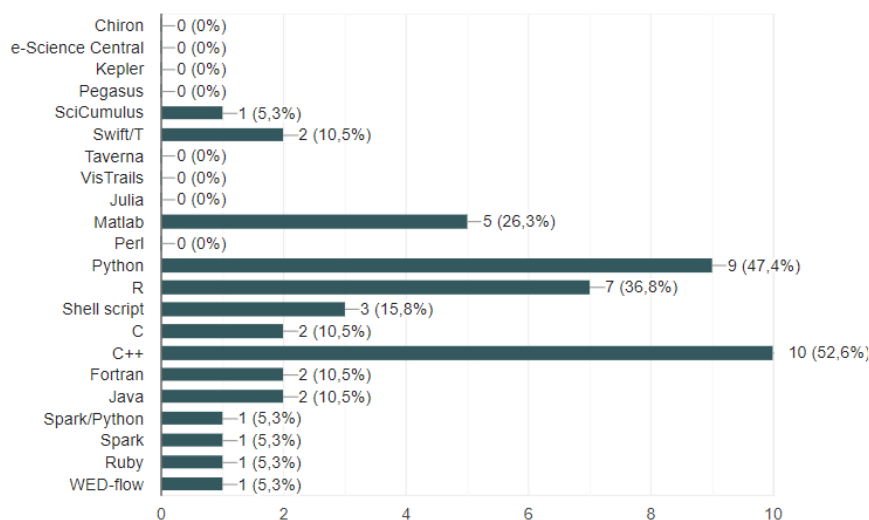


Figure A.4: Question (5) summary of "Participant Characterization and Previous Experience" Section: (5) What are your preferred/more often used tools to run experiments? (select up to 3 tools)

ticipate in your research group?", we can see that the vast majority conduct experiments in groups and that the groups are typically large (with 4 or more collaborators).

Another aspect covered by this questionnaire section regards which tools scientists use to run their experiments. The most used tools are programming and script-based languages (C++, Python, R, Matlab), as shown in Figure A.4. For illustration purposes, we choose to keep in the Figure the options that participants did not select.

These data show that these scientists typically conduct research in groups and do not use workflow-based solutions often – they prefer to use programming and script-based languages.

A.2 Collaboration Experience

Although the absence of proper provenance-aware tooling support for collaboration in script-based approaches has been identified as a research opportunity [44], a question arises: is this gap relevant from the scientist's point of view? This questionnaire section consists of 9 discursive questions aiming at this question.

The questions of this section comprehend questions 7-15, and we decide to take a deeper look here in two of the questions: "Do you face any difficulties when performing

joint research with other members of your research group?" and "Do you face any difficulties when continuing and/or reproducing the work of other researchers?". We collected the answers and we classified them as "yes" or "no" for quantitative purposes – blank answers were considered as "no". As shown in Figure A.6, the majority of respondents face difficulties in collaboration. Moreover, even those answering that do not have problems with collaboration answered that they face reproducibility problems when trying to continue or reproduce the work of other scientists in the group.

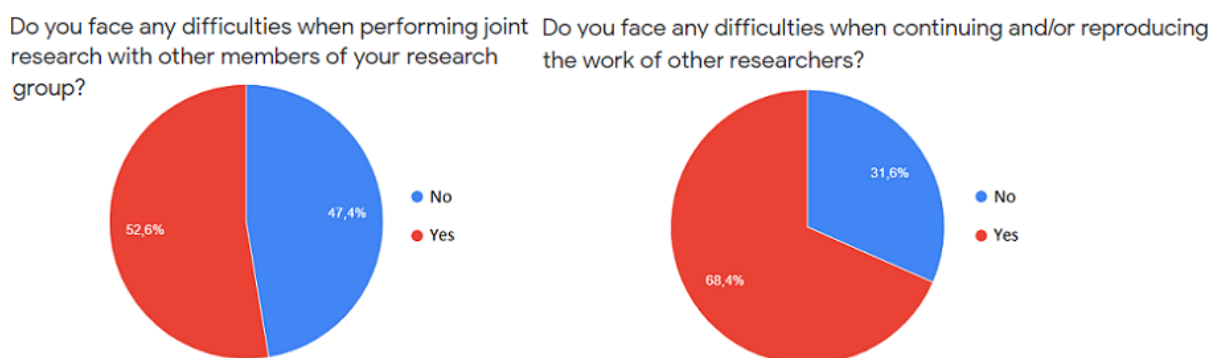


Figure A.5: Collaboration difficulties.

Figure A.6: Question (9) and (11) summary of "Collaboration Experience" Section: (9) Do you face any difficulties when performing joint research with other members of your research group? (11) Do you face any difficulties when continuing and/or reproducing the work of other researchers?

For the other questions is harder to summarize the obtained feedback, but is possible to notice that communication and tools to register the work done by the scientist within the group is a recurrent topic.

Those questionnaire results corroborated the gap identified by the survey we made (Chapter 2 and [44]) regarding the lack of support to collaboration aspects on Script-based systems and motivated us to take a deep look at this particular problem.

Table A.1: Questionnaire responses. Questions 1-8

Id	What is your educational level?	How many scientific experiments have you ever performed on computational environments?	How long have you been running scientific experiments on computational environments?	In which roles have you performed computational experiments? (check all that apply)	What are your preferred/more often used tools to run experiments? (select up to 3 tools)	How many people usually participate in your research group?	How do you get aware of the work done by other researchers in the group?	How can this experience be enhanced?
1	PhD degree	More than 10	More than 12 years	Postdoctoral Researcher	SciCumulus, Swift/T, R	4 people or more	Collaboration, reproducible works, publications	Collaboration in situ with expert in those technologies
2	PhD degree	More than 10	More than 12 years	Ph.D. Student/Candidate, Research Institute Researcher	Swift/T, Python, R	4 people or more	Through regular group meetings and collaboration tools like mailing lists and Slack.	
3	PhD degree	More than 10	More than 12 years	University Researcher, Principal Investigator	Matlab, Python, R	4 people or more	e-mail, collaborative tools	
4	PhD degree	More than 10	More than 12 years	Master's Student, Ph.D. Student/Candidate, Postdoctoral Researcher, University Researcher	Spark/Python	3 people	meetings, presentations	a system to manage the evolution of the research
5	Undergraduate degree	More than 10	Between 7 and 12 years	University Researcher, Company Researcher, Principal Investigator	Python, R, Spark	4 people or more	Conferences, articles, recommendations, etc ...	Greater integration and sharing
6	PhD degree	More than 10	More than 12 years	Undergraduate Student in an Undergraduate Research, Master's Student, Ph.D. Student/Candidate, Postdoctoral Researcher, University Researcher, Principal Investigator	Matlab, Python, C++	4 people or more	Weekly meetings	To keep track on the versions of the results more frequently
7	Master's degree in progress	1	Less than 1 year	Undergraduate Student in an Undergraduate Research	C, C++, Fortran	I develop scientific researches alone.		
8	PhD degree	More than 10	More than 12 years	Undergraduate Student in an Undergraduate Research, Undergraduate Student in a Course, Master's Student, Ph.D. Student/Candidate, Postdoctoral Researcher	Matlab, Python, C++	4 people or more	Update-meetings every 15 days	Maybe talking in English, we currently talk in Portuguese.
9	PhD degree	More than 10	Between 5 and 7 years	Ph.D. Student/Candidate, Postdoctoral Researcher	R	4 people or more	In person meetings, video conferences and e-mail	
10	PhD degree in progress	More than 10	Between 7 and 12 years	Undergraduate Student in an Undergraduate Research, Undergraduate Student in a Course, Master's Student, Ph.D. Student/Candidate	Matlab, C, C++	3 people	Talking, group meetings etc	I am not sure
11	PhD degree	More than 10	Between 7 and 12 years	Ph.D. Student/Candidate, Principal Investigator	Shell script, C++, Java	3 people	email and WhatsApp exchanges, git push/merge requests, shared Dropbox folders, shared Overleaf projects, shared Trello cards	We've been considering the use of Jupyter
12	Master's degree	5 to 10	Between 2 and 5 years	Undergraduate Student in an Undergraduate Research, Undergraduate Student in a Course, Master's Student	Shell script, C++, Fortran	4 people or more	We have a meeting per week	
13	PhD degree	More than 10	Between 7 and 12 years	Undergraduate Student in an Undergraduate Research, Undergraduate Student in a Course, Master's Student, Ph.D. Student/Candidate, Postdoctoral Researcher	Matlab, Python, C++	I develop scientific researches alone.		
14	PhD degree in progress	More than 10	Between 5 and 7 years	Undergraduate Student in an Undergraduate Research, Master's Student, Ph.D. Student/Candidate	Python, C++, Java	4 people or more	Presentations and meetings	N/A
15	PhD degree	1	Between 2 and 5 years	Ph.D. Student/Candidate	Ruby	I develop scientific researches alone.		
16	PhD degree	More than 10	Between 7 and 12 years	University Researcher	WED-flow	4 people or more	Using systematic review	It is not easy to improve this work based on systematic review
17	PhD degree	1	Less than 1 year	Postdoctoral Researcher	R	4 people or more	By scientific papers	
18	PhD degree in progress	More than 10	Between 2 and 5 years	Master's Student, Ph.D. Student/Candidate	Python, R, C++	4 people or more	Through meetings and e-mail.	We could have a tool that helps our communications and sharing of our experiments
19	PhD degree	More than 10	Between 5 and 7 years	Undergraduate Student in an Undergraduate Research, Undergraduate Student in a Course, Master's Student, Ph.D. Student/Candidate, Postdoctoral Researcher	Python, Shell script, C++	4 people or more	Throughout meetings	Improving the use of specialized computer tools

Table A.2: Questionnaire responses. Questions 8-15

Id	Do you face any difficulties when performing joint research with other members of your research group?	How can this experience be enhanced?	Do you face any difficulties when continuing and/or reproducing the work of other researchers?	How can this experience be enhanced?	Do your colleagues face any difficulties when continuing and/or reproducing the work done by you?	How can this experience be enhanced?	How do you think your collaboration experience could be improved?
1	Always a challenge		Medium		Medium		
2	Sometimes computer scientists want to test hypotheses that are not of interest to domain scientists and conversely.		Yes, instructions to execute a computational activity often don't work due to, for instance, version changes.	Maybe with better computational environment preservation techniques such as virtualization.	We usually adopt provenance management systems but this is sometimes not enough.	A more detailed provenance trace and better execution environment preservation.	Maybe having every participant to agree on adopting FAIR data guidelines and good practices for reproducibility.
3	Yes		Yes		No		
4	yes	use of communication tools whenever a problem is raised	yes	Integration of problem evolution and corresponding code	probably	improve documentation	diver more clearer the responsibilities and duties and be able to track them in the experiment being developed
5	No		No		No		More sharing opportunity
6	Not much, considering the time limitations.	To employ more visualization tools	Not frequently, when the records are ok.	To keep records ok regarding previous researches.	I don't know.	I am not sure	With an environment that would lead us to keep track and records of the experiments in a more productive way.
7							
8	I do, I am not Brazilian, there is a lack of inclusion in my lab. Both Brazilians and foreigners tend to create non-inclusive groups.	Promoting non-research activities and to include both Brazilians and foreigners.	Sometimes	Usually, I face math-related problems, I mean, bad math definitions in manuscripts.	I do not have feedback about it. But I think they do.	Talking with me, I always open to discuss research topics. Especially if I am involved.	We need to improve our math skills, English, and cultural isolation.
9	Achieving deadlines		Yes	By sharing code and data	Yes	Training	Yes
10	Not at all		It depends on how well it was documented and on the materials and methods made available by the authors		I don't think so, most of the time I document it in an easy way to be reproduced		
11	Yes, mostly on terminology and fundamental concepts (I work on a multidisciplinary group)	I don't know, apart from learning (hard!) the terminology of others!	Always!	Not sure. Jupyter and org-mode might alleviate, but not solve the problem	Yes, even though I try hard to help them!	I don't know, honestly	As I said, we've been considering the use of Jupyter, but it's not clear to any of us whether it will really help
12	None so far		A few		I don't think they tried reproduce my work yet		
13							
14	Yes	N/A	Yes	N/A	Yes	N/A	Yes
15							
16	No difficulties	It is not necessary.	Yes	We try to do this using open software and open data approaches	Yes, I have face some difficulties.	I am using the same solution for reproducing processes open software and open data.	improving the interaction with others researchers and with conference and journal forums.
17	No		Sometimes		Sometimes		
18	Yes. Communication issues.	We could have a tool that helps our communications and sharing of our experiments	Sometimes.		Not that I know.		My collaboration experience could be improved if we used a tool where it would be easy to register and communicate about our work set and experiment results.
19	Yes. Difficulty in expressing clearly a discussion topic	Increasing the use of practical collaboration activities during school	Yes	Improving the details / clarifying discussion given in their work	I do not know	-	Yes

APPENDIX B - Example of Bundle Serialized Object

Following there is an example of the "bundle" object serialized as a JSON. This object is used in the process of synchronizing the SQLite database between the machine acting as the server and the client machine. In this example, there is only one trial (the trial of id: "64872e6d-fbf9-4f26-9ac4-2f8d1da5199b") being synchronized, with all of its related entities. Note that since noWorkflow captures provenance in a fine-grained way, this JSON could be very large, so to save space we "cut" the lists to a max of 2 elements in this example.

```

1 {
2   'trials': [{
3     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
4     'id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
5     'script': 'teste_arquivo.py',
6     'start': '2021-10-10 20:17:02.630142',
7     'finish': '2021-10-10 20:17:02.812370',
8     'command': 'run teste_arquivo.py',
9     'path': 'C:\\Users\\eduar\\noWorkflowSamples\\testefile -
      Copia',
10    'status': 'finished',
11    'modules_inherited_from_trial_id': None,
12    'parent_id': None,
13    'main_id': 1
14  }
15 ],
16 'activations': [{
17   'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
18   'id': 1,
19   'name': '__main__',
20   'start_checkpoint': 0.12187930000000002,
21   'code_block_id': 1
22 }, {

```

```
23         'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
24         'id': 2,
25         'name': 'open',
26         'start_checkpoint': 0.12190689999999998,
27         'code_block_id': -1
28     }, ...
29 ],
30 'arguments': [{
31     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
32     'id': 1,
33     'name': 'script',
34     'value': "'C:\\\\Users\\\\eduar\\\\noWorkflowSamples\\\\
35         testefile - Copia\\\\teste_arquivo.py'"
36 }, {
37     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
38     'id': 2,
39     'name': 'bypass_modules',
40     'value': 'False'
41 }, ...
42 ],
43 'codeBlocks': [{
44     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
45     'id': 1,
46     'code': 'ca8f435ec508ed2b123c4317b77084c83582a544',
47     'code_hash': 'ca8f435ec508ed2b123c4317b77084c83582a544',
48     'docstring': ''
49 }, {
50     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
51     'id': 2,
52     'name': 'teste_arquivo.py',
53     'type': 'script',
54     'mode': 'w',
55     'first_char_line': 1,
56     'first_char_column': 0,
57     'last_char_line': 7,
58     'last_char_column': 7,
59     'container_id': -1
60 }, {
61     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
62     'id': 2,
63     'name': 'import os',
```



```
65         'type': 'import',
66         'mode': 'n',
67         'first_char_line': 1,
68         'first_char_column': 0,
69         'last_char_line': 1,
70         'last_char_column': 1,
71         'container_id': 1
72     }, ...
73 ],
74 'compositions': [{
75     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
76     'id': 1,
77     'part_id': 3,
78     'whole_id': 2,
79     'type': '*op_pos',
80     'position': 0,
81     'extra': None
82 }, {
83     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
84     'id': 2,
85     'part_id': 2,
86     'whole_id': 1,
87     'type': '*body',
88     'position': 0,
89     'extra': None
90 }, ...
91 ],
92 'dependencies': [{
93     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
94     'id': 1,
95     'dependent_activation_id': 1,
96     'dependent_id': 2,
97     'dependency_activation_id': 1,
98     'dependency_id': 3,
99     'type': 'argument',
100    'reference': False,
101    'collection_activation_id': None,
102    'collection_id': None,
103    'key': None
104 }, {
105     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
106     'id': 2,
107     'dependent_activation_id': 1,
```

```
108         'dependent_id': 2,
109         'dependency_activation_id': 1,
110         'dependency_id': 6,
111         'type': 'argument',
112         'reference': False,
113         'collection_activation_id': None,
114         'collection_id': None,
115         'key': None
116     }, ...
117 ],
118 'environmentAttrs': [{
119     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
120     'id': 1,
121     'name': 'OS_NAME',
122     'value': 'Windows'
123 }, {
124     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
125     'id': 2,
126     'name': 'OS_NAME',
127     'value': 'Windows'
128 }, ...
129 ],
130 'evaluations': [{
131     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
132     'id': 1,
133     'code_component_id': 1,
134     'activation_id': 0,
135     'checkpoint': 0.12409520000000002,
136     'repr': "<module '__main__' from 'C:\\\\Users\\\\\\\\eduar\\\\\\\\
noWorkflowSamples\\\\\\\\testefile - Copia\\\\\\\\teste_arquivo.
py'>"
137 }, {
138     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
139     'id': 2,
140     'code_component_id': 7,
141     'activation_id': 1,
142     'checkpoint': 0.123029,
143     'repr': "<_io.TextIOWrapper name='teste34' mode='w+'
encoding='cp1252'>"
144 }, ...
145 ],
146 'fileAccesses': [{
147     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
```

```
148         'id': 1,
149         'name': 'teste34',
150         'mode': 'w+',
151         'buffering': 'default',
152         'content_hash_before': '
           e7515ea57031994f29dbde1b213ce439ab588aa9 ',
153         'content_hash_after': '
           da39a3ee5e6b4b0d3255bfef95601890afd80709 ',
154         'checkpoint': 0.12197859999999999,
155         'activation_id': 2
156     }
157 ],
158 'members': [{
159     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
160     'id': 1,
161     'collection_activation_id': 0,
162     'collection_id': 5,
163     'member_activation_id': 0,
164     'member_id': 5,
165     'key': '.__class__',
166     'checkpoint': 0.121920100000000003,
167     'type': 'Put'
168 }, {
169     'trial_id': '64872e6d-fbf9-4f26-9ac4-2f8d1da5199b',
170     'id': 2,
171     'collection_activation_id': 0,
172     'collection_id': 4,
173     'member_activation_id': 0,
174     'member_id': 5,
175     'key': '.__class__',
176     'checkpoint': 0.121920100000000003,
177     'type': 'Put'
178 }, ...
179 ],
180 'modules': [],
181 'users' : []
182 }
```
