THE UNIVERSITY
of EDINBURGH

Active Provenance for Data Intensive Research

Alessandro Spinuso

Doctor of Philosophy
Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh

2018

Abstract

The role of provenance information in data-intensive research is a significant topic of discussion among technical experts and scientists. Typical use cases addressing traceability, versioning and reproducibility of the research findings are extended with more interactive scenarios in support, for instance, of computational steering and results management. In this thesis we investigate the impact that lineage records can have on the early phases of the analysis, for instance performed through near-real-time systems and Virtual Research Environments (VREs) tailored to the requirements of a specific community. By positioning provenance at the centre of the computational research cycle, we highlight the importance of having mechanisms at the data-scientists' side that, by integrating with the abstractions offered by the processing technologies, such as scientific workflows and data-intensive tools, facilitate the experts' contribution to the lineage at runtime. Ultimately, by encouraging tuning and use of provenance for rapid feedback, the thesis aims at improving the synergy between different user groups to increase productivity and understanding of their processes.

We present a model of provenance, called S-PROV, that uses and further extends PROV and ProvONE. The relationships and properties characterising the workflow's abstractions and their concrete executions are re-elaborated to include aspects related to delegation, distribution and steering of stateful streaming operators. The model is supported by the *Active* framework for tuneable and actionable lineage ensuring the user's engagement by fostering rapid exploitation. Here, concepts such as provenance types, configuration and explicit state management allow users to capture complex provenance scenarios and activate selective controls based on domain and user-defined metadata. We outline how the traces are recorded in a new comprehensive system, called `S-ProvFlow`, enabling different classes of consumers to explore the provenance data with services and tools for monitoring, in-depth validation and comprehensive visual-analytics. The work of this thesis will be discussed in the context of an existing computational framework and the experience matured in implementing provenance-aware tools for seismology and climate VREs. It will continue to evolve through newly funded projects, thereby providing generic and user-centred solutions for data-intensive research.

Lay Summary

The concept of *Provenance* is subject to different interpretations. Nonetheless, it applies to a large number of domains, which brings in new challenges and requirements mostly driven by striving for a pragmatic and effective use. Consensus on its definition is limited to ”the identification and description of the origins of artifacts”. In archival science this is knows as *respect-de-fonds*. This fundamental concept is extended with additional properties associated with the influence and the role that other players had on the generation of the subject, that vary depending on the specific concern.

In computational-intensive and data-intensive research, effectiveness of provenance generation and management is overwhelmed by the complexity of its potential granularity, scale and intrinsic overhead. Thus, how to systematically capture these influences in a way that can be effectively exploited has been considered for long time the *Elephant in the Room*. This has lead to the current reproducibility crisis experienced in many scientific fields such as Physics, Biology, Earth science, Medical science *etc.* This bears the risk of undermining trust in the implications of the research.

In this thesis we explored and delivered solutions to enrol provenance earlier in the research practice, engaging the developers of the scientific tools and their users in the *Active* participation in its production. They are encouraged to do this experiencing improved productivity in their current work. We investigated ways to foster engagement in the context of an existing computational framework, aiming at implementing provenance-aware tools in Virtual Research Environments. VREs provide an integrated and convenient work context to help researchers perform their investigation. They typically involve tailored user interfaces to prepare and execute scientific workflows and manage their outputs. The latter eventually contribute to publishable results. In VREs scientists, research-developers, data-architects and managers are involved, with different roles, in the “computational-research cycles”. They all benefit from the provenance information, when its production can be tuned and its access adapted to different use cases and universes of discourse. We took into account a spectrum that spans from fine-grain details to the aggregation of campaigns that entail collaboration among users, involving a large number of intermediate experiments and metadata, selectively merging domain specific concepts with the underlying technical details.

Acknowledgements

This thesis was made possible thanks to the interactions with an uncountable number of inspiring people. Scientists, engineers, managers and students – each of them contributed to make this a journey that was worthwhile pursuing. The guidance provided by my supervisors Malcolm Atkinson, James Cheney and Rosa Filgueira has been fundamental and has represented a source of intellectual and professional growth. I thank them for their dedication and for being welcoming hosts during my frequent visits to Edinburgh.

My gratitude also goes to Albert Klein Tank who persuaded KNMI to endorse and support this challenge, with the encouragement of my colleagues from the Observations and Data Technology Department. Thanks to Andrej and Thomas for their energy, to John and Wim for their commitment to the professional and social wellness of this group.

This thesis was also supported by many projects that provided the diverse and stimulating scenarios that inspired my research. I want to thank all my colleagues from the projects VERCE, CLIPC, EPOS and finally DARE, where we will continue to collaborate and progress. Especially in this context, I want to thank Torild van Eck, who was my first boss, colleague and friend when I started my professional career at KNMI. Torild inspired and made possible the success of many projects I was and I am still involved in. His positive energy and ability in establishing cooperations across countries and cultural differences survive and continue to drive us further within the competitive and multidisciplinary landscape we work in.

A special thanks goes to my beloved family in Italy for their constant presence and affection. Finally, although it is impossible to mention them all, I want to thank all my friends who have contributed, with their supporting words and patience, to a healthy and serene progress of the writing of this thesis.

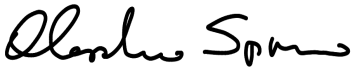
Declaration

I confirm that the work submitted is my own, except where work which has formed part of jointly-authored publications has been included. These cases are enumerated here and in the Publications section 7.6. I confirm that appropriate credit has been given within the thesis where reference has been made to the work or discussions with others. The specification of the use cases supported by the provenance system presented in this thesis were discussed and validated within a team of domain and technical experts in the context of funded FP7, H2020 and *Copernicus* projects. (VERCE, DARE, CLIPC). I confirm that this work has not been submitted for any other degree or professional qualification except as specified. Contributions and material provided by other authors to this work have been explicitly indicated below.

Chapter 5. The work mentioned in sections 5.6 and 5.5.2 referring to indexing strategies and map-reduce postprocessing was conducted in cooperation with Thomas Kok in the context of a Bachelor of Science project. I personally defined the project and supervised the student throughout his internship at KNMI, until his successful dissertation in January 2018. Thomas kindly provided Image 5.7. Andrej Mihajlovski, developed the *Docker* containers for the deployment of the `S-ProvFlow` system.

Chapter 6. The the GUI components for the combined integration of workflows' results and the configuration of preprocessing pipelines of figures 6.5 and 6.6 were developed by Jonas Matser and Rafiq Saleh. The experiment shown in Figure 6.3 was performed by Federica Magnoni. The simulation and misfit analysis workflows introduced in Section 6.1.1 were developed in cooperation with André Gemünd, Amrey Krause, Cerlane Leong, Rosa Filgueira, Federica Magnoni, Emanuele Casarotti and Lion Krischer. The integration of provenance in the CLIPC portal, see Section 6.1.2, was realised in cooperation with Andrej Mihajlovski, and Maarten Plieger. Andrej Mihajlovski also contributed to the initial implementation of the *Jupyter Notebook* page.

Chapter 7. The project DARE is a funded H2020 initiative. Partners are mentioned on the project website at `http://project-dare.eu/consortium/`



(Alessandro Spinuso)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The role of provenance information in data-intensive research is an actual topic of discussion among technical experts and scientists. Especially with the ever growing variety of data and the advances in computational software libraries, new challenges emerge and the typical use cases addressing traceability, versioning and reproducibility of the research findings, are extended with scenarios in support, for instance, of computational steering, results management [183, 124, 118] and the production of *report-worthy* provenance traces [92]. As suggested by Myers *et al.* [195] an evolving documentation of the experimental process, characterised by precise traces rich with domain and experimental metadata, benefits users in the early stages of their work. It supports the improvements of their methods and has the effect of reducing the efforts required for an effective curation of the scientific products.

Yet, it is fundamental to consider what Boose, et al. [105] state in their work. That is, a dataset is reliable when the scientific process used to create it is reproducible and analysable for potential defects. Thus, to conduct the analysis of a scientific process it is important to produce meaningful provenance traces in the first place. Issues may occur in methods at different granularities and fully automated mechanisms can hide details or provide too many, hindering the usefulness of the provenance. Moreover usefulness must be supported by interactive visual tools that support meticulous validation, as well as comprehensive perspectives and visual summaries. These involve very large quantities of provenance traces produced by different users and workflows.

This work will describe and demonstrate a provenance framework that assists the users in the configuration and contextualisation of provenance information within data-intensive systems, fostering its usable exploitation in different phases of their research practice. We will show how an *Agile* approach to the engineering of provenance management enables runtime analysis, short feedback-loops and workflow portability, with special attention to streaming systems. This is all supported by tools for a detailed validation of the computation and comprehensive visual analysis, fostering reproducibility, data reliability and exposing insights in collaborative processes and exploitation of resources.

Our integrated and holistic approach supports different classes of users and allows the collection of the provenance data at tuneable precision. We consider expert users as part of the process of providing provenance-sound data-intensive applications to be used by researchers and the wider community, including outreach officers, managers of community platforms and data-architects to gain insights into the exploitation of the resources. In this chapter we will introduce the reader to the general concept of provenance, its interpretations and why it is considered important in modern computational disciplines. We will present an overview of the common challenges and the initiatives to develop guidelines for its adoption. Finally, we summarise the contributions provided by this thesis and the main contents of each chapter.

1.1 Provenance Interpretations

The concept of Provenance may be subject to different interpretations, mostly related to its intended use. Nonetheless, it applies to an extremely large number of domains, which brings in new challenges and requirements mostly driven by striving for a pragmatic and effective use. The archivist Shelley Sweeney, writes in *The Ambiguous Origins of the Archival Principle of Provenance* [214] that:

"Broadly speaking, the word provenance, whether used by a rare book librarian, an archaeologist, an art curator, or an archivist, refers to the origins of an information-bearing entity or artifact. But there the consensus ends."

The archivists belonging to a very special historical period gave a large contribution to the establishment of one of the most discussed interpretations of provenance, known as *respect des fonds* [130]. The foundation of this pragmatic and innovative principle was established in France, already in the XIX century. It was driven by the need to merge private and public archives throughout the country into a single national archive (Les Archives Nationales), in a period when important and rapid changes were determined by the political turmoil brought by the uprisings of the French Revolution. The first 50 years unveiled a number of challenges that kept the archivists busy, trying to regroup and classify the massive amount of documents received. This task revealed itself to be far from easy. Pursuing a classification based on the objective properties of the artifacts, was not considered informative enough and it had to be complemented with additional contextual information about the original archive and the initial classification and description of the artifact. This brought the evidence that the information collected by the original archivist was far more important than the object itself and led therefore to the definition of the principle according to which:

The unity of the archive took precedence over the material objects

This principle better fits also the context of this thesis, since it states how important it is to consider a classification that takes into account the information selected by the initial curator of an artifact, relying on the assumption that they own the best knowledge about the subject and the domain to which it is associated. As already mentioned, this is extremely important when talking about provenance, since it totally affects the way it is perceived and used.

Below are reported as an example, a few different ways of looking at the interpretation of provenance, according to the context:

- *Archeology:* The place where an object was found or recovered in modern times; the findspot
- *Librarians:* Information concerning the transmission or ownership, as of a book. (Never used for retrieval)
- *Geology:* The reconstruction of the history of sediment movements over time (Many types of detrital records to unveil regional tectonic history)

- *Wines:* A documented history of wine cellar conditions, influencing transactions of old wine with the potential of improving with age
- *Data-science:* Assisted validation of the data products through their description, in the evolving context of the generating methods, infrastructures, domains and users' roles, at scale, fostering the shareable understanding of reproducible results.

Methods for the adoption of provenance as a first class citizen in the production of scientific data are widely discussed in the literature. Typically, the aim is towards a better support for the curation of the research results, for their reproducibility and the long-term understanding of the associated processes. It is a fundamental element of the FAIR data principles [221], which state that research data must be *Findable, Accessible, Interoperable, Re-usable*, with a special role in assuring re-usability.

1.2 Users and Usage in Data Science

Recording provenance is often coupled with the adoption of scientific workflow systems, or with campaigns dedicated to data acquisition and preservation. In raw-data acquisition, provenance can be used to log the changes in, for example, a seismic sensor's configuration, coupling these with geolocated data, mutations in the soil condition, *etc.* ċomplementing and improving the overall quality of the recorded raw-data. The curation of data, in general, requires to provide a provenance coverage which spans across its acquisition or generation, its processing and attribution.

In the literature other ways of enrolling users with provenance practices have been explored, especially for what concerns the processing tasks. They tend to be less dependent from the machinery and the adopted software, promoting a more explicit control in their hands. Obviously this approach may open issues concerning the quality of the provenance produced, suggesting the need for introducing evaluation strategies and quality measures, which may be difficult to put in place. The solution pursued by this presented work, is in favour of mitigating the poor quality and usability shortcomings of the production and exploitation of provenance information by offering an

adequate support to mediate the production of this type of data, keeping the researcher developing new methods at the centre of the whole process.

This controversial but still precious source of information, sees the growing interest from its potential users, which can be divided, for the purposes of this thesis, into four different groups.

- *research-developers:* Use advanced systems to realise tools to perform a specific class of analysis. Empirical evaluation, based on direct observations at different scales and execution environments. They may access facilities offering Platform as a Service (PaaS), to perform their evaluation on appropriate resources.
- *end-users (domain-scientists):* Use these tools via virtual environments offering Software as a Service (SaaS). Confidence and trust in the tools is incremental. It requires consistent feedback and contextual information. It is important for domain-scientists to understand which results can be re-used from past experiments. These may be produced also by others in a collaborative environment.
- *data-architects:* This class of users benefits from detailed insights in the way the advanced systems are operating. The understanding of the details associated with the execution performed by their software suggests possible improvements and therefore better systems.
- *data-curators:* Focus their interest on the role of provenance for the long-term preservation of the scientific results. These can therefore be referenced and their veracity guaranteed in the context in which they have been created, progressing towards reproducibility and consistent exploitation over time.

Scientists definitely appreciate experiencing an immediate impact in the daily practice of their ongoing research. Having ways to effectively exploit provenance for the rapid validation and comparison of the experiments, can massively reduce the scientist's workload. They no longer have to manage their sometimes obscure logs that are often produced by different systems, making their data generally better understandable and improving at the same time collaboration among peers [188].

To make an additional differentiation in the scientists as a group of users, we have

to consider that while some researchers readily adopt off-the-shelf tools to analyse and produce data, according to known formats and *de facto* standards, others, especially those adopting new data-intensive methodologies, deal considerably with an environment where self-developed algorithms crunch and create large quantities of data. These data can either represent the final result of their research or, in many cases, are intermediate stages in its progress.

Scientists and research-developers retain the best knowledge of what data flows in their code and what is important in order to understand their processes. To encourage the adoption of a provenance data-model requires us to provide a level of automation that balances between the identification and the classification of intrinsic properties of a computation, providing enough flexibility for the annotation of sensible characteristics, as they come along [184].

1.3 Challenges in Provenance for Data-Intensive Research

This section presents a number of aspects related to the management and the fruition of provenance data. Although these may be commonly associated with any kind of research practice, they are particularly challenging in the data-intesive context [151]. Here, systems are involved in parallel computations processing growing data volumes produced by sensors and simulations. They are required to provide scientists with rapid feedback to recognise and validate significant results. This suggest the need for new metadata models and management technologies that can deal with heterogeneity, distribution, concurrency and complexity of the methods and the data-resources. These requirements set the background for a more detailed discussion, trying to answer the following questions. What is the role of provenance in data-intensive scenarios and what is the level of completeness required to be considered both useful and manageable? This thesis provides initial work pointing to promising approaches.

- *Efficiency of provenance collection.* There are situations where a potential overhead may discourage researchers from pursuing provenance recording actions. In seismic interferometry [103], new applications aimed at the monitoring of

CO_2 capture and sequestration projects, may feed in near real-time massively parallel processing pipelines and cross-correlation algorithms with thousands of seismic traces. The need for rapidity and accuracy can not be underestimated to assure that life-threatening hazard brought by CO_2 plumes reaching the surface of a containment basin, can be timely mitigated. Similar approaches are already covered in literature, targeting the forecasting of volcanic eruptions based on *quasii*-real-time monitoring of relative velocity changes [131]. Here, tracing the intermediate changes affecting the incoming flow of data, may be extremely costly, although still be worthwhile for runtime diagnostic purposes triggering immediate corrections.

- *Handling of provenance for streaming data.* Stream-based processing is a computational paradigm that is suitable for a variety of data-intensive use cases, such as those mentioned above. It is currently implemented by several software frameworks. Most of the data flowing into and transformed by such systems are consumed at a very high rate and intermediate stages are often volatile, thus, making the injection of lineage recording procedures complex and expensive. Because of the different priorities across customers, use cases and performance comparisons, which often makes one product look better than another, popular streaming engines, such as Apache Storm [5], do not provide built-in provenance mechanisms, focusing more on scalability and performance aspects, leaving this task as an open research and systems-engineering issue. Given the high demand in I/O of such digital ecosystems, even storing partial lineage data could be unmanageable and even uninformative, limiting the chances of its effective exploitation. Efficient approaches to collect and structuring provenance for data-intensive scenarios are needed, in favour of a concise representation for its production, storage and access.

- *Control by researchers of the provenance collected.* How can we scale the concept of the *respect des fonds* to data-intensive practices in a sustainable and effective way? How can we involve the researchers in the archival process from its early stages, returning immediate benefits at an acceptable overhead? This requires also to guarantee a certain level of freedom in letting users choose the

data-intensive engine which best fits their need or the tools supported by the available computational infrastructure.

- *Automated use of provenance data.* Considering provenance as actionable data may allow the automation of metadata-driven operations, such as transfers of data across infrastructures at runtime, or the allocation of dedicated resources for the post processing of intermediate results. For instance, intermediate raster graphical content could be immediately rendered by a dedicated system, which could be different from the architecture executing the computation. In these circumstances, the target system may even take into account contextual information associated with the provenance of the received data (i.e. the adopted parametrisation and the relevant dependencies), to enrich the presentation with complementary information.

- *Consistent and rapid provenance collection across heterogeneous systems:* Guaranteeing ways to rapidly produce and visualise provenance data regardless of an infrastructure's limitations, security policies and enactment architectures, would foster the evaluation of preliminary runs across computational environments, (*e.g.* when the same code is moved from the scientist's personal machine to larger clusters) as well as long lasting production runs. Runtime access may save scientists waiting for the evidence of the correctness of their current process, enabling them to save expensive computing resources and energy when they spot an error earlier.

  In modern computational environments, concepts of SaaS and PaaS (Software as a Service and Platform as a Service) blend within a single integrated container. They offer access to a variety of resources, from HPC to cloud, provided by diverse and autonomous organisations, which host ready to use scientific software. Integrating these services as being part of a federation, favours loose coupling and fosters synergy to assure that the requirements of many research contexts are preserved across the various infrastructures. Integrating a pervasive provenance framework that merges standard models and contextual information contributes to reach a common quality goal for the services offered, in terms of usability, consistency of the archival operations, results validation and ultimately repro-

ducibility.

Another scenario may be the analysis of large datasets generated by research teams that are geographically distributed and that typically work in isolation on similar tasks. For instance, in the Human Brain Project (HBP) [27, 28], large datasets are distributed across partners and need to be integrated to study the dynamic nature of the interactions occurring in the whole brain. By using a provenance model the team produced an example showing how the workflows followed by different labs lead to similar results, for instance when estimating cell distribution and volume estimation, thus, bringing together experimental work and enabling its reconstruction and cross validation. The team acknowledges the need of important services, such as a monitoring and knowledge API, and that the provenance records should be capable of linking to the original data resource via UUIDs (Universally Unique Identifiers). This case may present similar requirements to the analysis and the integration of the content generated by popular web communities, where the data production rate and its size require scalable and distributed acquisition and processing mechanisms, to combine different sources of information, identifying relationships, recurring patterns, suggesting actions and recommendations.

- *Ease of use of provenance data by researchers:* Even when efficiency and a certain level of descriptive completeness may have been achieved, reducing the risk of a cognitive overload is an essential prerequisite fundamental to making use of provenance effectively. This is an issue that mainly concerns the interactive provenance consumption by the users [125]. Provenance is data that has to be understood consistently by the researcher and is not subject to transformations. It definitely stops being useful when too much approximation leads to erroneous or incomplete interpretations. Its presentation should offer qualitative views, as well as fine-grain details, providing exploration tools that do not overwhelm the scientists. A balance is needed between the typical DAG (Directed Acyclic Graph) presentation, based on established dependencies, and the combination with retrieval techniques relying, for instance, on pre-identified combinations of metadata and hierarchical classifications or *facets* [216, 204]. These could

be proposed to the user as a means for discovery and exploration through advanced visualisation techniques. Moreover, provenance information should be directly exploited by virtual research platforms dealing with distributed systems and complex applications, for instance as we experienced in the VERCE Virtual Research Environment for Seismology, see Figure 1.1. In Chapter 6 we will describe the use cases implemented within the platform and the approach to provenance management and exploitation in more detail. Finally, providing useful tools based on provenance motivates researchers to adopt and review provenance data. This improves quality assurance of results particularly in collaborative contexts.

We could argue that hazard mitigation systems may not be considered as research applications *per-se*, suggesting that provenance could be discarded in favour of better performance. Though, the experimental phases required for the realisation and the tuning of such systems can benefit from an exhaustive runtime documentation related to test runs, it can also be used for regular quality checks. Comparisons of different hazard mitigation systems is a vital issue and needs to be established carefully. Provenance may help in this task. Therefore, we can consider a number of mitigating factors that makes the development of a provenance strategy also worthwhile in this context.

Users may accept that rapidly produced large volumes of provenance data do not need to persist long, thus making the exploitation of fast storage systems offered by new technologies [1] possible with acceptable risks. Engaging the researchers and developers in decisions about which provenance data should persist and for how long is as important as engaging them in shaping the selection and the content of the provenance data produced. Hence, a tuneable and selective framework embedded in a data-intensive system coupled with tools that enable detailed exploration, as well as layered and comprehensive views on the provenance data is a valuable option. It contributes to the realisation of complex experimental applications until their deployments in operational contexts. Once integrated into production environments, provenance can contribute to quality assurance and support automation and optimisation. Though, trade offs may be different in this context.

Finally, the overhead introduced by the handling of large volumes and rates of prove-

nance information in data-intensive research, has to be minimised when the rapidity in decision-making is a priority or when time and costs may become a project life-threatening parameter, as often happens in science. This suggests that implementing solutions offering a good balance between systems and human performance in the cycle of scientific methods, is a route that is worth pursuing.

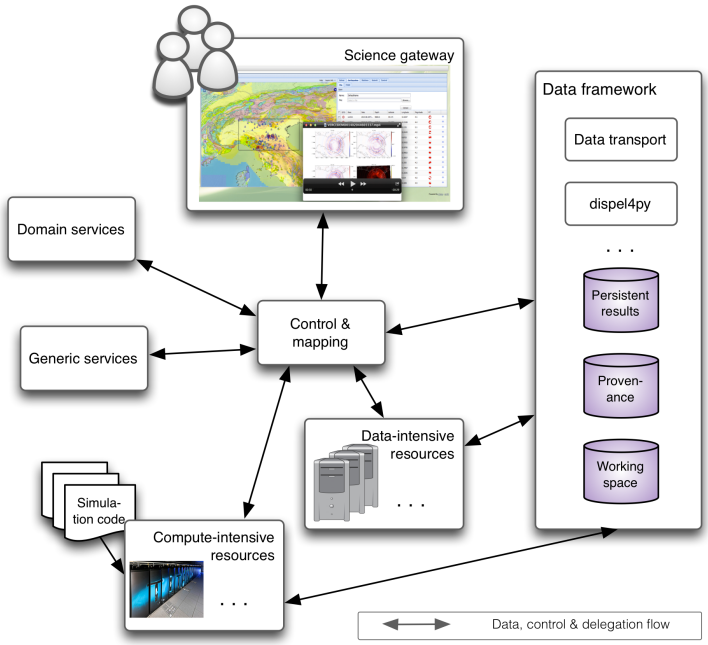


Figure 1.1: Overview of the VERCE e-Infrastructure architecture.

1.4 Active International Initiatives

Despite all the technical and conceptual challenges, the importance of the role of provenance data for the advancement of scientific production, at all scales, and its preservation, are still considered a fundamental issues to be addressed. Provenance matters are discussed globally, trying to evaluate and establish policies, data models and best-practices. More specifically, active discussions are currently ongoing in several international initiatives. The Research Data Alliance (RDA)[64] hosts a number of working groups, that introduced a comprehensive conceptual framework for reproducible science. These are known as The Data-fabric [60], Research Data Provenance [61] and the most recent Provenance Patterns Working Group [63]. Provenance

management also gained the attention of the DataOne [16] organisation, funded by the National Science Foundation (NSF) [77], promoting innovation for environmental science. DataOne supports the specification of the ProvONE Data Model [52], an extension of the more generic W3C recommendation PROV [83], see Figure 1.2, which aims at unifying the provenance representation across scientific workflow systems.

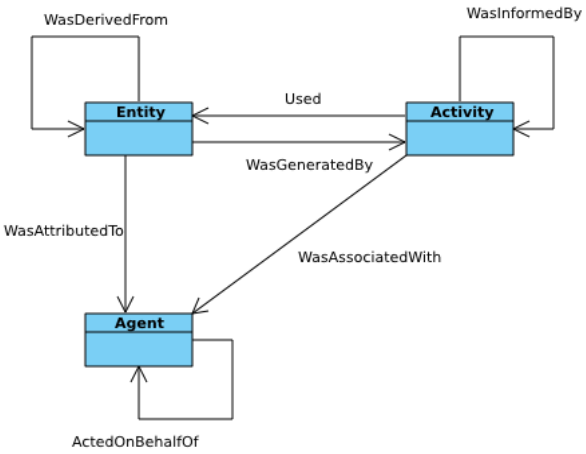


Figure 1.2: Schema of the essential elements of the PROV data-model: *Activities* inform other *Activities* by means of an exchange of information and they generate and use *Entities*. *Entities* can be derived from other *Entities*. Respectively they are associated and attributed to *Agents* who perform or delegate an action. Image obtained from the PROV online documentation [83]

Having personally followed the discussions in a number of RDA working groups, the recurrent issue related to preservation and metadata management, is the evident gap between the data managers' priorities and the scientists' daily practice, as already anticipated in Section 1.1. While the former pushes for standardisation, strongly typed classification and agreed procedures, the latter deals with a more unpredictable and unexplored landscape, with grey areas and dynamism in the interpretation of the data and the ongoing conceptual work. This turns into having the scientists worried that the adoption of standardised methods and data-management procedures introduces a set of limiting constraints, causing unnecessary overkill and delays in the actual progress of their research. As a consequence, they start approaching the data preservation services offered by the institutional data archives only at the time of their publication, when much useful information about the process leading to a certain final result may already be lost or, in many cases, never produced, leading to what nowadays is acknowledged

as a reproducibility crisis in scientific research. "Nature" has assembled an archive of reports and analyses, called *Challenges in Irreproducible Research*, where concerns about the problem are motivated by documented evidence [9]. Encouraging the researchers to approach data-intensive challenges with the possibility of influencing effectively the production of the metadata and of meaningful provenance traces, from the preliminary phases of their projects onwards, is an open challenge that has to be addressed. The solutions investigated by the presented work, acknowledge the need to bridge the gap between data curators', data managers' and researchers' viewpoints, accommodating the different priorities emerging during the stages of the research practice.

1.5 Research Contributions

This thesis focuses on how the productivity of data-intensive investigations can benefit from a human-guided acquisition of provenance information and from its rapid exploitation. By looking at the limitations experienced by the most common practices in computational sciences, it motivates the need for an interactive and configurable provenance framework that can combine the provenance of the underlying computational model with the context of the scientific investigation, facilitating user contributions. It promotes provenance-aware thinking and provenance-driven actions to assist the different phases of the ongoing study, in the shorter as well as longer timescales. It develops a balance between automation and human intervention, aiming at facilitating the precision and usefulness of the lineage information produced by digital means. This requires human-machine and machine-machine short-cycled feedback loops, managed at scale.

The context is set around stream-oriented computational systems and the Virtual Research Environments (VREs) [136] where these are employed. The conceptual work starts, bottom-up, by looking at the computational characteristics of a streaming-system. It refers to the basic principles of the *Actor Model* concerning actors' atomicity and their concurrent interactions. Then, the formal background is transposed to a more schematic model that specialises general PROV and ProvONE abstractions, combining

the observable elements of the analysis, with system and user contextual information.

The thesis acknowledges the need for smoothing the intellectual and technical ramps experienced by the targeted categories of a community. It takes up the challenge of investigating easy paths to gain early benefits from a provenance aware computation. The provision of views and tools, which have been designed, prototyped and exposed in official complementary training programmes, suit four categories of user: domain scientists, ITC experts, platform operators, and research leaders. The tools aim at lowering the entry costs in the adoption of a common intellectual and technical framework in which all four categories of R&D effort can be pooled, as envisage in the Data Bonanza book [98] and in the ABC of research [209]. Especially the latter, promotes an approach to the research life cycle that combines basic and applied research that has to be supported by engineering and design. The way this thesis was developed and the solutions provided try to follow these principles, towards the evaluation through realistic prototypes and conducting a communication and collaboration campaign for the adoption by target research communities. Eventually this will engage them into discussions on technical and conceptual aspects, aiming at improving the systems in parallel with the identification of new conceptual challenges.

The contributions are now presented, starting with introducing the concept of *Active Provenance*, followed by the overview of the achievements, in relation to the general challenges and the pursued principles.

(C-1) *Active* Provenance: the PhD's conceptual and technical work develops and advocates the idea of *Active* provenance. It suggests and demonstrates how the participation of provenance information can be beneficial from the early phases of the research practice onwards. It encourages the selective and semi-automated enrichment of provenance information for its actionable participation in the production of research data. *Active* provenance technology allows for multilevel and reactive monitoring of context-rich traces that, on the short-term, can improve methods and trigger computational steering operations. On the medium-to-long-term, its proximity to the progress of the research facilitates the re-use of intermediate and experimental products, as well as final and polished results. Thus, stimulating and revealing collaborations among peers, at different granularities. This is achieved by combining, in a retrospective anal-

ysis, tuneable user-driven and domain-specific metadata with information about the computational dynamics and the methods involved in the creation of research products. It accommodates priorities and requirements across different classes of users and usage.

(C-2) Provenance for data-intensive systems: a model is defined, which uses and further extends PROV and ProvONE concepts for the description of a stream-oriented and data-intensive system. The model includes aspects related to delegation, concrete mapping and steering of *stateful* operators.

(C-3) Scale of the provenance records: while the model has to hold its knowledge within comprehensive records, whose usability stretches over time, special attention has to be dedicated to a selective management of the provenance records that users want to be captured and processed. This thesis proposes solutions offering tuneable provenance and a practical approach to its representation, storage and exploitation. The proposed system encourages a provenance-aware design of data-intensive applications, aiming at an increased precision of the provenance information, with immediate benefits at a controllable overhead.

(C-4) Support for multiple levels of understanding: provenance information can be interpreted at different levels, from technical to high-level abstractions. We devise and demonstrate possibilities of tuneable context-switching across concerns, providing high-level views from detailed records, bridging across interests and understanding of the different classes of players in a computational environment.

(C-5) Integration with tooling: the capture of provenance data is conducted by exploring a modular and dynamic approach. We pursue ways to store, manage and interactively exploit the provenance data addressing a number of use cases. An outline of the main topics and the associated contributions is summarised below.

- *Engineering of selective and actionable provenance.* Modern workflow systems should stimulate and support the implementation of new workflow applications which are provenance-aware. Taking into account the principles illustrated in (C1-C4), a software architecture approach to integrating semi-automated provenance collection has been developed, refined and tested in realistic contexts. It

relies on a combination of types, selectivity controls, conceptual clusters and user's specific configuration. It helps interactive computational environments to overcome, "by-design", possible infrastructure limitations in terms of access to external connectivity at runtime by investigating a preliminary solution to enable the pre-analysis of the lineage traces. It takes into account effectiveness of integration via actionable metadata and programmable provenance sensors. The principles and ideas developed in this thesis have been implemented using an existing data-intensive streaming library, `dispel4py` [17] and processing API currently adopted by different disciplines to enable computations in their VRE's [97]. However this approach could be transferred to similar systems based on *stateful* operators that compute any data, with no specific programmable abstractions for sizing the input or for choosing different synchronisation models.

- *Holistic access system.* An archive supported by a collection of services and a suite of explorative and visual-analytic tools havs been implemented. The archive, which is based on a NoSQL document-store, acquires and organises live streams of lineage, while services and tools rapidly repurpose them via user controlled views. Experts can review information from applications running across multiple e-Infrastructures, monitoring and investigating results and behaviours across runs.

Ultimately, the thesis's contributions explore and suggest ways to include new provenance functionalities in systems that are particularly oriented to data-intensive and stream-processing. The pursued approach assigns more value to the participation of the application developers, the domain-experts and the data-architects, for a better production and exploitation of the lineage traces. It shows how user interfaces can use provenance information to repeat and re-configure experiments or to suggest, combine and re-use results into new workflows. This facilitates understanding, rapid contextual-switching and method improvement. This substantial set of contributions was made possible by the interaction with supportive colleagues within projects in the domains of computational seismology and climate impact analysis. This has built a platform, a collaborative research community and research momentum. This thesis is a snapshot of a field that is still advancing.

1.6 Thesis Structure

This work is looking at issues that interest the early phases of the computational and data-driven research practices, leading eventually to the curation of the scientific material. The latter can typically rely on community agreed standards for its description and provenance tracking. This thesis will focus on how human empowered provenance acquisition and rapid analysis could foster better research in the field of data-intensive investigations. It will look at offering interactive systems that make the computational processes reactive and adaptable to the user's priorities based on their improved understanding. Researchers should be provided with ways of reducing the cognitive and technical overhead, preserving an acceptable informative power of the provenance information still produced.

The challenges of *handling of provenance for streaming data* and the *control by researchers of the provenance collected*, will be the main topics covered in the following chapters. Potential cognitive limitations and *ease of use by the researcher* approaching provenance data will also be tackled, accommodating views and abstracting from the complexities of various enactment strategies and computing infrastructures.

Chapter 2 introduces advances in the state of the art for provenance-based solutions addressing use cases in reproducibility and productivity for computational science. It show and overview on the models, the technologies and the methodologies for storing, querying and visualising provenance information.

In Chapter 3 we investigate models of computation and models of provenance, especially targeting dataflow systems, such as scientific workflows or computational libraries presenting streaming and stateful operators. We consider relevant work in the literature concerning provenance models for similar systems and refer to the generic actor model for the concrete execution of the workflow. Then we exploit and extend existing provenance representations (PROV and ProvOne) adding explicit semantics to capture workflow's runtime characteristics (S-PROV). These include aspects of delegation and management of the actors' internal state. We add concepts to capture dynamic scenarios, where a distributed workflow application may change the behaviour of its components at run-time, for instance by means of relocation, implementation

and re-parametrisation.

Chapter 4 introduces the conceptual and technical framework enabling tuneable and actionable provenance in data-intensive workflow systems. It introduces the concept of *Agile* data-intensive systems and shows a novel approach to the integration of provenance mechanisms. The contribution aims at offering flexibility in the scale and in the precision of the provenance data collected by streaming operators, assuring its relevance to the context of the scientific domain, as well as the user's interest.

In Chapter 5 we describe the `S-ProvFlow` system. This consists of a set of components to support storage, interactive exploration and visualisation of the provenance information characterised by the S-PROV concepts. It includes a NoSQL document-store MongoDB as a provenance database, a service layer in the form of a RESTful Web API and two interactive provenance exploration tools. We show how the lineage can be explored in different layers, from the detailed validation of the data transformations to comprehensive perspectives, also covering multiple runs.

Finally, in Chapter 6 we will discuss the implementation of the system for two domain specific VREs, respectively dedicated to computational seismology (VERCE) [80, 78], see Figure 1.1, and to climate impact studies (CLIPC) [10]. We will describe the services with more detail, discussing the experience of implementing provenance-aware mechanisms and their role. We will report what we observed during seismology training and the feedback collected in a dedicated workshop, where experts in implementing climate services were exposed to technical solutions.

The thesis will end with a summary, which will highlight conclusions and envisaged future work. The latter will be pursued in the context of new projects that will build upon and extend the results of this thesis and that will further develop its components.

Chapter 2

Background

This chapter provides information about the state of the art of handling provenance data addressing several use cases in support of data driven and computational research. They constitute a large source of challenges and scenarios that stimulated the work pursued in this thesis.

2.1 Supporting Data-Intensive Research

Today many scientific achievements are obtained by combining and analysing large amounts of data coming from simulations and sensors. This brings a *data deluge* [181] that hits the limits of the facilities available at research institutions and may overwhelm the cognitive powers of researchers. Moreover, both types of data often need to go through a number of tuned transformations before being able to show the evidence for new findings. This presents new computational challenges and provokes the re-thinking of the design of HPC systems and the software which exploits them. Aspects to take into account include usability, access models and, to some extent, also solutions for low-impact energy consumption of these expensive machines which, according to recent work, could be independent from application tuning [220]. The ultimate goal is to obtain a better sustainability and the productive usage of these systems from an increasing variety of scientific users, who are going to produce a wealth

of data that will require adequate data curation plans [154]. Such scenarios began to interest a growing variety of scientific disciplines and fostered the birth of a new research branch in information technology, known as *eScience* [155]. A survey on the most typical characteristics associated with handling, managing and interpreting provenance in *eScience*, was conducted in 2005 by Simmhan *et al*. [211]. Today, this is still considered a relevant topic of discussion, especially for its actual implementation within the common research and data-management practices of specific research domains.

As mentioned in Section 1.4, the recent establishment of an international working group cataloguing and discussing patterns and use cases which would benefit from a formal provenance description, demonstrates the growing interests on the subject and the global awareness of its importance. The book "Fourth Paradigm" [153], clearly defines this new model of scientific exploration, where challenges brought by large-scale simulations and observations need to be supported by an effective data management strategy. The one-size-fits-all tools for all domains can not be feasibly pursued, but the focus should be moved towards domain-specific variants of generic tools often mapping onto shared middleware systems that integrate relevant computational, storage and data movement resources. These differ to be well adapted to the sub-tasks they support. Leveraging from more generic practices and tools is crucial and these must be tuneable for the domain's most important issues, following an engineering approach. This has not been considered as a priority until very recent times. Handling data in this context requires us to support efficiently different phases of the data life cycle, which includes *simulation, capture, curation and analysis*. It should aim at providing a comprehensive data-exploration framework which can extract value from such a wealth of information, besides supporting reproducibility, as a service.

2.1.1 Reproducibility

In recent times, in the field of BioInformatics, as reported by an author of the Fourth Paradigm, Tony Hey, in one of his recent talks at the 3rd National eScience Symposium in Amsterdam [2], the impact from the lack of reproducibility is affecting more

than 60% of all the biomedicine studies. Such evidence has been produced by major pharmaceutical firms including Bayer and Agmen. Some of Agmen's scientists, for instance, could reproduce only 7 out of 53 cancer results published in Science and Nature. This shows the enormous impact that the lack of practices and technologies supporting reproducibility can have in cross-cutting studies, which affect the quality of commercial products, the quality of science and ultimately, the uptake of significant research results in modern society.

In Solid-Earth Science, for instance, several scientific breakthrough are only possible via the processing of millions of hours of data calculations and the availability of an extremely dense network of sensors. A recent study of the relationship between the plumes of hot rock coming up directly from the Earth's mantle and the formation of volcanoes, in areas such as Hawaii, Iceland and Samoa [137], made use of large datasets consisting of seismic waveforms and synthetic elements. In this paper, the methodology is described in a relatively high level of detail, including the adopted parametrisation of the processing steps applied to the data. However, no lineage information or metadata samples are made available to show how the processing pipeline affects the data, favouring instead more attention on the final results and the computational challenges. Although this may be justifiable by the scope and space constraints of this type of publication, very few details or references are provided about the data management procedures that were in place, suggesting that substantial *ad-hoc* and probably tedious and uninteresting work is still required. The only thing we get to know is that IRIS [30] is the repository supplying the data but the subsequent stages are unclear.

Reproducibility is one of the reasons why workflow engines and systems built on top of models inspired by DAGs (Directed Acyclic Graphs) started to gain momentum in science [90]. Looking at contemporary data-analysis frameworks, such as Apache Spark [4], reproducibility is obtained at a very fine-grained scale within operational scenarios. Spark has the ability to recover partitions of its distributed resilient datasets (RDD) [224] if failures occur in the worker nodes. It relies on the lineage information inferred from the description of the user's script, which is then transformed into a physical execution plan that can be re-executed for recovering purposes. Although

this approach to reproducibility is effective to guarantee resilient and fault tolerance mechanisms, it can not be easily used for cross platform and long term reproducibility purposes and validation. Especially the latter would require to extend the lineage information with rich domain metadata. Recent work discuss the integration of a data provenance capturing mechanism for Spark, especially tackling debugging use cases at an affordable overhead [161].

Systems like VisTrail [82] carry the flag of reproducible science, thanks to its approach to the management of evolving scientific workflows based on the generation of prospective provenance data, telling the history of the variations in the workflow's structure. It keeps track of workflow changes involving both its parameters and its specification, supporting exploratory computations. However, this may not be sufficient to reproduce the results when the application is handed over to third parties or even when it gets used again after a long period of time. In such scenarios it is fundamental that the results can be evaluated based on the evidence brought by previous executions, which includes domain metadata and information on the computation environments used. It is necessary to identify with enough detail the code used for the implementation of the scientific methods and its dependencies, such as the set of libraries, interpreters and all of the sensible information associated with it.

Curated catalogues of software, virtual images or docker-based containers [18], should be maintained by institutional organisations offering PaaS, for instance like the EGI AppDB initiative [20]. Referencing to these sort of services would already be way to assure that operations could be reproduced in the same or at least on emulated environment. Processes and data must described by precise metadata, including scientists' annotations. Annotations can be extremely helpful since they represent an extensible and flexible way of describing those features which are tightly coupled with the kind of investigation pursued.

Reproducibility can be supported by the combination of prospective and retrospective provenance, the former describing the workflow variations and the latter showing the progressive effects on the data. These information could also help with the identification of those components and settings that might be automatically imported and reused within a new computation and to discover relevant reusable datasets. Recent

work has introduced the concept of *Research Object* to illustrate a framework that builds a semantic container around a research artifact [102]. The container manages and establishes relationships among all of the provenance information, including methods, abstract workflows' descriptions and support versions. This facilitates the process of handing the results to new investigators, which access the artifact in the form of a *Live Research Object*, which is a copy of the original and persisted instance. Such frameworks should be aware that within the lifecycle of a experimental product obtained by a a well defined scientific workflow, all the datasets, constants or sources of standard constants, databases acting as inputs are also subject to revision and curation [110].

2.1.2 Towards Provenance for Productive Data Science

Mattmann identifies four research tracks as critically important for future data science [182], based on his many years of experience at NASA and at the Apache Software Foundation. These consists of *Rapid scientific algorithm integration*; *Use of Cloud Computing*; *Harnessing the power of open source in software development for sience*; last but not the least, *Intelligent data movement* during the computation. This is key to meeting operational constraints and to achieving performance. For instance, HPC jobs can't be seen as closed systems, but rather active entities which can trigger behaviours and state changes in external services while they run. Obviously, digging into the depth of experimental processes would be beneficial from a user's perspective if the retrospective provenance produced would activate tools and complementary operations within her domain, automatically supporting established and unforeseen metadata terms and structures. This is an important aspect covered by this thesis and sees the timely growth of the debate, which is also supported by preliminary work on the role of domain metadata within provenance databases [124, 139].

The *Rapid scientific algorithm integration* and the *Use of Cloud Computing* are two aspects that are becoming fundamental for the support of data-intensive science. Scientists and data analysts develop scripts which fit their purpose and are often self contained, written with no attention in atomicity and modularity. Although this ap-

proach may not be suitable for massive data-intensive computation, it may still present data ingestion and computational requirements which could be large enough to justify the need of being executed in a performant cloud environment. Moreover, the data of interest may be subject to relevant privacy and copyright policies, which prevent their download, therefore the computation has to be moved to where the data resides. These scenarios open new challenges in provenance management which go beyond data-intensive research. In their support, interesting recent approaches, called *no-workflow* and *yes-workflow* [194, 128], try to perform the extraction of provenance from users' scripts respectively by relying on the automated analysis of the Python interpreter's function-calls stack, or, in the case made by *yes-workflow*, leaving this task in complete control of the developer, providing a full range of inline annotations. Both approaches may apparently satisfy the need for a *Rapid scientific algorithm integration* which also includes the support for provenance and can be used within the cloud. Especially the *no-workflow* approach, underwent experimentations to include it within cloud enabled deployment of the iPython notebook [201]. On the other hand, while *no-workflow* overlooks the importance of sensible domain metadata, tracking very low level traces, *no-workflow* does not take into account volatile data associated with intermediate stages, inferring the the retrospective provenance associated with files and it relies on the researcher who should use the annotations consistently [122].

Scientific workflows aim to ease the execution of the research practices within a unified and sustainable framework, supporting data analysis across heterogeneous computational models and infrastructures [148]. Provenance has a key role in enabling an effective user-centred communication of the workflow execution phases and results, as well as fostering an efficient enactment of the processing. Though, different disciplines may express specific requirements which, in a way, drives the development of the functionalities of a workflow management system. In geoscience, specifically in meteorology or for early warning systems in seismology, the requirement for accessing continuously collected data by sensors imposes the need to process data streams in near-real time. In this scenario, the processing component should be able to adapt to special conditions where data packets can be dropped, networks can go down and the data can be subject to wrong reads over the wire. This adaptation can lead to inaccuracies which might influence the decision-making. Provenance can provide support

for a partial solution to this problem [218] by exploiting low-overhead tracing techniques, to monitor the behaviour of the processing elements and the anomalies that might be encountered. Eventually, the adaptations of the processes should be captured to recognise, evaluate and filter the anomalies which may have occurred.

Besides the important reproducibility problem and the provenance gathering techniques illustrated so far, it is worth considering on a number of use cases that can be addressed by using provenance, leading towards a productive data-science.

Iterative validation: Scientist often need to execute partial runs of their workflows to tune the parameters of the components involved in their computation. These runs can be considered as part of a preparatory phase that will lead eventually to a the execution of the workflow on the whole dataset. Recording provenance traces about the intermediate and fine-grained steps, which may refer to volatile data described by runtime user-defined annotations can provide support for this preparatory phase. The SEAD project, for instance [195], encourages the storage of intermediate research datasets, offering the possibility to preserve and curate experimental results from the very beginning of the scientific investigation. It does assign to these kind of data different maturity tags: 'live', 'in-curation', 'published'. In this context, a flexible and tuneable provenance system could be used to handle, store and share 'live' products, within an active research group. Here, useful information can already describe processes and data before committing a complete and comprehensive provenance trace for the finite and published result. Processes could be organised into temporary composites made by unrevised code, and traces may be produced in high detail, but only for those data characterised by a selected set of properties and values.

Site usage and failures characterisation: In the case of system failures, we would expect the workflow management system to take over, providing feedback to the user when any attempt to recovery has not succeeded. On the other hand, recovering can be extremely complicated in streaming workflows, because the data is most of the time volatile. Streaming systems would require to maintain intermediate buffers or replicas to guarantee loss-less failover scenarios [173]. Indeed, loss-less solutions must take into account whether the high-availability

policy of the infrastructure can justify the overhead brought by an expensive recovery technique. For example, a more economic recovery strategy based on checkpoints and provenance analysis [119], would introduce a sort of user-driven declarative approach, which reacts to those failures which are located in pre-selected sections of the workflow. If logical errors occur within the analysis code of a delicate workflow component, the effects could be recorded and notified during the execution. This solution coupled with the access to the provenance data at runtime, could prompt the user to terminate the computation. This can prevent the useless exploitation of the resources and avoid unfruitful waits.

As for many other monitoring tools, provenance can record, besides errors, also performance metrics. An interesting publication illustrates the adoption of provenance information [179], to perform statistical analysis of the recurring failures, by correlating resource exploitation with workflow executions. This approach could lead to the redistribution of the computation that could foster improvement of performance as well as decreasing the probability of the failure recurring.

The main advantage though of using provenance data to analyse the behaviour and the usage of an infrastructure, is mostly related to the fine granularity of the information (in many programming languages a process can consist of multiple threads performing specific tasks), which explicitly refer to a specific execution context, characterised by well described data and execution plans (the workflow specification and its components).

Data curation and citation: Among the commonly agreed practices to deal with data curation and data preservation, there is the attribution of a Persistent Identifier (PID), which can be used to access and reference datasets, for example, within a publication (coarse-grain provenance). That being said, it seems obvious to recognise also workflow provenance (fine-grain provenance) as a crucial source of information which definitely contributes to characterising the referenced data. The theory and practice about how this should be done is currently under investigation in several research groups and international projects, for example, *Wf4Ever* [86] provides best practices, formats and tools to preserve research

data and provenance within a unified interoperable framework.

The attribution of the PID to a dataset and its granules is often performed according to community requirements, which often depend either on the nature of the data or on the users' practices. For instance, the policy for the assignment of PIDs to quality-checked time-series produced by real observations, might be different from the one applied to experimental and finite datasets, obtained by a simulation. The first kind of data could grow with time and could be reprocessed in order to apply corrections and gap filling. While its growth doesn't necessarily affect the features of the data (besides the time coverage), reprocessing the dataset might affect significantly the results obtained by its usage, requiring therefore some sort of version management control.

Whether PIDs should also be associated with semantic information describing the characteristic of the data-source, for instance, as suggested by the PID Information Types [219], or whether they should be completely agnostic from the type of resource they identify, such as the DOI approach [74], are challenges which are populating the agendas of dedicated international working groups [62, 12]. These efforts aim eventually to find commonalities among research areas, in order to promote consistency and interoperability.

Composition and Reuse: In any collection of independent procedures, which may have been implemented with different workflow management systems or software packages, interoperable semantic provenance models coupled with common harvesting mechanisms can foster the automatic production of the information that can be used to discover and ingest the data produced by these independent systems [188]. This can be coupled with the semantic constraints that can be specified about datasets and components, as allowed for instance by workflow systems like WINGS [85], that may be coupled to semantic registries, as presented in the Data Bonanza book [98].

Collaborative Environments: Virtual communities, such as `myExperiment` [38], allow scientists to share and discover new workflows that can be imported and modified, promoting in a way the improvements and the collaborative identification of common reusable fragments. It is also important to consider aspects

associated with the communication of research progress aiming at a better understanding of scientific collaboration and its effective realisation and in support of long-term curation. Recent work provided an overview and comparison of semi-automated and manual approaches to the generation of abstractions, aiming at the production of *report-worthy* summaries from the provenance traces [92]. They found that automated abstraction systems are skewed towards the description of the processes, overlooking the data.

In respect to collaboration, with the growth of scientific communities advocating open science, sharing scientific workflows and their execution traces as web resources becomes the means to facilitate their access. A *Linked Data* approach has been proposed [141] that defines a collection of requirements and a methodology that consider the specification of conventions and metadata. However, how the methodology and the technical solutions could be used in the early stages of workflow development and evaluation is a topic for further investigation. Web formats combined with a layered approach to provenance representation will be investigated in this thesis. We will address use cases involving rapid exploitation of experimental runs, such as monitoring and combined integration of workflows experimental results. These are supported by high-level services for querying and visual exploration.

2.2 Provenance Models

Data provenance is a widely known and discussed research topic, which presents different challenges across several computational models. Recent effort has been dedicated to the definition of standardised and unified representation models for provenance information, such as the Open Provenance Model (OPM) [191]. This high-level specification, together with its most recently community agreed extension W3C-PROV [83], provides a means to compare the conceptual coverage of more specific provenance representations, which typically are coupled to a particular computational model, a domain specific semantic or a workflow engine implementation. Such models can be therefore extended to match scenarios which are more specific to a certain community or a type of analysis that a user or a group wants to perform on the traces.

These extensions can also be referred as provenance *profiles* and aim to improve interoperability and understanding in the context in which the traces are produced.

The provenance information that can be collected in scientific workflows can be divided into two classes, prospective and retrospective. Prospective provenance collects information on workflow specifications, while retrospective provenance aims at describing the data derivation relationships and processing metadata within workflow executions [122]. The coverage of these two complementary representations of the scientific computation is a well known challenge and opens up more research investigations. Since the release of the PROV data model and its conceptual framework, further specialisations have been presented to support different application domains and computational scenarios. A recent activity promoted by the DataONE [16] project, proposes an extension to PROV in order to represent the most common workflow structures. ProvONE [52] aims at providing a general framework to represent prospective and retrospective provenance. PROV-Wf [118] instead, represents coarse-grain information about the runtime features (retrospective provenance) involved in a workflow computation, proposing mappings from the traces produced by different systems in different formats. Other projects contemporary to the progress and specification of PROV tried to address similar challenges. The project `Workflow4ever` [86], defines ontologies aiming at a more generic representation of a workflow-driven computation, including the creation of shareable containers, *Research Objects* [101], providing a unified framework to manage data and provenance annotations.

2.3 Storing and Querying Provenance

The variety of representations for provenance information triggered several studies on the optimisation of provenance queries [189] for *black box* workflow components that offer primitives for the manipulation of collections, such as Taverna [198] and Kepler [178], and on the identification of query patterns [140] in multi-task scientific computing. The introduction of the aforementioned unified models, inspired new investigations on the implementation of more general purpose storage strategies. Initial research has been conducted on the evaluation of the adoption a relational databases

for OPM, adopting SQL query languages, in order to measure the performance obtained by the adoption of standard querying techniques [172]. A follow up to this work presents a more OPM specific query language, OPQL [171]. OPQL aims at expressing provenance queries, including lineage and data derivation queries, adopting graph patterns and a sound OPM-based algebra. This approach will save the user from coding complex recursive queries, hiding moreover, the details of the adopted storage strategy.

Provenance models for workflow systems are commonly expressed as a Directed Acyclic Graph (DAG), connecting in many cases also artifacts like user annotations and configuration parameters, defined by flexible vocabularies [139]. This typical representation provoked a natural curiosity for the exploration of alternative and recent storage systems such as graph databases, for instance, Neo4j [40]. This solution has been used for the implementation of the *PBase* provenance system [121], which stores and query traces according to the ProvONE model. It enables queries on workflows traces that were previously uploaded onto the system in VisTrails XML formats. The interrogations include lineage and execution queries specifically associated with the relationships between data and processes, and with search capabilities focusing on the involvement of processes within runs. Recent work [217] has conducted benchmarks to compare standard query performance over DAGs stored in MySQL and Neo4j. The results show an overall gain in performance of the graph database in answering structural and full-text queries, acknowledging on the other hand, a more efficient response of MySQL when dealing with numerical types of queries. In this thesis we decided instead to experiment with a well established document-store, MongoDB [35], tying to give priority to use cases that require accessing the provenance information involving data properties and processes' parameters described by domain and user-defined metadata terms. Interesting integration of the two technologies are currently available [39], aiming at combining the flexible schemas and indexing possibilities of MongoDB with the performant graph traversals offered by Neo4j, suggesting new research for polyglot database solution addressing provenance scenarios. Our use cases and their engineering are presented in Chapter 5 and Chapter 6.

2.3.1 Centralised vs Distributed Provenance Archives

A computation is typically distributed, not only across the nodes of the same cluster, but also across different computational models (*e.g.* HPC or Data Intensive) and institutional infrastructures. In the latter case, security and storage constraints may interfere with the possibility of updating and querying consistently multiple provenance stores as the computation proceeds. Though, the adoption of a centralised provenance archive, should not impose a data transfer to a remote repository, to the extent that it significantly reduces the rate of processing, wasting precious and expensive CPU time in the HPC case. These scenarios suggest the need to investigate strategies to balance the trade off between the availability of consistent provenance views at runtime and its efficiency. Existing approaches, which also support heterogeneous workflow systems, map distributed provenance archives into centralised storage, translating the information into interoperable formats [118] such as PROV. On the contrary, systems like SPADE [143], investigate distributed storage solutions, introducing the concept of *Network* artifacts, called *Connections*, in order to represent, in local databases, the provenance of the transmission of data across different logical or physical nodes of the network.

2.4 Harvesting Challenges

Recently, the growth of the size of the data and the processes involved in scientific computations, led to the rise of new challenges for an efficient harvesting and preservation of the provenance data. For instance, with the introduction of streaming workflows, the data ingestion phase typically requires the raw data files to be sliced into many small and volatile data chunks, leading to the possible explosion in the number of the provenance traces produced. Moreover, these traces are often describing unpreserved and therefore not reusable artifacts. A better understanding of the streaming system could help in identifying efficient caching strategies to support smart workflow re-runs or capturing the propagation and effects on the data after, for instance, the re-parametrisation of concurrent copies of the same component. Below, we introduce two

usage scenarios that present interesting harvesting challenges that have been addressed during the progress of the thesis.

Runtime provenance: Provenance data should be accessible at runtime, during harvesting, in order to allow for immediate interaction by users who can then monitor the production of the results. This approach is useful, for example, when a preliminary evaluation of partial results might suggest actions to be taken before the termination of the computation. Recent work [118, 117] tackle similar problems respectively within heterogeneous workflows environments, proposing a distributed harvesting infrastructure. They focus on the adoption of Chiron [197], which seems to be one of the few existing workflows providing provenance data at runtime.

Selective provenance: Provenance systems generally collect data about transformations [135], and in a streaming workflow system, the high rate of the production of these information could present the risk of overwhelming the system. In those circumstances where provenance gathering may be expensive, either in terms of size or I/O time, users should be allowed to define at workflow level the set of processes to be traced or, for instance, the properties of the data-stream that they might want in the provenance trace and its precision in respect to the occurring dependencies. In order to reduce the production rate of the provenance data, some works suggest the extraction of the traces from *Virtual Processing Elements* [187], obtained by clustering more processing steps into macro activities, aiming therefore at the definition of a higher level logic. Another previous work [125], proposes the recording of provenance data only for portions of the workflow, supporting the argument that the generation of huge quantity of trace data can be excessively demanding for an effective visualisation tool, provoking a cognitive overload for the analysis of a realistic application.

Once provenance has been produced and stored, its exploitation can also be enacted through effective and interactive systems and visualisation techniques. We will provide an overview on the relevant state of the art in the next section.

2.5 Approaches to Visualisation

This section presents an overview of common and innovative approaches for the visualisation of provenance data. We have introduced so far several challenges associated with describing, harvesting and storing large provenance traces. However, the real value of provenance in the researcher's practice comes with its exploitation. Effective graphical applications can engage groups of scientists, data providers and systems engineers, highlighting trends in the usage of the data and resources, or revealing dynamics which are internal to the workflow's execution. The distinction between the two concepts of retrieving and revealing information is as subtle as it is important. What visualisation techniques should aim for is providing a variety of perspectives over the provenance landscape, unravelling the dense bundle of paths that characterise the relationships between interconnected and heterogeneous players. Many meta search engines exploit contextual information provided by the user to retrieve valuable results, like *Inquirus* [167], or more recently, the IBM's *Watson* system [133], which are thereby applied to several domains. Visualisation in provenance should also exploit context awareness enabling the value to surface rapidly from the users own collection of provenance traces, merging domain metadata as well as experimental descriptions.

Interactive tools built on top of intuitive visualisation methods may have short and long-term impacts in the research lifecycle. In the case of data-intensive engineers, comprehensive views including low-level detail about the underlying resource, may reveal odd trends, stimulating the investigation of a different distribution of the computation within the target infrastructure. While in-depth visual probes, focusing on small sections of a computation and on a few properties of a large data-stream, could bring to the surface inconsistencies in the behaviour of the workflow components or in the interactions among them. Issues related to the data may require mechanisms to push the forefront the most relevant entities, according to metadata values' thresholds, while running processes may produce different messages that would be visually encoded. The identification of visual patterns occurring within a graphical representation of the provenance data, could help orientate the users across different application contexts, rapidly shaping their understanding of the usage of data and processes. This sort of aid can support collaboration and the re-use of workflows' results across scien-

tists, especially if combined within contextual searches, as previously mentioned.

Therefore, flexibility in the granularity of the visualisation and relevance to the user's context play together a determinant role. *Map Orbiter* [208], for instance, tries to produce summarisation nodes which can be expanded on demand, allowing navigation across control-flow and data-flow relationships. It offers filtering functionalities and an integrated view which superimposes a time-line on the process tree, highlighting chronological relationships across all of the scripts and the sub-processes involved in the computations. In the long-term, the size of provenance data produced from data-intensive analysis could be overwhelming and not all of the provenance may be relevant for preservation purposes. Supporting a user-controlled re-composition of sub-traces can foster the identification of the provenance data that better represents and describes the crucial aspects of the computation, possibly discarding the surplus. Another approach similar to *Map Orbiter*, tries to reduce cognitive overhead proposing a provenance representation model which is inspired by the functional programming paradigm [111]. It allows navigation through nested structures which are determined by the hierarchy of function calls. These can be expanded on demand across all of the items of a potentially unlimited list of inputs. The *Provenance Explorer* [158] offers to users functionalities to visually compose summaries consisting of representative traces, which are extracted from the broader collection of provenance data. In respect to common provenance models and scientific workflows, a proof of concept user interface visualising multiple execution traces expressed in ProvONE was developed for the *PBase* system [121].

The works just mentioned, take advantage of the most common representation of provenance data, which is characterised by a Directed Acyclic Graph. Alternative techniques are emerging. For instance, the *Sankey* diagram has been adopted by *PROV-O-Viz* [156] to represent magnitude of flows between activities entities in a PROV document. Other solutions instead map the graph, or some of it, on to radial diagrams [129]. This technique has been explored for text data visualisation [206] and in other fields, from genomics data [166], to the analysis of systems which make large use of parallel I/O [210]. *InProv* [106] pioneers its adoption for provenance visualisation in the context of the recordings obtained by PASS (Provenance-aware Storage System) [45, 193].

It proposes a time-dependant distribution of provenance entities, like processes, files and communication pipes, on a radial visualisation. It connects these entities according to parent-child relationships constructed from control-flow information. The aim of this technique is to reduce the visual clutter which is typical from an excessively flattened node-link visualisation, by bringing the most important nodes to the forefront. Computer graphics research has focused on the advantage of this radial distribution of interlinked information and ways to improve the visual efficiency and tuneability of such representations have been presented in literature. For instance, the *Hierarchical Edge Bundles* [157] method, which can be used in conjunction with existing tree visualisation techniques, aims at reducing visual clutter of highly connected structures via the generation of bundles, whose bundling strength can be tuned to allow low-level and high-level views on adjacency relationships. The technique has been experimentally adopted within the *ProvStore* [160, 53], for the general representation of provenance relationships extracted by static PROV documents. In this thesis, we have also experimented with the same technique obtain from configurable queries over comprehensive views about single computations, as well as larger campaigns involving multiple runs. These are performed by different users across distributed resources, and in the context of large amount of data and metadata. Finally it is fundamental to have tools that can scale visually over large collections, and possibly between devices. Detailed views are as important as comprehensive summaries, and ways to switch easily between these exploratory modes have to take into account humans' cognitive limitations, guaranteeing a level of consistency between them, to avoid confusion in the interpretation of the semantics of the items involved.

Chapter 3

Model of Provenance for Data-intensive Computation

This chapter focuses on the analysis of the fundamental *observables* of data-intensive systems with special attention to data-flow and streaming workflows composed by stateful operators. It describes a model of provenance that combines abstract and concrete representation of the data-intensive application and discusses the properties of the model with respect to the workflow's logical and runtime characteristics. The contributions, listed below, address modern stream processing systems [196, 203] and cross-platform abstractions [134], where logical operators are parallelised and have an internal state. The operator's state can be managed automatically or explicitly by the developer and is dependant on the history of previously processed tuples [95, 116].

(C-2) Provenance for data-intensive systems: Analyses the properties of data-intensive system and its associated model of computation (MoC). The corresponding *observables* are presented as the key elements of a schematic representation of the provenance model (MoP) that makes use of and further specialises PROV constructs. This chapter describes the properties of the model, especially those aspects associated with stateful operators and dynamic steering [183].

(C-4) Support for multiple levels of understanding: Given the schematic description of the model and the formalisms, this chapter illustrates how it enables

contextual switching. From technical details and concrete execution plans, to high-level abstractions, supporting different classes of users and usage scenarios.

Overall, this succession of models takes us from abstract computation to consistent provenance that might be analysed, interrogated or visualised, as discussed in subsequent chapters.

3.1 Data-flow models

We identify the most general model of computation that can represent the target system, and then introduce further specialisations to represent its properties. The following sections will illustrate the concepts which set the basis for the model of computation (MoC), leading to the description of the provenance model (MoP).

Generic models describing synchronous and asynchronous exchange of information among processes within a network have been introduced in the literature and provide a solid basis to start our investigation. The Kahn process network [163, 170] model assumes unbounded communication channels, where *non-blocking* reads can be enforced. This model describes the continuous ingestion and production of messages, or *tokens*, by the nodes of a network. It guarantees determinacy and the preservation of a partial order of the tokens across the whole system. Though, it does not cover additional causal properties that can be observed or regulated with the occurrence of read and write events within a specific node. A useful specialisation is the *data-flow* model with *firings* [168]. It considers networks as directed graphs and associates the concept of *firing* with the invocation of a process, that ingests, evaluates and eventually produces data. A *firing* is characterised by a number of consumed *tokens* and the subsequent production of new *tokens*. The number of produced tokens can be different from the number of the ones ingested. Data-flow models can be further specialised to include more restrictions in favour of determinacy, meaning that for a given input sequence, all resulting sequences are uniquely determined. For instance, a more restrictive specialisation of this model is the SDF (Synchronous Data Flow) [169], where

processes read a fixed number of input tokens before producing a fixed number of output tokens. Both the *data-flow* model with *firings* and SDF guarantee determinacy. In data-intensive applications, this restriction limits the possibilities for parallelisation, therefore, as we show in the next section, we need to address a computational model that allows for more flexible scenarios.

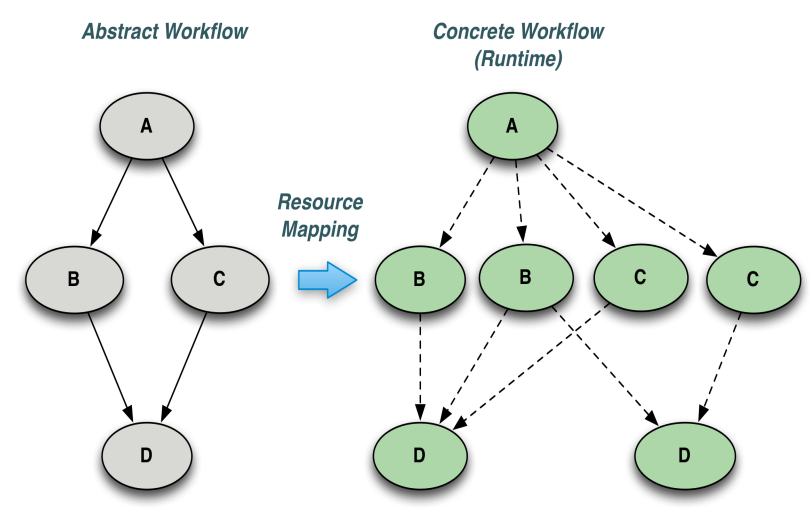


Figure 3.1: Abstract and concrete topologies: The left side shows an abstract workflow structure as composed by a developer, while the right side shows its actual representation in the system, where more actors run on behalf of each abstract component. The transformation from abstract to concrete is typically know as *resource mapping*. Dotted arrows represent a possible routing of the data between actors.

.

3.1.1 Determinacy and the Actor Model

The design approach of scientific workflows systems that consider an Actor-oriented models are extensively discussed in the literature [107], especially for what concerns the realisation of the Kepler system. Here, determinacy is a desirable characteristic that is enforced by the data-flow models mentioned in the previous section. In this work we address data-intensive systems that support parallelisation of user-defined workflows at the level of the single workflow component. In Figure 3.1, this is illustrated by the translation of an abstract workflow, defined by a workflow developer

adopting a high-level language or API, into its concrete representation, which is distributed across a pool of available computational resources. This translation phase is known as resource *mapping* [127]. A tool which offers such primitives especially designed for data-intensive streaming processing, is the *dispel4py* system [134]. Here, in the executable workflow obtained by the abstract definition, each of the original components, or processing element (*PE*), are mapped to multiple instances that are executed concurrently. If we assume the independence of each instance, this scenario could be representative of the *Actor Model*, according to the interpretation provided by Agha [89]. Here, instances can be considered as *actors* that, according to the model, these are self-contained, concurrently interacting entities of a computing system. The model can be used to obtain a wide range of computation paradigms. It allows for dynamic topologies and enables token switching among adjacent *actors*, with the associated side-effects. For instance, in a more concrete realisation of the *Actor Model*, where systems process and re-distribute continuous data-streams, this may result in unbounded nondeterminism, since the arrival and the order of processing of each token by a target actor can not be determined. This invalidates the principe of monotonicity, which may again result in nondeterminism.

In Kahn processes [163, 170], monotonicity is guaranteed. Two *actors* sharing the communication channel can therefore process and exchange portions of the data stream in parallel consistently. However, in flexible and scalable systems inspired by the Actor model the number of *actors* serving the same purpose may change at runtime, making use of different switching policies among them. This does not guarantee that the tokens are processed by the workflow component as they are produced, neither that the results are returned in the same order, as we exemplify in Figure 3.2. In some circumstances, in order offer scalability and consistency of the results, modern systems, for instance *Apache Storm* [5], allows the developers to specify grouping rules within the definition of the workflow's component, instructing consistent redirections to its instances at runtime. The grouping rules are based on the data and metadata properties and foster deterministic matching and merging operations, bending the *Actor Model* towards a stream-oriented implementation of another popular computational model, known as *map-reduce* [126]. However, when capturing the lineage of an streaming operator that is subject to grouping rules, we also need to take into account that the data of

the same group arrives interleaved with data of other groups that have been assigned to the same instance. These settings require developing the component as a *stateful* operator, which involve saving and accessing intermediate stages of the acquired data. From the provenance perspective, we represent this, as well as other *stateful* scenarios, by associating with such operators an internal provenance state. This will be covered in detail in Section 3.2.2 and in the Chapter 4, as part of the proposed provenance model and capturing framework, which allow us to record grouping and other complex lineage patterns.

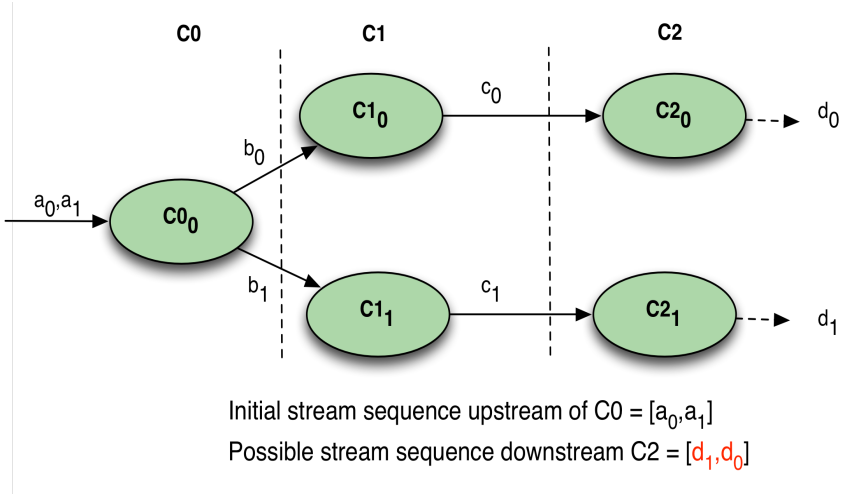


Figure 3.2: Non-monotonic computation in a pipeline of three components $C0, C1, C2$. The data received by the component's instance $C0_0$ is processed and passed to the instances of $C1$ and $C2$. If we assume no constraints on the sequence of acquisition and processing of the data by the components' instances, the scenario where $a_0 \prec a_1$ upstream of $C0$ and $d_1 \prec d_0$ downstream of $C2$ shows, from a components' perspective, a non-monotonic computation.

3.1.2 System and User-driven Observables

The Kepler group [176, 108], pursues the definition of a provenance model for the Kepler system, by introducing the concept of *observables* which relate to the occurrence of *firings*. They consider that *observables* can be associated with more phases of a *firing* depending on how the system works and the required level of detail. Each actor that fires, is characterised by internal mechanisms and logic that can also be modelled

and captured as a relevant *observable*. These could include its location, the workflow component it relates to, the runtime details associated with its invocations or the preservation and management of an internal state. The definition and capture could be completely automated or user-driven. In principle, an ideal provenance system should be capable of mediating between automation and user intervention, accommodating system and domain specific details in a single framework.

Definitely dynamic are the information related to data dependencies and these could be classified as system and user-driven. Trivial dependencies between the incoming and the produced data involved in a single *firing* can be easily identified and captured. Though, as mentioned in the work of Ludaescher *et al*. [176] and Bowers *et al*. [108], it is useful to expand the concept of *firing* to a wider scope, in order to take into account data dependencies which can still be valid across many invocations.

Ludaescher introduces the concept of transactions as formed by a group of invocations (or *firing-round*) where any output data can be considered in a dependency relationships with all the input data that previously occurred, up to an initial condition, see Figure 3.3. If we consider a system that provides hourly an average of the day's incoming values on a 24-hour period; each output will be dependent on all of the previous inputs, up to the beginning of the day. Given this scenario, the actor's *state* is defined as the entity responsible for keeping track of all the tokens within the transaction. The re-establishment of the initial condition is obtained by a *state-reset* event [108], which can be invoked automatically, according to some default system policy, or instructed by the developer who retains the knowledge of the component's semantics.

As previously mentioned, an actor's *firing* takes into account also the actor's internal behaviour. The actor may establish data dependencies by accessing and using alternative resources, like filesystems or webservices, or via the manipulations of the incoming data elements. To guarantee the consistency and usefulness of the provenance information, these should be also traced, if relevant. We show how extending the *observable* characteristics of the actors, also in respect to the management of their *state*, fosters a holistic provenance representation that accommodates hybrid workflow systems that support multiple ways of accessing and manipulating data. This offers the

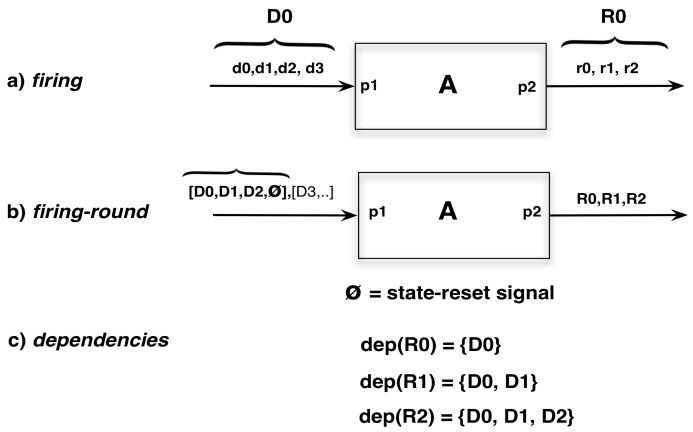


Figure 3.3: Image (a) shows a *firing*, which is characterised by the ingestion from a process A, of a number of input data (tokens) before writing outputs (represented in the figure in their order of arrival and production, *e.g.* $d_0 \prec d_1$)). In (b) instead we show a *firing-round*, which consists of the sequence of many firings until the occurrence of a *state-reset* signal ($\emptyset$). Finally, the image shows (c) the dependencies associated with the outputs of some of the firings.

.

possibility of increasing the granularity of the lineage as needed, besides its precision, by allowing it to capture dependencies that otherwise would be ignored and not represented, as well as preventing the production of less relevant lineage traces. Ultimately, we aim at improving the usability of the provenance traces with respect to the context of its application, infrastructure and implementation choices, with a minimal and, to some extent, re-usable contribution from the developer.

3.2 S-PROV: Resource Mapping, Stateful Operators and Dynamic Changes

As anticipated in the previous sections, we consider systems where the data-intensive applications are specified with a high-level language allowing users to define abstract, machine-agnostic fine-grained workflows, such as *dispel* [96]. The *Actor Model* [88] is then our reference model of computation for the concrete mapping and execution of the workflow onto the underlying resources. This can be implemented by a system,

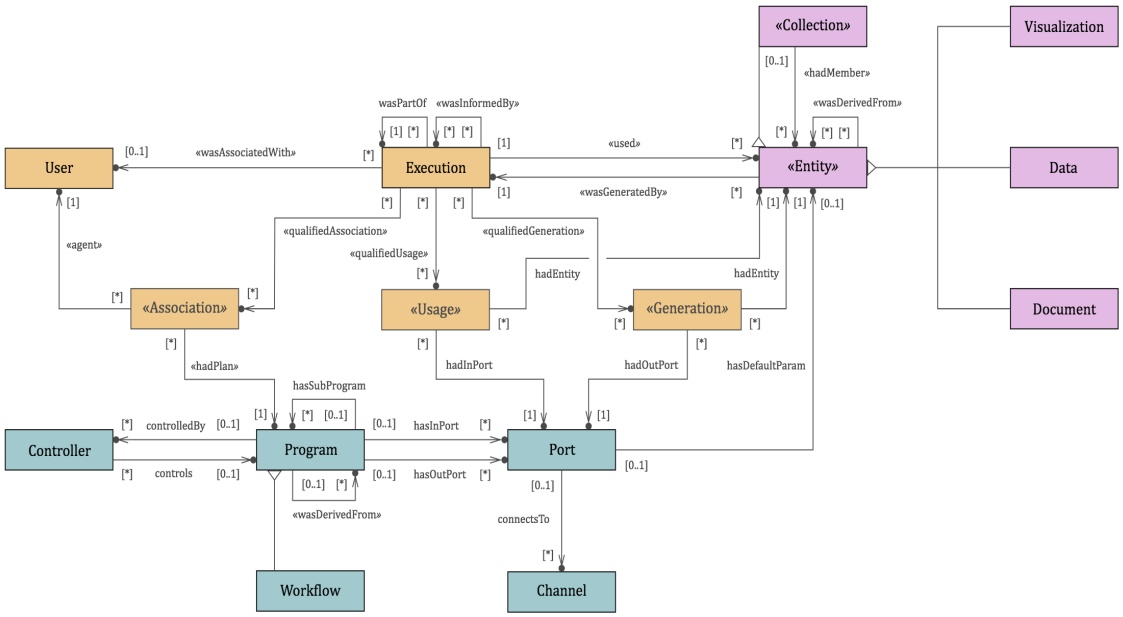


Figure 3.4: The ProvONE Data Model. Image obtained from the ProvONE online documentation [52]

such as *dispel4py* [134], that deals with the distribution and the parallelisation of the single operators, and that adds extra functionalities to enforce deterministic *data-flow* behaviours, as needed.

In this section we introduce a schematic representation of the *observables* of such a system and propose a model to define their provenance relationships. We adopt for the purpose a UML notation, known as Class Diagram [79], and we import and further specialised concepts introduced by the PROV [83] data-model in combination with ProvONE [52]. PROV is intended as a conceptual framework offering machine understandable descriptions of records that describe with contextual metadata people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of information. In PROV, *Activities* inform other *Activities* by means of an exchange of information and they generate and use *Entities*. *Entities* can be derived from other *Entities*. Respectively they are associated and attributed to *Agents* who perform or delegate an action. ProvONE, shown in Figure 3.4 in its latest version, offers instead an extension point to accommodate the provenance representation of more particular workflow computational processes.

More specifically, while ProvONE represents the structure of a *provone:Workflow* as a graph of interconnected entities of type *provone:Program*. These are executed according to the computational model specified by a *provone:Controller*. In the provenance model proposed in this chapter, that we call S-PROV, we extend the description of the abstract workflow by introducing a new class, *Component*, which extends the basic PROV class *prov:Agent*. Components participate in the concurrent execution of a workflow, by making sure that the activity described by the *provone:Program* take place in parallel on the incoming input. Thus they delegate the execution of a program to multiple instances of the program itself. We represent the instances in S-PROV by introducing a new class *ComponentInstance* that extends the PROV class *prov:SoftwareAgent*. Making this semantics explicit enables us to represent in the provenance traces detailed information about the execution of parallel operators. For instance, the model captures the occurrence of asynchronous changes propagated to the instances' of a component, as well as the updates to their internal state with intermediate and reusable data. We will provide more information about these properties, respectively in sections 3.2.1, 3.2.2 and 3.2.3. Moreover, ProvONE, through the concept of *provone:Workflow*, which is also a program, and the relationship *provone:subprograms*, shown in Figure 3.4, offers support for workflows encapsulation, which is an important and reusable feature of the model. Here the activity class of *provone:Execution* applies to workflows, as well as their internal components. We extend this concept in order to differentiate between the execution of a complete workflow and a simple process. The former is described by the class *WFExecution*. It shows associations with all the *Component* agents participating in the execution of a workflow and links to the adopted initialisation inputs. Instead, the *Invocation* activity class, describes the execution of a specific *ComponentInstance* that iterates over the incoming data on behalf of one of the workflow's *Component* agents. During an invocation the relationships between the input and output data and the set of parameter values used by the specific instance are established. Finally in S-PROV we combine system-level provenance with contextual information that is relevant to the users' interests and to the workflow's application domain. This is achieved programmatically by the developer and it is offered as a set of configurable options to the user of the workflow, as we will discuss in Chapter 4, when we will introduce the *Active* framework.

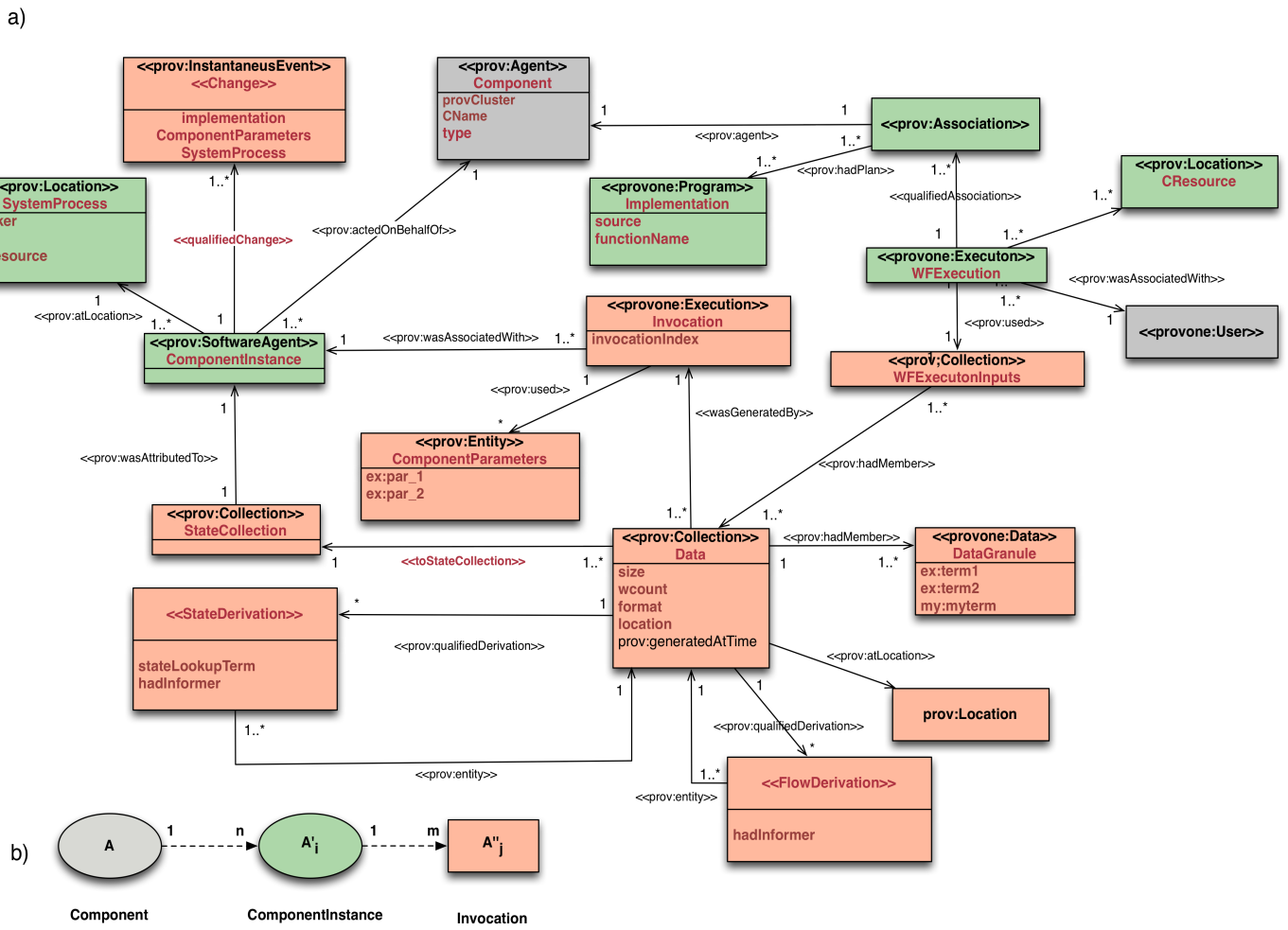


Figure 3.5: (a) S-PROV provenance model. Colour coding indicates the following: (grey) elements for abstract and prospective provenance; (green) concrete workflow elements and state; (red) execution elements and fine grain dependencies. Extensions of PROV and ProvONE are indicated for each class. (b) After the abstract workflow is deployed to the underlying resources, each *Component* is mapped into several *ComponentInstances* that perform many *Invocations* to execute the workflow task on the incoming data. *Components* and *ComponentInstances* are *prov:Agents*. They are represented as ovals throughout the chapter.

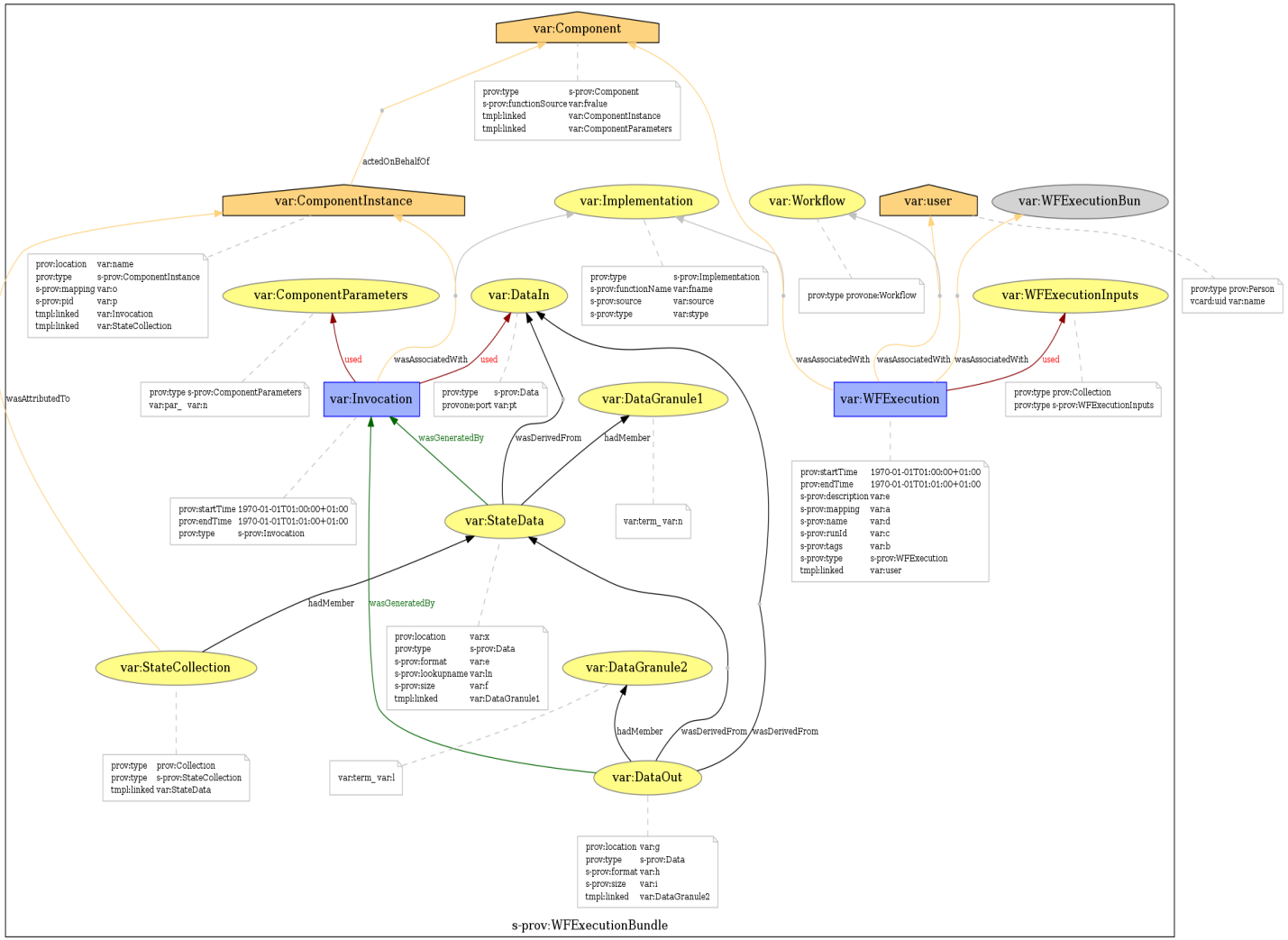


Figure 3.6: The image shows a representation obtained from a PROV-N document that describes the provenance relationships between the S-PROV entities and activities characterising the *Invocation* of a workflow's operator. The semantic association with S-PROV concepts is obtained by assigning S-PROV qualified class names to the *prov:type* attribute of the fundamental PROV elements.

In summary, S-PROV addresses aspects of mapping between logical representation and concrete implementation of a workflow and its enactment onto a target computational resource. Here the model captures aspects associated with the distribution of the computation, volatile and materialised data-flow and the management of the internal state of each concrete process. Moreover, it captures changes occurring to the workflow at runtime, especially concerning dynamic steering [183]. The work of Mattoso *et al*., refers to the capabilities of a workflow system to adapt to changes which are classified by an *Adaptation Taxonomy*. This taxonomy covers aspects such resource migration, tasks and data configuration refinement, domain data reduction and parameter-space slicing. Changes can be triggered by users and by automated rules that are activated during the monitoring and analysis of the running workflow. S-PROV captures the provenance characteristics of some of of the taxonomy's elements, especially concerning task refinement, dynamic parametrisation and resource migration.

Thus, S-PROV introduces new classes and properties as extensions of PROV and ProvONE to broaden the coverage and the precision of provenance information with more explicit semantics. Figure 3.5 (a) represents the relationships occurring between the different players of a data-intensive workflow, according to S-PROV, that deal with distribution and scale-up of the single tasks, stateful operators and dynamic steering. PROV and ProvONE concepts are expressed as stereotypes when they are extended by S-PROV classes with more specific properties and semantics. For simplicity of notation throughout the chapter, we consider S-PROV as the base namespace for all the elements without an explicit namespace indication.

S-PROV is available as an ontology[1] in RDF (Resource Description Framework) for its experimental adoption. RDF [58] is a general-purpose language and conceptual framework which is commonly used to represent information over the Web. Here resources are substantially represented as nodes of a directed graph, where edges are relationships of the form *subject-predicate-object*. An ontology adopts the RDF data-modelling vocabulary, or RDF Schema [59], to formally define new resource classes and datatypes including their properties and relationships.

[1] `https://github.com/aspinuso/s-provenance/blob/master/resources/s-prov-o.owl`

Table A.1 and Table A.2 in the Appendix A, describe the classes and the properties of the model in detail. Figure 3.6 depicts a visual representation of a S-PROV snapshot produced from a document in PROV-N [49] notation. It shows the relationships between entities, agents and activities involved in a single *Invocation*. We will briefly come back to this representation in Section 4.3.1.

3.2.1 Resource Mapping

Once the workflow is defined as a *provone:Workflow* plan, its execution associates the workflow's tasks with logical *Components*. These are agents that delegate the execution of their tasks to concurrent *ComponentInstances*. Instances fire an *Invocation* of the task with each occurrence of new input data. A *ComponentInstance* can be precisely located within a running *SystemProcess*, which is a *prov:Location* with coordinates such as *worker* and *pid*; respectively the cluster node and the operating system process identification number. Multiple instances can be located in the same worker and executed by the same system process. The *mapping* property captures information on the type of execution engine adopted for the enactment of the workflow application, depending on the ones supported, *e.g.* multiprocessing [56], MPI [37], *Apache Storm*, *etc.*

Each *Invocation* is counted and indexed (*invocationIndex*). This feature can provide in-depth perspective on the distribution of a data-flow computation, especially if combined with other properties, such as execution timestamps and the size of the *Data*. This allows workflow developers, as well as automated optimisation engines, to identify through provenance analysis overcommitted or under committed instances, gaining a better understanding of the actual exploitation of the underlying resources.

3.2.2 Stateful Operators

As already mentioned, new relationships and classes in S-PROV address the provenance of stateful operations. The scope is to capture and exploit information about the state of a process in order to produce lineage data with tuneable precision, improving the overall usability of the provenance traces. This supports diagnostic and fine-grain

validation use cases, especially when complex dependencies are established with intermediate stateful data that depends on a potentially large and sparse portion of the input. In case of a materialised state, the combination of the *prov:location* property with stateful data entities can enable the validation of stateful bulk operations [174] or suggest a provenance-driven implementation of fault tolerance mechanisms [116].

The relationship *toStateCollection* (Figure 3.5) indicates the update of the *StateCollection* by an *Invocation* with new *Data*. The *stateLookupTerm* is a key associated with the data. It can be generated according to the logic encoded within the workflow management system, its components, or defined by the user. In order to represent and capture provenance patterns and custom provenance assertions about stateful operations, S-PROV extends the *prov:Derivation* class with two new classes: *FlowDerivation* and *StateDerivation*, with the aim of distinguishing the interactions of a process with the incoming data-flow from those involving internal data entities. In Table 3.3 we will provide a more formal definition of these two classes in terms of their relationship with the *StateCollection*, and a preliminary description of common patterns supported by this model.

While a *FlowDerivation* describes a *prov:wasDerivedFrom* dependency between output data and data received in input, a *StateDerivation* expresses again the same relationship, but in this case between output data and products associated with some particular logic internal to the component. As an example, we can consider an operator that computes and outputs periodic averages on the incoming data, with threshold overflow detector as second output. In Figure 3.7 we show a trace of the data dependencies for a detection. This is described by a *StateDerivation* with the last computed average, which is preserved in the component's state, and by a *FlowDerivation* with the input value just received. The average is preserved in the component's state and shows in its lineage a *FlowDerivation* with all the incoming data contributing to its value. Optionally, it could also show a *StateDerivation* with the previous average.

Enabling developers and users to be in control of such dependencies should be considered among the fundamental functional requirement of a workflow system that fosters reproducible and traceable science. In Chapter 4 we will further describe how we can model and capture these behaviours in our active provenance framework.

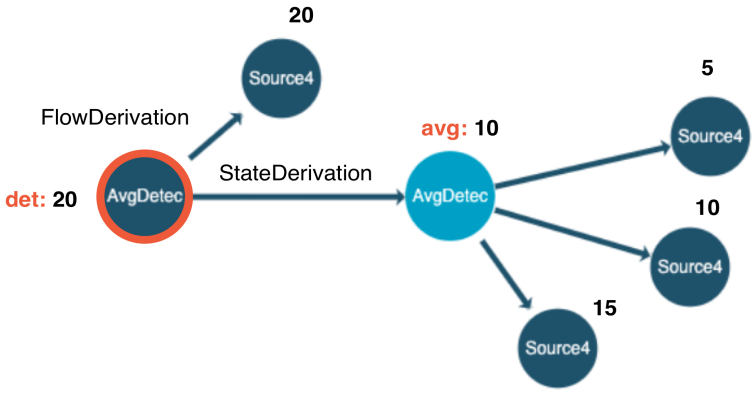


Figure 3.7: Data derivation trace produced when detecting a threshold overflow (circle with red contour), during the computation of periodic averages. Circles are data entities labelled with the name of the generating component, while edges represent the *prov:wasDerivedFrom* relationship. In the example, the overflow is detected by the `AvgDetect` on the input value (20) received from a `Source` component after this has been compared with the the periodically computed average (light blue circle with value 10). Thus, the detection is derived from the current input (*FlowDerivation*) and from the average (*StateDerivation*). The image shows also the dependencies (*FlowDerivation*) between the average and its contributing values.

As further clarification for the realisation and use of these derivations in PROV, we take advantage of the qualification pattern used by the PROV ontology (PROV-O) [50] that allows us to introduce additional descriptions about the binary *prov:wasDerivedFrom* relationship between two entities. Listing 3.1 shows an example of state derivation in RDF expressed in Turtle notation [73]. Among all the possible serialisations of RDF, the Terse RDF Triple Language, or Turtle, is considered the one which is more readable by humans, given its natural text and compact format. Turtle represents triples as simple groups of URLs, with less verbosity and editing complexity respect to an XML serialisation.

Here the *s-prov:hadInformer* specifies which invocation communicated the `:stateData` entity. This property allows to include more details to self-contained documents that describe the provenance entities and the relationships involved in the production of a specific output of a workflow component.

Listing 3.1: Qualified derivation using a *StateDerivation* Class to describe stateful data dependencies.

```
1 :dataOut
2         a s-prov:Data
3         prov:wasDerivedFrom :stateData;
4         prov:qualifiedDerivation [
5                 a s-prov:StateDerivation;
6                 prov:entity :stateData;
7                 s-prov:stateLookupTerm "term"^^xsd:string,
8                 s-prov:hadInformer :invocation];
```

For instance, as a short anticipation to what we will discuss in chapters 4 and 5, Figure 3.8 shows an example of such self-contained document in JSON-LD format [33]. It depicts how the internal structure of the document represents and links output data with different types of derivations. In the document sown by the image, the *prov:Derivation* list contains *FlowDerivation* and *StateDerivation*. They reference the entities used by the process and describe the details about how they were acquired, *i.e.* by accessing the state or by reading from the input ports. The example refers to the lineage of a *stateful* operator of a correlation analysis workflow. We will present the details of this workflow in the next chapter, when we will use it as a test case to demonstrate our provenance capturing framework. We proceed now, introducing another provenance class, which describes the changes occurring to a *ComponentInstance* at runtime.

3.2.3 Dynamic Changes

Besides statefulness of functions and associated dependencies, another aspect covered by the model is the traceability of runtime changes which affect one or more components of the workflow. ProvONE can handle traceability of processes and workflow evolution via the *prov:wasDerivedFrom* relationship between them. We complement this capability by observing changes triggered by steering actions. Changes can be provoked by observers (human or software) during the runtime analysis of the progress of the workflow execution, and get propagated asynchronously to those agents (or actors) that concurrently perform a specific task on behalf of a logical component.

Coordinating the synchronisation of changes to a large number of instances is a com-

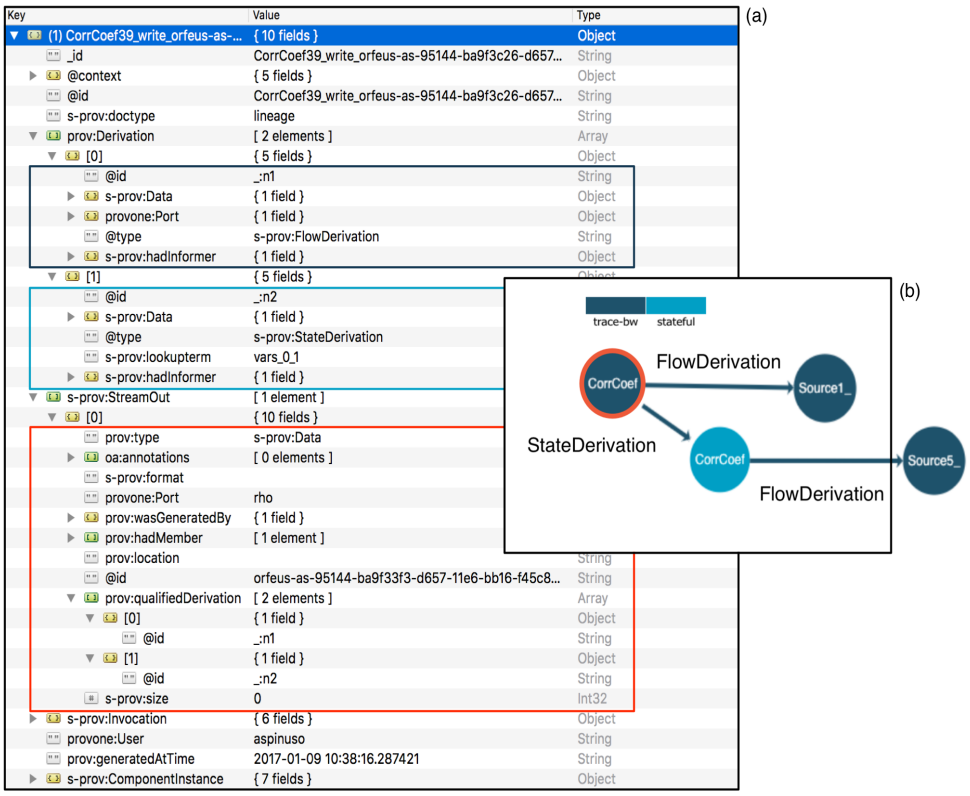


Figure 3.8: JSON-LD with S-PROV elements. The document (a) describes all the provenance relationships established when a stateful streaming operator (`CorrCoef`), produces a new output (red square). Graph (b), shows the data derivations represented in (a). The circles are provenance entities of type *Data* and their labels indicate the generating operator. For clarity of representation we depict the data in the red square with a red contour. Input and output data are in dark-blue, while the light-blue circle represents a stateful data entity. Arrow edges are *wasDerivedFrom* relationships. The dependency on the data entity which is placed outside of the box in (b), indicates that its derivation is described in another document.

plex operation. Especially in near real-time applications, any reconfiguration should be carried out with minimum disruption to the data flow and the buffering of data is not always an option. Therefore, we consider the generic scenario where runtime changes may be propagated asynchronously to all the *ComponentInstances* acting on behalf of a logical component.

Listing 3.2: Qualified change of a *ComponentInstance* `:component_A_instance_0` with new *ComponentParameters* `:cpar_1`

```
:component_A_instance_0
        a s-prov:ComponentInstance
        prov:wasChangedWith :componentParameters;
        prov:qualifiedChange [
                a s-prov:Change, prov:InstantaneousEvent;
                prov:componentParameters :cpar_1;
                prov:atTime "2018-02-29T00:00:01Z"^^xsd:dateTime;
                ];
```

To handle the provenance of these applications within parallel enactments, we consider *Component* and *ComponentInstance* as subclasses of *prov:Agent*. The former delegates to its instances the responsibility of an activity to take place, according to its current *Implementation* and *ComponentParameters*. The dynamic adjustment of an instance is represented in S-PROV by the qualified property *qualifiedChange*, that provides additional descriptions about the *wasChangedWith* relationship. This refers to a *Change* object that characterises the nature of the adaptation, as a new set of *ComponentParameters*, a different *Implementation*, or the migration of the instance, and its state, to a different *SystemProcess*. Moreover, we highlight the instantaneous effect of the change by adopting the notion of *prov:InstantaneousEvent*, as shown in Listing 3.2.

With the support of the provenance model, users can get a global view of the execution behaviour in respect to changes occurred at runtime. For example, when processing and combining data from many sensors, the re-parameterisation of a stage of the pipeline may affect the computation consistently only after the change was propagated to every instance of the stage. Thus, by recording the nature of the change and time-stamping its occurrence at instance level, users can take decisions on how to interpret those results that show dependencies with data generated during its propagation.

We introduce in the next section a set of functions that use the model to validate, through retrospective analysis, the structure and the consistent behaviour of the workflow and that exploit state management for the identification of the data dependencies. The latter as the means to foster provenance precision and usability.

Set	Description
C	The set of *Component* agents of the abstract workflow.
Ci	The set of *ComponentInstance* agents acting on behalf of the workflow components.
I	The set of *Invocation* activities of the *ComponentInstance* agents.
D	The set of *Data*.
$Sd \subset D$	The set of *Data* which are members of a *StateCollection*.
Dg	The set of *DataGranule* entities that are members of elements in D.
M	The set of *DataGranule* entities' metadata *(key,value)* pairs.

Table 3.1: Overview of the main S-PROV sets.

Operations	Description
(3.1) $\alpha : I \rightarrow Ci$	Function of the Multi Invocation Model [94]. Returns the *ComponentsInstance* associated with a specific*Invocation*. Its inverse is $\alpha^{-1} : Ci \rightarrow \mathcal{P}(I)$.
(3.2) $\beta : Ci \rightarrow C$	Returns the *Component* the workflow related to a specific *ComponentInstances*. Its inverse is $\beta^{-1} : C \rightarrow \mathcal{P}(Ci)$.
(3.3) $edge_C : C \rightarrow \mathcal{P}(C \times C)$	Returns the edges connecting a *Component* with others in the abstract specification of the workflow.
(3.6) $edge_{Ci} : Ci \rightarrow \mathcal{P}(Ci \times Ci)$	Returns the edges connecting a *ComponentInstance* with others in the concrete representation of the workflow.
(3.4) $state_{inv} : I \rightarrow \mathcal{P}(Sd)$	Returns the set of *Data* stored in the *StateCollection* at the time of the occurrence of an *Invocation*.
(3.5) $dep_{bw} : D \rightarrow \mathcal{P}(D)$	Returns the set of data from which a *Data* element was derived. Its transitive closure retrieves the complete data dependency graph (backward navigation).
(3.8) $dep_{fwinv} : (D \times I) \rightarrow \mathcal{P}(D)$	Returns the set of forward dependencies generated by a given *Invocation* and derived from a given *Data* element.
(3.7) $gval : \mathcal{P}(C \times M) \rightarrow \mathcal{P}(I)$	Grouping validation. Returns the set of component's *Invocations* that generated data with the same metadata values and that are associated with different instances.

Table 3.2: Operations defined on the S-PROV model that contribute to answer provenance queries. They offer views on the model at different levels of detail.

3.3 Multi-Invocation and S-PROV

With this section we will introduce a framework for the exploration of the lineage information. The scope is to demonstrate how we can combine the elements and the relationships of S-PROV involved in the representation of runtime observables to explore different properties of the model scenarios. In tables 3.1 and 3.2 we summarise the sets and the description of the functions.

We adopt the Multi-Invocation Model [94] as our reference model to represent basic interactions between the observables in a streaming computation. Here, the set of lineage traces are defined as follow: $T = (V, E, \alpha)$ where V are verices of $D \cup I$, where D is the set of *Data* elements obtained by a computation over a data-stream, and I is the set of process *Invocations*. The set E consists of edges between elements in D and I. We will interpret the Multi-Invocation Model through the S-PROV schema, introducing additional operations that contribute to use provenance data to perform structural validation, trace the propagation of steering actions and represent complex lineage scenario, such as those associated with *stateful* components.

The Multi-Invocation model defines a function $\alpha : I \rightarrow Ci$ to map from an invocation to its actor or, according to the S-PROV schema, to its *ComponentInstance*. Thus, the function α returns the *ComponentInstance a*, by following the *wasAssociatedWith* relationship between an *Invocation i* and the instance, as shown in Figure 3.5:

$$\alpha(i) := a \ \in \ Ci \tag{3.1}$$

Its inverse $\alpha^{-1} : Ci \rightarrow \mathcal{P}(I)$, with $\alpha^{-1}(a) := \{i \in I \mid \alpha(i) = a\}$ is the set of invocations of an actor a during a workflow run.

The function $\beta : Ci \rightarrow C$, is computed by following the *actedOnBehalfOf* relationship between a *ComponentInstance* and the abstract *Component*:

$$\beta(a) := c \ \in \ C \tag{3.2}$$

It returns the component which the instance *actedOnBehalfOf* according to the resource mapping of the workflow to the target enactment. Its inverse is $\beta^{-1} : C \rightarrow Ci$,

with $\beta^{-1}(c) := \{a \in Ci \mid \beta(a) = c\}$.

Now we can introduce $edge_C : C \to \mathcal{P}(C \times C)$ as follows:

$$\begin{aligned} edge_C(c) := \{(c,c')|\ i \in \alpha^{-1}(\beta^{-1}(c)),\ d \in D, j \in I : \\ (d\ wasDerivedFrom\ i) \wedge (j\ used\ d)\ \wedge\ c' = \beta(\alpha(j))\} \end{aligned} \tag{3.3}$$

Which is the set of all pairs (c,c') representing the *wasInfluencedBy* relationship between the given *Component* c and other components c' exchanging information. This relationship, in PROV terms, describes the capacity of an entity to affect the behaviour of another by means of delegation and, as in this case, communication. We could include a new sub-property to specify communication through agents or suggest to extend the one already included by PROV for activities (*wasInformedBy*). This function can be used to validate that all the communications occurring among the instances match the abstract definition, for instance, described by the prospective provenance of ProvONE.

3.3.1 The *StateCollection* and Provenance Patterns

This section addresses the provenance support for stateful dataflow programs that perform complex multi-step computations. These applications must combine the potentially large and continuous input data with data derived from previous batches, leveraging persistent state to store prior or partial results across multiple iterations. Incremental processing includes aggregate operators (*e.g.* min, median, sum, correlation, etc.), data mining algorithms [180], page-rank implemented through continuous bulk processing (CBP) [174], streaming map-reduce [109], *etc.* In such context state is a fundamental requirement to obtain outputs efficiently and it is not limited to window-based functions, that typically only depend on a recent finite set of input data. [87, 207]. The adoption of stateful operators opens complex scenarios when it comes to the collection of provenance recordings that correctly describe the lineage of such workflows. For this reason, in this section we introduce of the provenance state as one of the fundamental observables to support such situations.

We assume that an *Invocation* performs data updates and look-ups of the *StateCollection* of its *ComponentInstance*. Data can be added and replaced, with the effect that the same look-up at different times may return different elements. The replacing elements in the state could be derived from the replaced ones, indicating the possibility to trace state transitions of the component's internal data structures that are iteratively updated.

Hence, in order to add and exploit the concept of state in our framework, we define two functions $state_{inv}$ and dep_{bw} as follows. The function $state_{inv} : I \rightarrow \mathcal{P}(Sd)$, with $Sd \subset D$, returns the set of *Data* stored in the *StateCollection* during the occurrence of an *Invocation* j.

$$state_{inv}(j) := \{\, d \mid d\ wasGeneratedBy\ j \wedge d\ toStateCollection\ s \wedge s\ wasAttributedTo\ \alpha(j)\} \tag{3.4}$$

Given $state_{inv}$, in Table 3.3 we provide a more formal definition of the two classes *FlowDerivation* and *StateDerivation* that we have already introduced in Section 3.2.2. These will allow us to distinguish and capture different patterns involving data-flow and stateful data dependencies.

S-PROV: Types of Data Derivations	
Derivations	**Description**
FlowDerivation	A `prov:wasDerivedFrom` relationship between $a, b \in D$, where b is ingested from any input port of a *ComponentInstance* during a *firing-round*, before writing a. Thus, for each invocation i, $b \notin state_{inv}(i)$
StateDerivation	A `prov:wasDerivedFrom` relationship between $a \in D$ and $b \in state_{inv}(i)$ for an invocation i that occurred before a was produced. This indicates that a depends on an entity in the *ComponentInstance*'s *StateCollection*.

Table 3.3: Types of data derivations in S-PROV that further qualify the *prov:wasDerivedFrom* relationship.

Finally, the function $dep_{bw} : D \rightarrow \mathcal{P}(D)$ returns the set of the dependencies of a *Data* element a. These are represented in the following definition in terms of their participation to the two qualified derivations.

$$dep_{bw}(a) := \{\, b \mid a\ wasDerivedFrom_{FlowDerivation}\ b\} \cup \{\, c \mid a\ wasDerivedFrom_{StateDerivation}\ c\} \tag{3.5}$$

While dep_{bw} concentrates on the inputs and outputs of the invocations of a single actor and the snapshot of its state, its transitive closure $dep^{*}_{bw}(a)$ will navigate through the dependencies across the other workflow components, returning all the *Data* contributing to a.

In the proposed representation, the application of the RWS (Read-Write-ReSet) provenance generation model [177, 108], where every output data of a component is considered dependent on all the data received since the beginning of the *firing-round*, the data dependencies of an output a returned by $dep_{bw}(a)$ will be all involved in *FlowDerivations*. In our framework, we want to address more sophisticated components that preserve in their state intermediate results that are reused across *firing-rounds*. Thereby, to resolve the ambiguity between the inputs of a *firing-round* and *stateful* data, we redefine the *state-reset* of the RWS as instead the event that, at runtime, discards all the elements in the *StateCollection* of an instance, and we rename the event that delimits the end of a *firing-round* as *flow-reset* event, with no effect on the *StateCollection*. As an example, if we consider again a streaming component that produces periodic averages, the *flow-reset* ensures the recording of the correct *FlowDerivations* within the *firing-round* generating the output. If the component also detects the overflow of the average by its input values, as the one whose trace is shown in Figure 3.7, a *state-reset* may capture the situation where the current average is discarded, with the effect of suspending the detection until the next computation. It is to be noted that triggering a *flow-reset* event after a detection, would cause to discard the dependencies on the inputs from the computation of the next average. This suggests the need to provide workflows' developers with usable and semi-automated controls over *FlowDerivations* and *StateDerivations*, to better tune complex scenarios. These will be covered in Section 4.3.1.

The rules regulating the presence of an entity into the state are dependant on the pat-

terns associated with input/output dependencies. These can be associated with common reusable pattern as well as driven by a more complex internal logic. These cases can be managed by the developer or by automatic mechanisms that associate specific provenance patterns to a component, as we will show in Chapter 4. We aim at a provenance framework that integrates within a workflow system and exposes a set of rules and extensions that provide immediate support to the developer in keeping dependencies consistent. Thus fostering the usability of the provenance produced.

3.3.1.1 Patterns Overview

The description of the dependencies just presented allows us represent the following provenance patterns:

(a) SingleInvocation: this is the simplest pattern, where a component's output depends on the data ingested during the current invocation. The function $dep_{bw}(d)$ will go through only the first part of its definition, because the state will always be empty.

(b) Accumulation: the data produced by an *Invocation i* depends on all the data ingested by the previous *Invocations* until a certain j (or *firing-round*). Given a *Data d* produced by this type of *ComponentInstance*, $dep_{bw}(d)$, will return all dependencies belonging to *FlowDerivations*,

(c) Filtering: Discarded items could be traced in a *read-write* cycle, for instance, as part as an informative payload to permit analysis of the correct functioning of the filter. However, most commonly this could be treated as a stateless **SingleInvocation** operator that establishes dependencies only on input and output data passing the filter test.

(d) *n*-by-*m*: in order to produce results, a process may need to read the data coming from all of its input ports, or parameters. This provenance profile assumes that all of the *n* input are traceable to the *m* outputs. In systems that do not in work in lock-step, for instance as in Apache Storm [5], references to the data may have to be added to the provenance state across multiple invocations, and accessed when all the needed inputs are collected.

(e) n-by-m partial accumulation: A more generic scenario, also typical of a streaming system, is characterised by this provenance pattern where a process produces m output data that depend on a subset of the data ingested by any combination of n input parameters. As an example of a n-by-1 partial pattern, we consider the component that produces the correlation coefficient of pairs of incoming variables that belong to more sampling sequences, see Section 4.4. In an asynchronous streaming scenario variables of different sampling sequence can arrive to the component at any time and need to be correlated to those belonging to the same sequence. If not handled properly, this may cause ambiguity in the lineage. Managing this scenario explicitly through the provenance *StateCollection* helps developers to establish relevant dependencies with intermediate data, avoiding the generation of incorrect or imprecise traces [91]. Therefore, for the correlation example, the set of dependencies extracted by the function $dep_{bw}(d)$, where d is a correlation coefficient generated by an invocation i, will include state's updates $state_{inv}(j)$, performed by some j, where j occurred before i.

In Chapter 4, we will cover with more detail how we address these and other cases, such as operators on sliding windows, grouped inputs and presenting more specific statefu dependencies. Enabling he flexible and semi-automatic manipulation of the *StateCollection* improves the usability of the traces and their correctness when used for results management [91]. Increasing the benefits yield by the production of lineage will motivate research-developer and users in adopting provenance-aware development strategies. However, the mechanisms tuning provenance precision have to be easy to integrate and instrument, fostering the users' contribution without overwhelming them by redesigning their established methods and procedures.

3.4 Using S-PROV, Examples

How to relate provenance at different levels of abstraction, possibly automating the extraction of high-level summaries from detailed records is one of the challenges that when solved leads to usable provenance products. We introduce in this section how we can obtain from S-PROV and its associated functions, views at many levels of depth

and in support of different interests. This can be achieved following a bottom up approach, from the *Data* and their dependencies up to the generating processes and the associated workflow's components and plans. Broader analysis can also include the production of comprehensive views across multiple runs and users. We will discuss these aspects later in this thesis, as well as our approach to assist end-users with customisation and contextualisation. We focus here on examples providing insights on single workflow executions.

Concrete and Fine-grain retrospective: An interrogation to the S-PROV model can provide interesting details about how the *ComponentInstances* are connected and distributed during a parallel execution of the workflow, for instance, highlighting their distribution across system's processes and across the working nodes of a MPI cluster. To obtain such view we can use a redefinition of $edge_C$ as $edge_{Ci}$: $Ci \rightarrow P(Ci \times Ci)$ as:

$$edge_{Ci}(a) := \{(a,a')_{ij} \mid i,j \in I : (\alpha(i) = a) \wedge (i\ wasInformedBy\ j) \wedge (\alpha(j) = a')\} \tag{3.6}$$

This function allows developer and computational experts to obtain insights about the distribution of the concrete workflow and data exchange. Producing all of the occurrences of the edges associated with the *Invocations* allows them to retrieve and then aggregate information about the overall computation such as, combined distribution patterns with indication of the actual size of the data exchanged between *ComponentInstances*. In Figure 3.9 we show, as an anticipation, how these possibilities can be offered by interactive visual interfaces based on radial diagrams and *Hierarchical Edge Bundles* [157], supported by customisable grouping. Grouping affects the clustering of the vertices of the diagram, which in the figure just introduced, is done by *worker* node, suggesting how the visualisations tuned on the explicit properties and semantics of the model, allow to observe the dynamics between the *ComponentInstances* distributed within a target system. More insights on the exploration of these techniques will be provided in Section 5.7. Finally, the possibility of capturing stateful operations allows a finer-grain precision of the provenance to describe

complex lineage patterns as well as specific scenarios. Chapter 4 will introduce how provenance state management can be included within a data-flow workflow system.

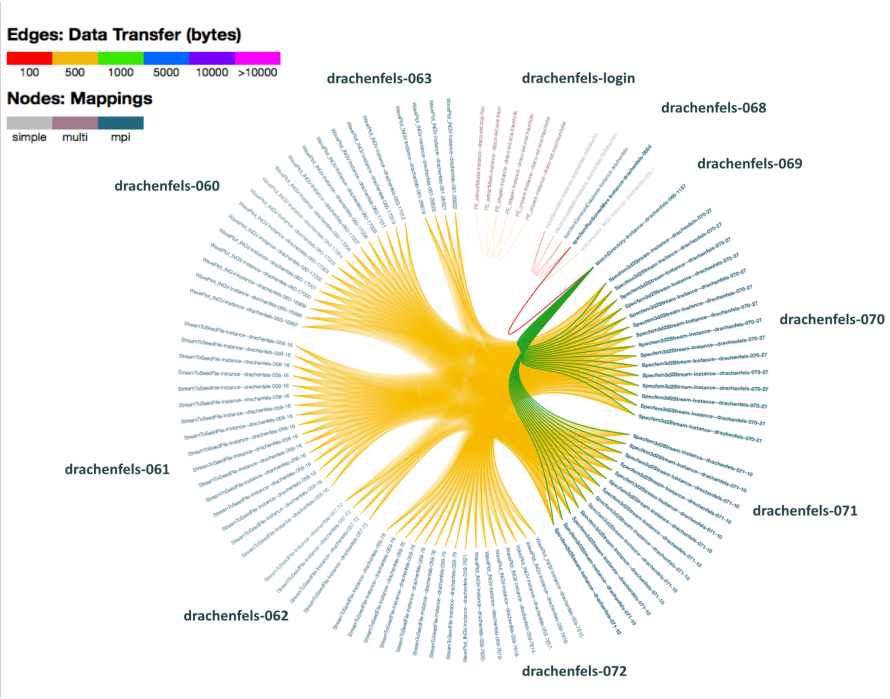


Figure 3.9: The diagram indicates data transfer interactions between *ComponentInstances* (vertices), grouped by the worker node where they are deployed within a computational cluster (provided by SCAI-Fraunhofer [65] in the example). The two colour-coded legends describe respectively the amount of data transferred and the type of enactment (single-process, multiprocessing, MPI). Users can hover on the nodes to highlight incoming (red) and outgoing (green) streams.

Diagnostic and Validation: Debugging distributed systems can be difficult and requires analysis of log files on several machines. By combining concrete mapping and fine-grain retrospective, S-PROV helps to describe and discover facts about such a system at runtime. Starting from the *Invocations*, we can perform workflow diagnostic and validation use cases, such as the detection of overcommitted instances and the identification of anomalies. The former could be obtained by observing the number of invocations occurring and the amount of data transferred in a certain time-window. This could help highlighting time-

dependent patterns that may be related to the workflow application logic or to the distribution plan. Anomalies, instead, may be provoked by errors or by dynamic changes. Errors may affect specific invocations within the same instance or group of instances of a component, suggesting localised behaviours. In that respect, S-PROV captures errors as messages associated with invocations.

Finally, in those scenarios where a workflow's component can change during its execution, for instance with a new parametrisation or implementation, if not synchronised across all the instances, such event may introduce anomalies into the results. We showed in Section 3.2.3 that changes are expressed in S-PROV by the *wasChangedWith* relationship of the *ComponentInstance*, and its qualified description is time-stamped, see Listing 3.2. If we consider workflows that combine data streams with others produced by a changing component, users may choose to invalidate a subset of the results. These can be selected among the ones that present in their ancestors derivations involving the changing component in the time-window in which the change was being propagated. In general, we consider scenarios associated with asynchronous runtime changes a provenance challenge that requires further investigation.

Moreover, if we consider a component that performs operations on groups of data that are characterised by specific metadata values, by combining the domain metadata of the *DataGranules* and the *Instances* that ingest them, we can exploit the model to identify erroneous or unexpected behaviour of the system, such as the routing of data towards wrong instances of the concrete workflow. To do so, we introduce the set M of *(key,value)* pairs describing a *DataGranule* metadata, and Dg the set of all the data granules. Let M_{kv} be any subset of M that appears among the properties of a data granule, $d_{M_{kv}}$, we can define the function $gval : \mathcal{P}(C \times M) \rightarrow \mathcal{P}(I)$ as follows:

$$\begin{aligned} gval(c, M_{kv}) := \{ i \in I | \; \exists \; j \in I, \; i \neq j, \; d_{M_{kv}}, b_{M_{kv}} \in Dg : \\ (i \; used \; d_{M_{kv}}) \wedge (j \; used \; b_{M_{kv}}) \\ \wedge (\alpha(i) \neq \alpha(j)) \wedge (\beta(\alpha(i)) = \beta(\alpha(j)) = c) \} \end{aligned} \tag{3.7}$$

The set produced by $gval(c, M_{kv})$ returns the invocations performed by different instances of the same component c that have ingested granules with the same metadata and value pairs. This set must be $\emptyset$, for all the M_{kv} that are composed by those pairs upon which c is expected to delegate its operations to the same instance, and thereby it proves that the same *(key,value)* pair is never received by different instances. Figure 3.10 shows an example of a wrong routing between from a component C0 and a grouped one, C1.

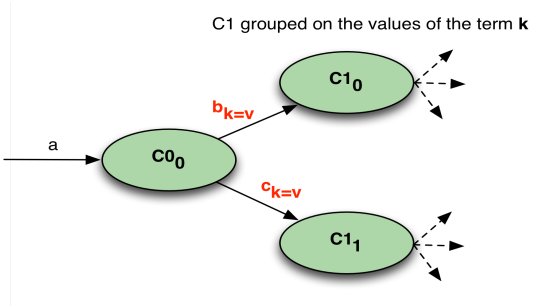


Figure 3.10: The image shows the wrong routing of data $b_{k=v}$, with metadata term's value pair *(k,v)*, sent from C0 to the component C1, which is grouped on the values of k. In this scenario, the function $gval(c, M_{kv})$ applied on C1 returns the instances $C1_0$ and $C1_1$. A correct routing would send $b_{k=v}$ and $c_{k=v}$ to the same instance.

Stream Reordering: We have previously discussed how the parallelisation of streaming pipelines, implemented according to the *Actor Model*, does not necessarily guarantee monotonicity. In the processing of data acquired from environmental sensors, this may prevent meaningful insights when correlating different events. Detecting patterns in sequences of data usually requires order of occurrence, rather than the order of arrival. This scenario is typical in concrete streaming system implementations, where data is serially acquired and then passed over to adjacent components implemented by parallel pipelines. In the literature similar challenges have been studied, for instance in the context where streaming application are mapped onto Multi Processor System-on-Chip [145], suggesting optimal solutions for rapid reordering.

By keeping the count of the write events across invocations (*wcount*), which we indicate here for brevity as *wc*, the provenance offers the possibility to overcome

the non-deterministic exchange of elements of a data stream between components, characteristics of a non-monotonic computational model (as discussed in Section 3.1.1), and to address use cases such as the offline stream-reordering, as we show in this section. Thus, given a simple pipeline, we propose an algorithm that returns an ordered stream by taking advantage of our provenance model, and that takes into account the stream fragmentation applied to the data by the components of the pipeline. For stream fragmentation, we intend the general scenario where, from one input data, a process derives a sequence of different outputs. We now introduce the function $dep_{fwinv} : (D \times I) \rightarrow \mathcal{P}(D)$, where d'_{wc} indicates the sequential index (*wcount* property) of the data produced by the instance through its invocations.

$$dep_{fwinv}(d'_{wc}, j) := \{\ d \mid d\ wasGeneratedBy\ j\ \wedge\ d\ wasDerivedFrom\ d'_{wc}\} \tag{3.8}$$

The function is used by our algorithm and returns the set of *Data*, which is generated by an invocation j using d'_{wc}, thus, having d'_{wc} among their dependencies. Given an interval of write indexes $start\_wc$, end_wc performed by an instance of any component when producing a set of outputs D, and the set $P_{C0,Ck}$ of all the abstract components of the pipeline from $C0$ to Ck, a snapshot of the $ordered\_stream$ downstream of Ck is returned by the algorithm provided below. The functions *orderIndexes*, *minIndex* and *maxIndex* are not reported in their detail. They return, respectively, the ordered sequence of elements in a list of data items (according their writing count *wc*), and the minimum and maximum *wc* index that appears in the list.

```
1: function ORDERSTREAM(start_w, end_w, D, Ck)
2:     for (d_wc ∈ D : wc in [start_wc, end_wc]) do
3: d_wc is read only by one invocation i of an instance of a pipeline's
4: component, thus the following for-loop iterates once.
5:         for (i ∈ I : (i used d_wc) ∧ ( β(α(i)) ∈ P_C0,Ck)) do
6:             D = λ'(d_wc, i)
7:             if Ck == β(α(i)) then
8:                 ordered_stream.append(orderIndexes(D))
9:             else
10:                ORDERSTREAM(minIndex(D), maxIndex(D), D, Ck)
```

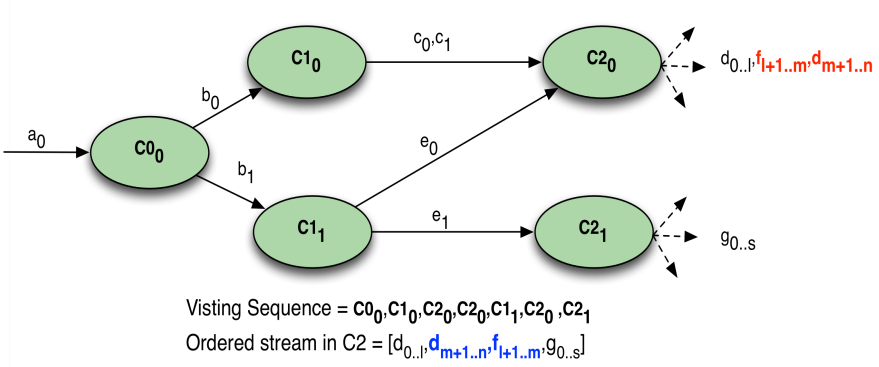


Figure 3.11: The effect of the stream reordering algorithm on a workflow consisting of a pipeline of three components $C0, C1, C2$. The output derived from the input a_0 is distributed across the $C1_x$ component's instances, and we assume its order to be relevant. The instances ingest and process the elements (represented in the figure in their order of arrival, *e.g.* $c_0 \prec c_1$) producing more output elements for each single input. The example assumes that $C2_0$ receives e_0 before c_1 and thus produces $f_{l+1..m}$ before $d_{m+1..n}$ (in red)., breaking the sequentiality of the events attributed to b_0 and b_1. The algorithm traverses the dependency graph by taking into account the *wcount* index of each output attributed to the instances, returning the elements produced by $C2$ in the correct order. Thereby, the ordered stream produced by the algorithm will return $d_{m+1..n} \prec f_{l+1..m}$, (in blue), despite their actual production sequence obtained as output of $C2_0$.

This procedure walks recursively a path in $P_{C0,Ck}$, starting from the source $C0$ and traversing the data dependencies produced by the instances of adjacent components, until Ck. When collecting D, which is the set of the downstream elements of a *ComponentInstance* produced by the same *Invocation* that used the same input data, the algorithm takes into account stream fragmentation.

To summarise, the algorithm is introduced as an example for a potential use of the provenance model in the context of workflow systems implementing computational models that do not guarantee monotonicity and with no global agreement on time or write counts among the workflow's components. It allows, thanks to the the indexes of the output fragments produced by a *ComponentInstance*, the progressive and ordered traversal of the transformations applied to the whole data-stream. More efficient formulations and additional use cases could be provided. Figure 3.11 illustrates a sim-

plified example. This is further explained in Table 3.4, where we report the status of the parameters, the content of the *ordered_stream* and the navigation across the instance elements of the provenance graph, performed by each call of the *OrderStream* function.

OrderStream function calls' parameters and effects		
$(start_{wc}, end_{wc}, D)$	*ordered_stream*	**Visited Instances**
$(0,0,[a_0])$	[]	$C0_0$
$(0,1,[b_0,b_1])$	[]	$C1_0, C1_1$
$(0,1,[c_0,c_1])$	$[d_{0..l}, d_{m+1..n}]$	$C2_0, C2_0$
$(0,1,[e_0,e_1])$	$[d_{0..l}, d_{m+1..n}, f_{l+1..m}, g_{0..s}]$	$C2_0, C2_1$

Table 3.4: OrderStream Function's calls for the example of Figure 3.11

3.5 Conclusions

We have discussed in this chapter the properties characterising a data-intensive system with special attention to aspects of stream computing. After considering different model of computations, taking into account their characteristics and constrain, we motivate our choice for adopting the more general *Actor Model* as our target. We define the provenance *observables* of such model and introduce aspects associated with the representation and management of the actors' internal state and resource mapping. It takes into account the abstract representation of a workflow and the concrete realisation of each abstract component as concurrent actors that are distributed and executed on a target resource. Moreover we consider dynamic scenarios where a distributed workflow application may change the behaviour of its components at runtime, for instance by means of relocation, implementation and re-parametrisation of its actors, which may occur asynchronously. We describe these observable and their relationships in terms of a provenance model, S-PROV, that uses and further extends PROV and ProvONE. A set of queries are introduced to show how the retrospective provenance captured by the model can be used to explore the workflow execution, with special attention to stateful operators, its concrete mapping and dynamic changes. Finally

more use-cases are presented. These include fine-grain visual-analytics combining information about the allocated resources and the intensity of data transfers, diagnostic of errors and resource consumption, the validation of grouped operations and offline reordering of portions of the data stream. We presented the application of the model properties in support of a steam-reordering scenario for concurrent and asynchronous processing pipelines.

Chapter 4

Active Provenance

This chapter illustrates the conceptual and technical framework enabling tuneable and actionable provenance in the context of a stream-based workflow system. It introduces the concept of *Agile* data-intensive system to define the characteristic of our target platform. It introduces of a novel approach to the integration of provenance mechanisms, offering flexibility in the scale and in the precision of the provenance data collected, fostering its relevance to the domain and to the purpose of the specific run.

The contributions of this chapter can be grouped as follows:

(C1) ***Active*** **Provenance.** We introduce the concept of provenance types as the approach for the integration of provenance mechanisms into a streaming workflow system. Enabling provenance can be transparent to the workflow developer and end-user, or fully controllable and customisable by them, depending on their expertise, and on the needs of the domain-scientist and the application's reproducibility, monitoring and validation requirements. After introducing the motivations and the target system, we present in sections 4.3 and 4.5 the framework's specification that allows the realisation and configuration of the mechanisms generating provenance data, especially with respect to the support of contextualisation and precision within streaming operators. We will discuss how the patterns captured by the provenance types relates to S-PROV concepts. Finally, we present new potential for exploiting the lineage messages selectively at run-

time and will introduce a preliminary design for in-workflow rapid analysis.

(C3 - C5) Scale of the provenance records and integration with tools.
The *Active* framework provides a means to configure the production of provenance for the whole workflow application. It allows users and developers to tune precision and the scale of the provenance collected exploiting rules that may depend on domain metadata, activating the production of lineage for a subset of the processed data. We illustrate how the provenance types are integrated by adopting a software engineering solution that facilitates "by-design" their selective combination, fostering portability and adaptation to the application domain independently from the underlying enactment system.

The framework for steering provenance collection introduced in this chapter should enable an effective alliance between practitioners and systems that lead to more reliable and reproducible evidence derived by the execution of scientific workflows. The investigation and the actual implementation is conducted by exploiting an existing data-intensive tool, `dispel4py` [17], a stream-based processing library that is adopted in different context and disciplines.

4.1 Provenance Integration in Workflow Systems

Typically, provenance tracking mechanisms are offered by workflow systems as a complementary module that can be optionally activated. Many approaches delegate the production of lineage traces to fully automated and often centralised components. The Kepler system [93] proposes a system based on a single provenance recorder listening to different events. This can be configured to store traces into files or to a local database. Taverna has a similar approach [72], it offers the possibility of accessing intermediate results for a complete run if the provenance gathering option is switched on. Kepler also introduced the need for a configurable provenance framework in support of workflow execution. It aims at exploiting provenance for monitoring purposes and to allow smart re-runs based on the intermediate data, which gets cached at runtime. Moreover, workflow components can be registered with the recorder in a way to limit

the scope of the lineage produced to only subsets of the workflow graph, consequently reducing the total volume of traces.

Other aspects related to a configurable and tuneable integration of provenance, concern its precision. Precision is determined by how data dependencies are recognised and captured. As already mentioned in the previous chapter, Kepler tries to mitigate the loss in precision by introducing a "read, write, state-reset" (RWS) logic [108], where the output data is considered dependent on all the data received after the last *state-reset* event. Another challenge comes from the Factorial Design [91] approach to workflow and components implementation. It aims at guaranteeing the precision of dependencies of results and parameters, which can be made ambiguous by components that merge or combine multiple input sources. In these circumstances the role of a workflow management system is to support the users in disambiguating these dependencies when they choose to, by providing adequate tools for the precise provenance identification and resolution.

The work pursued by in this chapter enables research-developers to tune and disambiguate the provenance records by applying different solutions. This can be achieved by capturing and re-using general provenance patterns across components and by instrumenting ad-hoc adjustments that can increase the precision in the representation.

Of more recent concern, is the role of data and domain metadata within the execution and the analysis of a workflow run. For instance, systems like *Wings* [149] and *SciCumulus* [118] value the role of domain information, which besides increasing the quality and the relevance of the lineage, can facilitate and improve the effectiveness of monitoring tasks and, as implemented in Wings, a semantic-driven workflow composition. The validation of the semantic association between connected outputs and inputs at runtime, can capture and propagate the metadata across the workflow, triggering different behaviours in adjacent components according to a rules-based approach. The *Active* framework here presented, supports the injection of external namespaces within the provenance types, to assure that the extraction of metadata and the classification of the workflow components is compliant with community managed vocabularies, although it is not limited to that as it allows for user-defined terms and concepts.

Finally, a configurable provenance framework is also the focus of this chapter. We

learn from what has been pursued by the workflow systems previously mentioned and explore ways to introduce more flexibility to the provenance capturing mechanisms. It aims to support developers and users in taking responsibility on their methods and on the way they are used and described. In the next section we introduce the concept of *Agile* data-intensive applications to define the technical characteristic of our target systems. This will be followed by the description of configurable provenance types, showing how they are instrumented in specific applications, activating and tuning provenance-awareness, assuring its relevance to the domain of the data-intensive tasks. It considers eventually a design which foresees a constellation of dedicated provenance *sensors*, offering the pre-analysis of the lineage information, with the ultimate aim of fostering a more interactive and dynamic role for the provenance data.

4.2 *Agile* Data-Intensive Applications

In order to explain the concept of *Agile* data-intensive applications, we start from the principles introduced by Haberfellner *et al.* [152] that distinguish *Agile Systems Engineering* from *Agile Systems*. Briefly, the authors describe the former as a collection of development and project management techniques, aiming at mitigating the side-effects brought by uncertainties during the phases leading to the realisation of a product. The latter, instead, is a property of a system that can respond to changes after its initial adoption and operational deployment. An *Agile System*, should (a) embed the necessary flexible elements that allow for rapid variations, and (b) should be equipped with monitoring functionalities to supporting decision-makers in the identification of situations of special concern. Finally (c), it should include, or at least facilitate the integration, of decision mechanisms by which the system's evolution can be triggered based on costs versus benefits evaluations. Given these premises, we consider that an *Agile* data-intensive system should also offer ways to migrate across several execution modes and architectures, according to application requirements and platform availability. This possibility should be obtained with minimal impact on the application logic and its implementation details, such as coding language, data-sources, imported libraries and methods. This a property which characterises the `dispel4py`

data-intensive library [134]. In this chapter we will show how this class of systems can be further extended by the flexible management and exploitation of provenance, which encompass adaptability to different use cases, including validation of data and methods and results management. Moreover, it should offer ways of providing at run-time feedback about the progress achieved, improving the understanding of the results and stimulating users' active participation. Thereby, we introduce provenance types as a feature that complements and enhance the *Agile* properties of a data-intensive application, fostering rapid variations, monitoring and decision mechanisms performed by human or machines, suggesting immediate benefits. We enable on-demand and data-driven activation of the lineage recording with customisable and changeable features capturing patterns and domain metadata. We consider here an engineering approach that instruments provenance mechanisms at the level of the system's abstractions, leaving the original types specifying the operators and their methods unchanged when provenance is not required. Moreover, we introduce into our design the concept of provenance *sensors*, to provide the mechanisms for the change, before the provenance is stored.

As a concluding note, the support for full reproducibility and re-enactment of a workflow application is a desirable property for a data-intensive tool. It depends on the consistency of data-locations and the exhaustive availability of information on the execution environments. While the provenance data can retain such information, current research on containers, virtualisation and sandboxing is improving the support for technical re-enactment [146, 185] within target infrastructures.

4.2.1 *Active* Provenance: Types and Configuration

A workflow is a program that combines atomic and independent processing elements via a specification language and a library of components. More advanced systems adopt abstractions to facilitate re-use of workflows across users' contexts and application domains [215, 223, 141]. While methods can be multi-disciplinary, provenance should be meaningful to the domain adopting them. Therefore, a portable specification of a workflow requires mechanisms allowing the contextualisation of the provenance

produced. For instance, users may want to extract domain-metadata from a component or groups of components adopting vocabularies that match their domain and current research, tuning the level of granularity. To allow this level of flexibility, we explore an approach that considers a workflow component described by a class, according to the Object-Oriented paradigm. The class defines the behaviour of its instances as their type, which specifies what an instance will do in terms of a set of methods. We introduce the concept of *Provenance Type*, that augments the basic behaviour by extending the class native type, so that a subset of those methods perform the additional actions needed to deliver provenance data. Some of these are being used by some of the pre-existing methods, and characterise the behaviour of the specific provenance type, some others can be used by the developer to easily control precision and granularity. This approach, as indicated in previous work [135, 187], tries to balance between automation, transparency and explicit intervention of the developer of a data-intensive tool, who can tune provenance-awareness through easy-to-use extensions. Moreover, in order to reduce the coding efforts of the expert, we will also show how these extensions have been used to provide sensible default behaviours in a library of ready-made variants. Our conceptual framework, described in detail in Section 4.3, gives to the provenance types the following characteristics.

Preserve native methods: types retain the original naming scheme for methods keeping their parameters and products unchanged.

Offer programmable extensions: code annotations or, as in the current implementation, extended signatures and optionally embeddable methods, delivers for provenance-state management, selectivity and control of provenance precision, enabling the implementation of reusable provenance patterns by research developers.

Handle the extraction of domain-metadata: types handle the automated extraction of metadata according to controlled vocabularies and community standards, as well as experimental and user-driven annotations.

Regulate the destination of the lineage traces: types store traces at tuneable rates, connecting to different storage systems, services or reactive sensors.

Trigger operations on data: types may embed triggers activating messages within the traces. These are used by the workflow management mechanisms, or embedded sensors, to trigger operations on the data concurrently to enactment. For instance, to transfer the data to remote locations or to local caching for rapid re-use.

Finally, provenance types provide the foundation to automatically handle input and output traceability, enable time-stamping and capture contextual information. Although not imposed by the framework, we may distinguish between provenance **Pattern Types** and **Contextualisation Types**, to be used in combination. The former capture provenance patterns by applying rules associated with the events triggered by the ingestion and production of data-streams within components. The latter are used to extract domain metadata from the newly produced *DataGranules*. Domain contextualisation is achieved by adopting standardised metadata formats, specific ontologies or vocabularies, the generation of unique identifiers in compliance with a well-defined schema. External ontologies may be provided by semantic workflow systems such as *Wings* [149], but are not limited to those. New terms can be supported and injected into the provenance traces, serving the need of adding more experimental annotations. Once available, types can be re-used across workflows and made available through a dedicated library. We discuss this in detail in Section 4.3.

In order to enable the user of a data-intensive application to configure the attribution of types, selectivity controls and activation of advanced exploitation mechanisms, we introduce in this chapter also the concept of provenance configuration. In Figure 4.1 we outline the different phases envisaged by framework. In that respect, we propose a configuration profile, where users can specify a number of properties, such as attribution, provenance types, clusters, *sensors*, selectivity rules, *etc.* The configuration is used at the time of the initialisation of the workflow to prepare its provenance-aware execution. We consider that a chosen configuration may be influenced by personal and community preferences, as well as by rules introduced by institutional policies. For instance, a Research Infrastructure (RI) may indicate best practices to reproduce and describe the operations performed by the users exploiting its facilities, or even impose requirements which may turn into quality assessment metrics. For instance, a certain

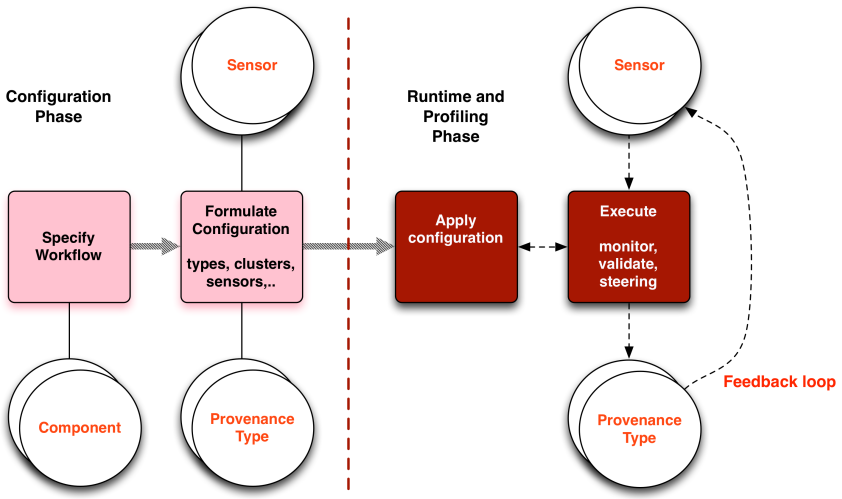


Figure 4.1: Provenance Configuration and Profiling. Phases involving the preparation of a workflow and its provenance-aware execution. Once the workflow is specified, the user formulates a configuration that will be applied when the workflow is initialised. She indicates the provenance types of the components, in order to capture patterns and metadata describing the use of the data flowing through the system. The configuration profile includes the specification of clusters of components and may include provenance sensors associated with these clusters. These read and analyse the provenance traces for profiling purposes, such as monitoring and validation and establish feedback loops into the workflow to trigger steering actions. These could include re-parametrisation or the re-implementation of the component internal functions, or the dynamic re-assignment of a provenance type.

RI would require to choose among a set of contextualisation types, in order to adhere to the infrastructure's metadata portfolio. Thus, a provenance configuration profile play in favour of more generality, encouraging the implementation and the re-use of fundamental methods across disciplines. We will describe this phase in Section 4.5, which is followed by a brief discussion about the technical solution adopted for the dynamic assignment of provenance types (Section 4.7). In Sections 4.4 we will show through a test case on correlation analysis, how contextualisation, assisted tuning and decoupling of the provenance and computational abstractions, increase the usability of the provenance traces, while encouraging the implementation of portable workflow programs. We continue in the next section introducing methods and concepts associated with a `ProvenaneType`.

Provenance Contextualisation Types: Implemented domain specific types extracting metadata from standard formats.	
Type Name and Description	**Behaviour**
`NetCDFType`: applicable to components that deal with a data formats containing array-oriented scientific data, including multiple variables and attributes associated with standard vocabularies and *uid* schemas.	Extracts metadata from data streams represented in *NetCDF* [42]. When the data is accessed from external resources such as files or `OPeNDAP` [199], it guarantees consistency between the *ids* of the ingested data and the lineage entities, by reusing the unique identifiers already expressed in the format.
`SeismoType`: works on seismic data consisting of multiple time-series and represented in any of the formats supported by the *ObsPy* library [43, 104].	Metadata are part of the format and match an experimental metadata and provenance profile for seismology, SEIS-PROV [66].

Table 4.1: A library of reusable contextualisation provenance types. These types are realised in support of applications in climate and seismology. Developers can easily extend the library to support additional domain specific data formats and vocabularies.

4.3 The *ProvenanceType*

The type-based approach to provenance collection provides a generic `ProvenanceType` that defines the properties of a provenance-aware workflow component. It provides a wrapper that meets the provenance requirements, while leaving the computational behaviour of the component unchanged. Types may be combined to represent complex patterns and contextualisations. We may anticipate an interactive system that allows workflow developers to select types from a library and to combine them with the nodes of their workflow. Developers will indicate which types fit better the components they implemented. This should be then recorded, *e.g.* as components' properties in a catalogue, and suggested to the users when they prepare the execution of a workflow with provenance tracking. Alternatives that capture lineage with less granularity, but

that can still collect information and attributes of interest at a reduced overhead, should also be suggested with the indication of the implications of using a less precise type.

As already anticipated, types can be divided in two categories, namely **Pattern Types** and domain specific **Contextualisation Types**. Combining two types of these categories gives the opportunity of capturing provenance patterns while handling the extraction and injection of domain metadata into the provenance stream, decoupling these task from the component's internal logic. In Section 4.4, we will introduce a test case on correlation analysis to demonstrate their use.

A *ProvenanceType* exposes an abstract interface. In the following list we introduce a list of abstract methods whose implementation characterises the specific behaviour of a *Contextualisation* type.

`makeUniqueId`**:** generates an returns an `id`. This can be implemented to adhere, for instance, to best-practices of a hosting research infrastructure.

`extractDataSourceId`**:** this method extracts the `id` from the incoming data, if applicable, to reuse it to identify the correspondent provenance entity. This functionality is handy especially when a workflow component ingests data represented by self-contained and structured file formats. For instance, the *NetCDF Attribute Convention* [41] includes in its internal metadata an `id` that can be reused to ensure the linkage and therefore the consistent continuation of provenance traces between workflow executions that generate and use the same data.

`extractItemMetadata`**:** this function extracts metadata from the data written on the component's specific output port, according to a particular vocabulary. Metadata are injected into the *DataGranules* that describe the new provenance entity.

`addNamespacePrefix`**:** namespace prefixes can be declared in order to map the metadata terms to external controlled vocabularies. They can be used to qualify the metadata terms extracted from the `extractItemMetadata` function, as well as for those terms injected selectively at runtime, as we will describe later in this section. The namespaces will be used consistently when exporting the lineage traces to semantic-web formats, such as RDF.

In order to capture of the lineage of *stateful* operators, thereby implementing a **Pattern** type, and in those circumstances where developers require to explicitly manage the provenance information within the component's logic, the framework offers three additional methods. The first is an abstract method to be implemented by the creator of the pattern type, the others can be used by the developers of the component to add provenance *Data* entities and *wasDerivedFrom* relationships by directly interacting with the *StateCollection* or as new output is produced. We will show in Section 4.6 an example that uses these two methods.

`applyDerivationRule`**:** this method is invoked by each iteration when a decision has to be made whether to ignore or discard the dependencies on the ingested stream and stateful entities, applying a specific provenance pattern. The framework invokes this method every time the data is written on an output port and every time an invocation ends. By capturing these two events, rules can be applied to establish flow and state derivations, defined in Table 3.3, thus, developing common and reusable **Pattern** types. More details on these rules will be covered in Section 4.3.1.

`updateProvState`**:** updates the *StateCollection* with a reference, identified by a lookup tag, to a new *Data* entity or to the current input. The lookup tag will allow developers to refer to the entity when this is used to derive new data.

`write`**:** this is the native write operation triggering the transfer of data between adjacent components of a workflow. Different systems may use different names, we call it *write* for simplicity. Such a method is extended with explicit provenance controls by superimposing a provenance type. We assume these to be ignored when provenance is deactivated. Also this method can use the lookup tags to establish dependencies of output data on entities in the *StateCollection*.

In addition to the `extractItemMetadata`, whose implementation is specific of the provenance type, the last two methods also allows developers to extend the description of the *Data* and *DataGranules* with more metadata at runtime, for further contextualisation. They may indicate details about formats, data location, as well as free-text messages, for instance, to report anomalies, runtime errors *etc.*. Metadata can be anonymous or qualified by the namespaces that are associated with the `addNamespacePrefix`

method. Similarly, developers can include the PROV property `prov:type`, in order provide further typing information to the data produced, *e.g.* a class of an external ontology. Tables 4.2 and 4.1 show a preliminary library of types, where the `SingleInvocationFlow` implements the default *stateless* behaviour. The types have been designed and implemented to satisfy the requirements of a number of applications that we have developed and deployed within the scientific domains of climate and seismology (which motivates the specific contextualisation types). These will be introduced in Chapter 6. We foresee this library to be extended to support additional scenarios and data formats. Their implementation, together with the whole provenance type framework for `dispel4py` is available in GitHub[1].

Provenance Pattern Types: A collection of implemented basic types that handle common provenance patterns.	
Type Name and Description	**Behaviour**
`SingleInvocationFlow`: component that presents *stateless* input output dependencies; *e.g.* the processing element of a simple I/O pipeline.	*FlowDerivations* are established between the output and input data received in a single invocation.
`AccumulateFlow`: component whose output depends on a sequence of input data; *e.g.* computation of periodic average.	*FlowDerivations* are established between the data read across more invocations until output is produced.
`SlideFlow`: component whose output depends on computations over sliding windows; *e.g.* computation of rolling sums.	*FlowDerivations* are established between each output and the inputs in the current window. It invokes a *flow-reset* event considering the window's *length*, which is set in the provenance configuration.
`Nby1Flow`: component whose output depends on the data received on all its input ports in lock-step; *e.g.* combined analysis of multiple variables.	*FlowDerivations* are established between the output and the input data received on all the input ports for the same index (lock-step).

[1] `https://github.com/aspinuso/dispel4py/blob/master/dispel4py`

Provenance Pattern Types: A collection of implemented basic types that handle common provenance patterns.	
Type Name and Description	**Behaviour**
`AccumulateStateTrace`: component that keeps track of the updates on intermediate results written to the output after a sequence of inputs; *e.g.* traceable approximation of frequency counts [180] or of periodic averages.	It updates the *StateCollection* every time an intermediate result is produced. It establishes *FlowDerivations* between the result and all the data received in input and *StateDerivations* between all the subsequent results. It invokes a *flow-reset* event after each output is produced.
`ASTGrouped`: (Accumulate State Trace Grouped) this type manages a stateful operator with grouping rules; *e.g.* a component that produces a correlation matrix with the incoming coefficients associated with the same *sampling-iteration* index, see sections 4.4 and 4.6.2.	At each invocation, if no data is produced, it updates the *StateCollection* with a provenance entity that derives from the current input using a lookup term which combines the value of the grouping property and the name of the input port. *StateDerivations* are established between state updates on the same lookup term, until an output is produced, see Figure 4.8.
`IntermediateStatefulOut`: stateful component which produces distinct but interdependent output; *e.g.* detection of events over periodic observations or any component that reuses the data just written to generate a new product, see Section 4.6.3	When data is produced on a specific `stateful-port`, set as parameter, it establishes *FlowDerivations* between the output and all the input data read after the last write, like in the `AccumulateFlow`. The *StateCollection* is updated with the data (and metadata) just returned. When data is produced on any other port, a *FlowDerivation* and a *StateDerivations* are respectively established between the output and the input data of the current invocation and with the content of the *StateCollection*, see Figure 4.3.

Provenance Pattern Types: A collection of implemented basic types that handle common provenance patterns.	
Type Name and Description	**Behaviour**
`ForceStateless`: It considers the outputs of the component dependent only on the current input data, regardless from any explicit state update; *e.g.* the user wants to reduce the amount of lineage produced by a component that presents inline calls to the *updateProvState* (*e.g.* see Listing 4.3), accepting less accuracy.	The adoption of this type forces capturing the data dependencies exclusively according to the *SingleInvocationFlow* pattern.

Table 4.2: A library of reusable provenance patterns types. These types are realised to support a number of applications and use cases encountered during the experimentation with the framework and for the realisation of real workflows for Earth sciences. Developers can extend the library to support additional patterns.

4.3.1 Managing Flow and State Derivations

The proposed provenance type system allows developers to tune the generation of provenance statements describing the behaviour of a streaming operator. They can handle stateful provenance entities via the explicit use of the `updateProvState` method establishing new derivations. These are divided into two categories: *FlowDerivation* and *StateDerivation*, as defined in Table 3.3.

To handle the data dependencies involved into these derivations, a provenance type holds two different data structures. A list containing `ids` referencing to provenance entities related to data read across one or more invocations, and a dictionary representing the operator's internal provenance state. Depending on the implementation, these data structures can be held in memory or accessed from a local resource. The provenance state contains `ids` of data entities that may be potentially reused during the computation. These are associated with a lookup term and can refer to data that has been received in input or newly generated, or to an entity that indicates an intermediate and traceable update of a component's internal data structure. A provenance type uses

these information to establish the derivations at runtime.

In order to support reusable lineage patterns, the *applyDerivationRule* method specifies rules that are triggered by different events, which we describe as follows:

`write-event`**:** it occurs when a component "writes" data on any of its output ports. Before the actual writing, it triggers a rule that establishes and describes the provenance derivations associated with the new output.

`end-invocation-event`**:** occurs at the end of an invocation, just before the process starts reading again. It provides indications whether the invocation had previously produced any data, for instance, by setting a boolean value to a *void-invocation* variable.

Depending on the pattern implemented by the type, the rules may update the *State-Collection* (`updateProvState`), ignore the inputs (`ignorePastFlow`), or reset the current data dependencies (`discardInFlow`, `discardState`). As a preliminary example, the provenance of a process that computes the average calculation over a stream of input values is captured by the `AccumulationFlow` type. In this case, *FlowDerivations* will be discarded after each *end-invocation-event* for invocations that produced output, which is indicated by the *void-invocation* set to `False`. That is, when output was produced. Table 3.3 includes the `SlideFlow` type that captures provenance of components processing over a sliding window of a certain number of input elements, this can be simply obtained by triggering a (`discardInFlow`) after each *end-invocation-event*, provided the length of the window (or a time-stamp and a *delta*, in implementations for real-time sliding). We will show more examples in Section 4.6, when we will discuss in detail the test case on correlation analysis workflow.

As we have already anticipated, the derivations are produced at runtime. In Chapter 5 we will cover representation, storage and access of the information produced by the *Active* framework in a provenance management system designed with particular attention to the characteristics of the S-PROV model. As further research topic, we should mention the recent specification of the PROV-Template [190] for the automatic generation of provenance in PROV format. It consists of a system based on templates encoded as PROV documents that describe recurrent provenance patterns and that are instantiated

by using template's variables bindings. From preliminary considerations of the author [51] and personal communication with Paolo Missier, there is an emerging interest in evaluating the adoption of templates in different context, including computational systems. In that respect, we foresee that the provenance contextualisation and patterns types could be combined with the templates to produce the bindings needed for the generation of the traces.

Listing 4.1: PROV Template document in PROV-N notation that declares a `bundle` to represent a named set of provenance descriptions that characterise the `SingleInvocationFlow` pattern. Bundles are also entities and have their own provenance. For instance, in line 3, the entity `var:SingleInvocationFlow` further describes the bundle, while line 4 attributes it to the `var:user` agent that executed the workflow.

```
1 document
2   agent(var:user, [vcard:uid="var:name"])
3   entity(var:SingleInvocationFlow, [prov:type="ex:StatelessPattern"])
4   wasAttributedTo(var:SingleInvocationFlow, var:user, -)
5   bundle var:SingleInvocationFlow
6     agent(var:Component, [prov:type='s-prov:Component', tmpl:linked='var:
          ComponentInstance'])
7     agent(var:ComponentInstance, [tmpl:linked='var:Invocation', prov:atLocation='var:
          worker'])
8     actedOnBehalfOf(var:ComponentInstance, var:Component, -)
9     used(var:Invocation, var:DataIn, -)
10    used(var:Invocation, var:ComponentParameters, -)
11    wasDerivedFrom(var:DataOut, var:DataIn, -, -, -)
12    hadMember(var:DataOut, var:DataGranule)
13    wasGeneratedBy(var:DataOut, var:Invocation, -)
14    wasAssociatedWith(var:Invocation, var:ComponentInstance, -)
15    wasAssociatedWith(var:WFExecution, var:user, var:Workflow)
16    wasAssociatedWith(var:WFExecution, var:Component, var:Workflow)
17    activity(var:Invocation, tmpl:startTime, tmpl:endTime, -)
18    activity(var:SingleInvocationFlow,-,-,-)
19    entity(var:ComponentParameters)
20    entity(var:DataGranule, [var:term_="var:n"])
21    entity(var:Workflow, [prov:type="provone:Workflow"])
22    entity(var:DataOut, [d-prov:port='var:op', prov:type='s-prov:Data', tmpl:linked='
          var:DataGranule', tmpl:linked='var:DataIn'])
23    entity(var:DataIn, [d-prov:port='var:ip', prov:type='s-prov:Data'])
24  endBundle
25 endDocument
```

These will comply with the requirement of a specific PROV profile, which is expressed as a collection of templates, that may expect specific metadata vocabularies and seman-

tic characterisation, and provenance relationships to be explicitly populated during the computation. For instance, we foresee that the lineage produced by a provenance type that, depending on its complexity, may consists in one or more documents, could be transformed into a binding for the instantiation of the template that describes the behaviour of the workflow component. However, the *Active* framework also provides dynamic metadata enrichment and tuneable precision through flexible state management, which may lead to less predictable situations. Evaluating this requirement with the template system, given the growing interest, is a subject for further investigation. As an example, the templates associated with `SingleInvocationFlow` and the `IntermediateStatefulOut` patterns are shown respectively in the Listing 4.1 and Figure 4.2, and in Figure 4.3.

In the next section we introduce, as a test case, a correlation analysis application that could be used to monitor indexes coming from the financial markets, as well as physical values received from sensors. We will use this test case to explain the properties of the framework.

4.4 Test Case: Correlation Analysis Workflow

The elements of the *Active* provenance framework will be illustrated incrementally, demonstrating their effects on a sample Correlation Analysis Workflow (CAW). For the discussion we will make use of the visualisation provided by the `S-ProvFlow` tool, which will be explained in detail in Chapter 5.

Correlation analysis is a common investigation technique used to infer statistical relationships between a large number of variables. It is commonly used to convey information to decision making systems. For instance, applications are found in Earth science for the monitoring of CO_2 sequestration sites [131] and in the field of seismic interferometry [103]. The latter uses its potential for revealing the structural characteristics of a given medium, thanks to the reconstruction of its impulse response properties. Correlation can also be adopted in financial studies to improve effectiveness of investment strategies based on near real-time updates on markets quotes, in support of

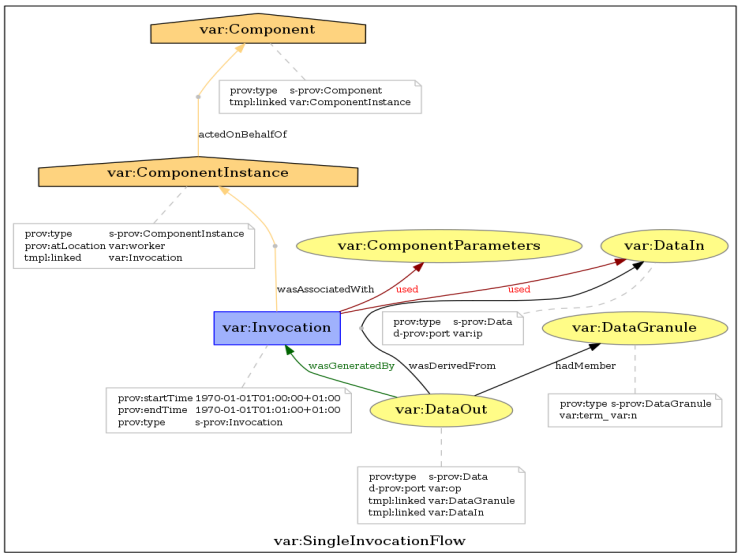


Figure 4.2: Visualisation of the template for the `SingleInvocationFlow` type. Associations witht user, workflow and implementations are omitted for clarity of representation.

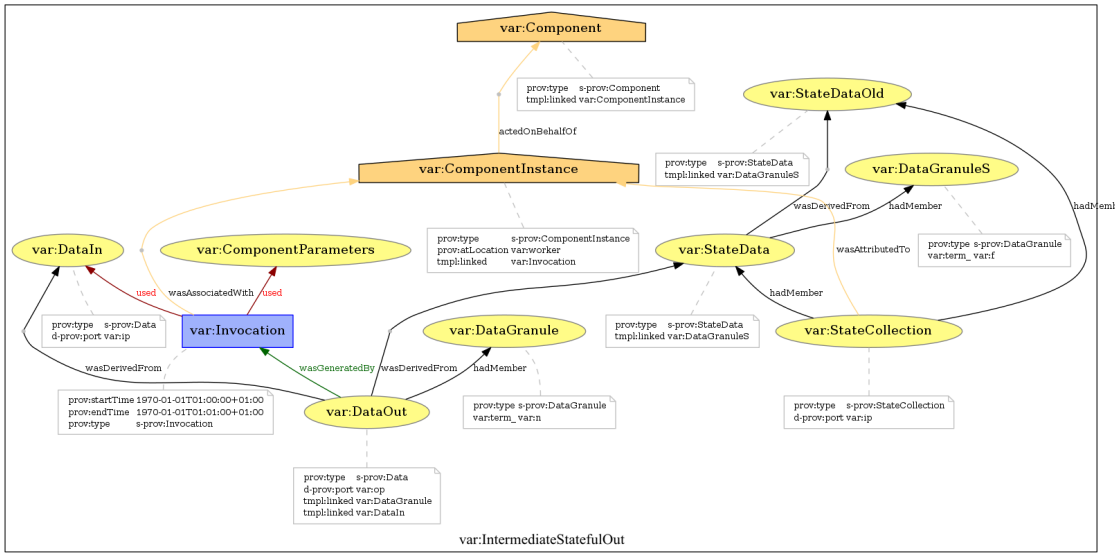


Figure 4.3: Template representing the `IntermediateStatefulOut` Type. Updates to the *StateCollection* are traced across invocations as well as flow dependencies. The provenance type can also capture one additional relationship, `wasGeneratedBy(var:StateData,var:Invocation)`, which occurs when a state update is performed by the current invocation.

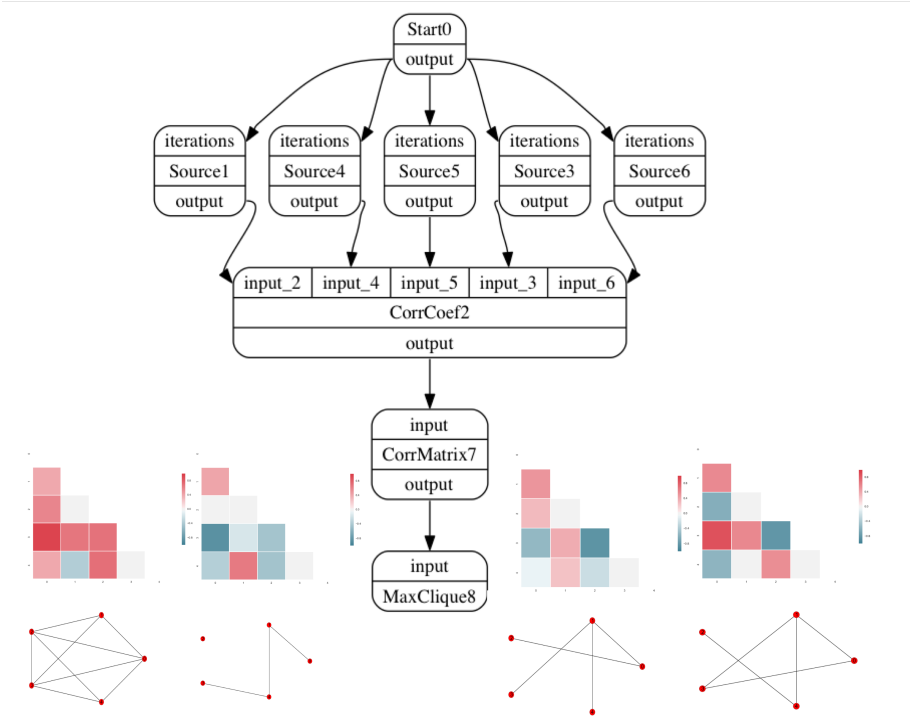


Figure 4.4: Correlation analysis workflow. The workflow generates a sequence of correlation heat-maps and max-cliques, given a correlation threshold, for a configurable number of real-time sources. The figure shows the heat-maps and the whole graph related to each map from which the cliques are then calculated.

high frequency trading (HFT) systems [205]. We describe here a real implementation of such workflow, shown in Figure 4.4, that has been developed with the `dispel4py` streaming library. The workflow is also available as *Jupyter Notebook* [47] notebook page[2].

The CAW correlates (`CorrCoef`) batches of variables (`Source`) across multiple iterations, and produces and visualises, as a heat-map, the correlation matrix (`CorrMatrix`). Eventually it computes the correlation graph obtained by including only the correlation coefficients above a certain *threshold* and finds and visualises its max cliques (`MaxClique`), that is, the fully connected subgraph with the highest number vertices. The workflow computes over data-streams initiated by the `Source` components. For

[2] `https://github.com/aspinuso/dispel4py/blob/master/d4py-prov-CAW.ipynb`

each *sampling-iteration*, a `Source` reads the variable's values at a certain *sampling-rate* and organises them into batches of a determined *batch-size*. Once a batch is ready the component writes it to the data-stream. Each variable is associated with a single `Source`. The sampling-rate and size of the batch are parametrisable and may vary at runtime. The logical components `CorrCoef`, `CorrMatrix`, `MaxClique` can be parallelised in a concrete deployment onto a target computational resource. To allow concurrency the inputs of the first two components are grouped by the *sampling-iteration* index provided by the `Source`. Meaning that each instance will receive all the data belonging to one or more sampling iterations. This makes `CorrCoef` and `CorrMatrix` two stateful components, in the sense that their instances need to preserve and combine input data across iterations. The `MaxClique` component does not require any grouping and can be easily parallelised into many instances. Although we may consider this component as stateless, we will show that it hides dependencies which could be captured by using a stateful pattern (Section 4.6).

To continue with the general properties of the framework, in the next section we will introduce our provenance configuration profile. It allows users to specify different provenance setups for a workflow application and foresees possibilities for the rapid exploitation of the lineage traces, as part of a comprehensive *Active* framework. To explain its use and effects we will refer to the CAW test case.

4.5 Configuration and Selectivity

Once the provenance types have been defined, these are used to configure a workflow execution to comply with the desired provenance collection requirements. Table 4.3 summarises the configuration possibilities. We will explain them starting from the framework's elements that enable users to prepare the workflow for provenance tracking.

`configureProvRun`: with this method, the users of the workflow provide general provenance information on the attribution of the run, such as *username, executionId, description, workflowName*, and its semantic characterisation *workflowType*. It allows users to indicate which provenance types to apply to each component

and the belonging cluster. Moreover, users can also choose where to store the metadata, locally in the file system or in a remote service or database. Lineage storage operations can be performed in bulk, with different impacts on the overall overhead and on the experienced rapidity of access to the lineage information. Once fired, the method initialises the graph according to the configuration setup following an approach based on dynamic class composition, which we describe in Section 4.7. Additional details on the proposed remote provenance storage and access service will be provided in Chapter 5.

Provenance Configuration Profile	
Configuration Property	**Description**
Components Types	One or two types namely assigned to each workflow component.
Assign Components to Clusters	Components can be grouped into clusters, `s-prov:prov-cluster`, for instance, enabling their semantic classification.
Selectivity-rules	Metadata rules associated with a specific named component to selectively enable the generation of traces for a *Data* element.
Transfer-rules	Metadata rules associated with a specific named component to selectively trigger the transfer of the output file associated with a *Data* element to a target location.
Namespaces	Namespaces of the vocabularies or ontologies used in the provenance trace.
Provenance Storage mode	Users can choose if the provenance can be stored as files, sent to services, `save_mode`.

Table 4.3: Provenance Configuration. Most relevant options to configure the production and handling of provenance. General information about the run, *e.g.* `username`, `description`, `workflowName`, `workflowType`, `runId` are also included.

`Selectivity and Transfer rules`: By declaratively indicating a set of *Selectivity* and *Transfer* rules for every component, users can respectively activate the collection of the provenance for particular *Data* elements or trigger transfer oper-

ations of the data to external locations. The approach takes advantage of the contextualisation possibilities offered by the provenance types. The rules consist of comparison expressions formulated in JSON that indicate the boundary values for a specific metadata term. Such representation is inspired by the query language and selectors adopted by a popular document store, MongoDB [35]. We have used extensively this database technology to implement our provenance repository, as we will discuss in Chapter 5. As an example, the following rule will produce provenance for the `CorrCoef` component of the CAW, only if the correlation coefficient, `rho`, is greater than 0.

```
{ "CorrCoef": {
    "rules": {
      "rho": {
        "$gt": 0
}}}}
```

This second example indicates that a user wants to produce and store provenance information only for those *graphs* where, according to the current threshold and the `order` of the *maxclique*, at least half of the variables are part it.

```
{"MaxClique": {
    "rules": {
      "order": {
        "$gt": (#variables/2),
        "$lt": (#variables)
}}}}
```

In Listing 4.2 we show a JSON representation of a possible profile for the CAW, as it is interpreted by the implementation for `dispel4py`. The `componentsType` property assigns a combination of provenance pattern and contextualisation types (`s-prov:type`), and clusters (`s-prov:prov-cluster`). The former are expressed as a tuple of one or two elements, typically a pattern and a contextualisation type. When applicable by the pattern type, these can be further configured by specifying the name of the output port that delivers stateful data (`s-prov:stateful-port`), which will be reused by the component's instances. Specific contextualisations could be used to match a certain provenance profile for a target community, in order to obtain measurements in desired units or to use appropriate terminology and precision of the traced produced. A cluster instead, assigns a specific semantic concept to the component or to a group of compo-

nents.

This allows to link the component to tasks and roles defined by an external domain ontology (`prov:type`), or to custom concepts defined by the user. We will discuss this property again in Section 4.8.2. Combining user-defined and ontology concepts with process' behaviour and domain metadata extraction, aims at reducing the gap between the processing and conceptual focus captured by the traces, in a semi-automated way, without overlooking the content, as indicated by Alper, *et al.* [92].

When *selectivity rules* are applied, in order to bridge the gap in the lineage trace, *FlowDerivations* are established between the data element produced by the next workflow component and the dependencies of the data that did not pass the selectivity rule. In such scenario, we can take advantage of an extension of PROV [123] that proposes the inclusion of concepts to express uncertainty of provenance. For instance, the *FlowDerivation* generated to compensate a gap in the lineage, will present the property `up:assertionType=up:Incomplete`. Such property should be visually recognisable in an interactive tool that allows for lineage navigation. We consider the more extensive specification and support of such annotations, in combination with additional selective use cases, as a relevant topic for future work.

Listing 4.2: JSON example showing one possible provenance configuration for the CAW. The `CorrMatrix` receives two provenance types to handle provenance in grouped operations and the extraction of stock exchange metadata.

```
configuration = {
  "provone:User": "aspinuso",
  "s-prov:description": "correlation of four continuous variables",
  "s-prov:workflowName": "CAW",
  "s-prov:workflowType": "hft:correlationAnalysis",
  "s-prov:workflowId": "190341",
  "s-prov:save_mode":  "service",
  "s-prov:componentsType":
  {"MaxClique": {"s-prov:type": (IntermediateStatefulOut,),
                 "s-prov:stateful-port": "graph",
                 "s-prov:prov-cluster": "hft:StockAnalyser"},

   "CorrMatrix": {"s-prov:type": (ASTGrouped, StockType,),
                  "s-prov:prov-cluster": "hft:StockAnalyser"},

   "CorrCoef":   {"s-prov:type": (SingleInvocationFlow,),
                  "s-prov:prov-cluster": "hft:Correlator"}}}
```

Applying similar rules for selective transfers allows the user to activate tailored data-movement mechanism, within the workflow system itself or within external observers, that enable data-driven operations on intermediate results concurrently to the workflow execution. This general functionality enables rapid distribution of the data across decoupled contexts and infrastructures. For instance, to feed post-processing activities being performed on other infrastructures or allowing for rapid data inspection within user-friendly environments, such as web portals and interfaces. Moreover, transfer *destinations* can be captured by the provenance, keeping track of the information about the "first-known-destination" of a copy of the data, in addition to its principal location. We have experienced the latter in a real scenarios where our framework could show evidence of the progress of an HPC simulation, triggering mechanisms to accommodate data transfer and visualisation across protocols and security boundaries [97]. Such oversight of computationally expensive processes enables the expert user to decide whether the results will be useful, and if necessary curtail the run early, thereby saving costs and energy. This shows an example of the *Active* exploitation of the lineage data, in addition to the typical use cases involving offline validation and reproducibility. Finally, the proposed framework allows practitioners to take advantage of provenance collection mechanisms in a controllable way. This fosters incremental adoption, increases confidence and promotes awareness of the importance of provenance exploitation, keeping it informative and often avoiding the generation of redundant and non-informative traces.

The availability of different configurations for the same workflow is of great use when the application is adopted, sometimes even developed, at different stages of a research investigation, or executed as a production system. In the first scenario users start with identifying incrementally the required metadata terms which are relevant for their experiments, allowing also some inconsistency in the dependencies for the sake of experimentation.

More formal metadata and precision is needed when the research gets closer to its final results. That is, when the need for outreach and reproducibility by other peers becomes increasingly likely and thereby provenance need to be complete, precise and rigorous, as in the case of publication and curation of the scientific results in citable

data repositories. In production systems instead, provenance information can feed automated quality check procedures that require specific metadata information and dependencies traceability to monitor routine operations. Data-managers are able to tune the impact of provenance generation on the infrastructure's resources and performance, especially in near real-time system, by combining the available provenance types with a set of *selectivity-rules* to narrow the focus of the lineage onto relevant situations. Similarly, the possibility to inject rules that trigger the transfer of a *Data* element to external resources, can ease the access to the data or feed parallel complementary processing tasks, which belong to an independent execution context not controllable by the data-intensive application.

4.5.1 Clustering in the CAW

The configurable provenance setup makes sure that the components of the workflow can be tagged, for instance, to group them within a specific macro task or role, possibly described by a domain specific ontology. Technically, it utilises the `prov:type` property, thereby taking advantage of the PROV framework to extend the description of the involved component, and their instances, with additional semantics.

To show an example on how this could in the CAW for visual-analytics purposes, we refer again to Listing 4.2. Here, we assign the `CorrMatrix` and the `MaxClique` to the `hft:stockAnalyser` cluster, and the `CorrCoef` component to the `hft:Correlator`. Now, we assume that also the `Source` components are mapped to the specific stock-market to which they belong, say: `FTSEMIB, TSE, NASDAQ`. Given this setup, we can produce a view on the ongoing computation that organises the information in macro areas, which can be then visually explored, see Figue 4.5.

In this specific case the view, which has been build interactively with the Bulk Dependencies Visualiser (BDV) (Section 5.7), is offering information about the total size of the data exchanged between the components' instances, grouped by their semantic tags. To be noted that the figure does not show the instance of the `Start` component, whose role is simply to bootstrap the computation by sending to all the `Source` components the number of sampling iterations to be performed. This exclusion is obtained by

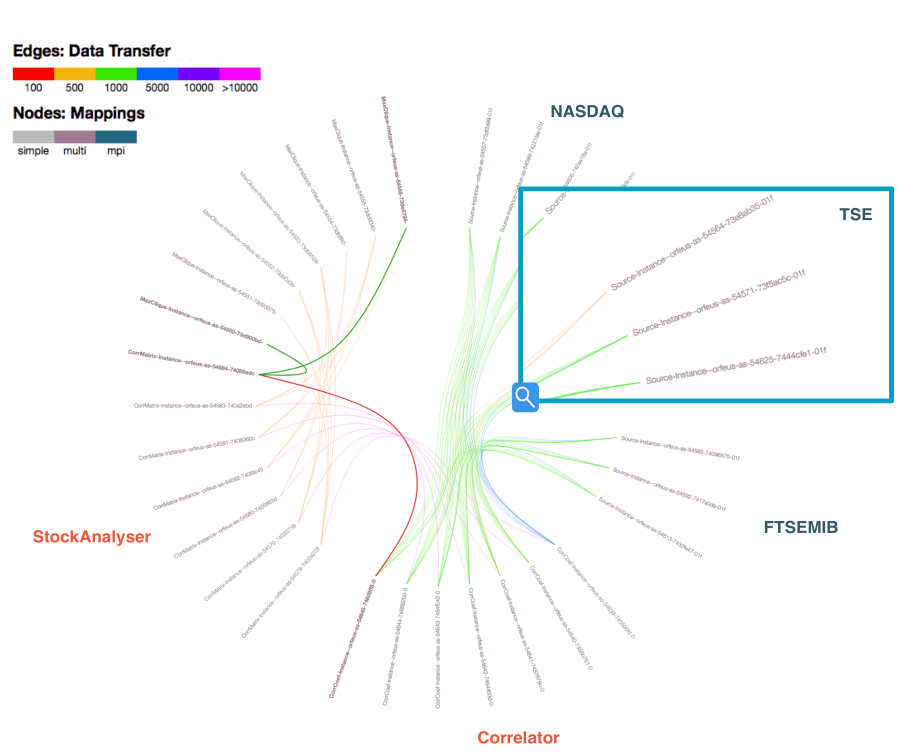


Figure 4.5: Visual-analytics of provenance clusters for the CAW application. Vertices of the graph represent the components' instances grouped by their cluster. Thick red and green edges represent respectively incoming and outgoing connections. The legends classify the amount of data exchanged and the type of enactment of each instance through colour coding. In the example, the low data transfer produced by one of the TSE sources is highlighted by the light red edges. This may suggest a potential issue to be investigated, which may be attributed, for instance, to an high latency during the sampling.

explicitly declaring a selectivity rule, for instance, that expects the property `iteration` associated with the output to be lower than `0`. A condition that obviously will never occur. In the next section we show with concrete examples, how the framework is used to capture the lineage of the CAW application.

4.6 Fine Grained and Tuneable Precision in the CAW

We have shown in Section 4.3.1, how through the management of *FlowDerivations* and *StateDerivation* associated with the events occurring in iterative invocations (`write-event`, `end-invocation-event`), we can set the rules describing different provenance patterns. However, there may be cases where a type alone may not be enough to capture all relevant dependencies for a complex operator, creating ambiguous or wrong traces. The undesired effect is to reduce the usefulness of provenance as a route to understanding processes, and as a mean for the management of the results, thereby making provenance less effective. We will look at these different scenarios by analysing the lineage requirements of the components of the CAW application.

4.6.1 Correlation Coefficient - `CorrCoef`

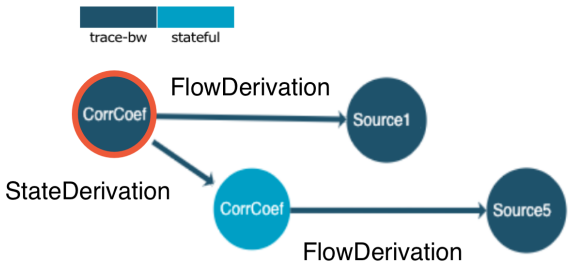


Figure 4.6: Correlation Coefficient. Fine-grain data derivations of the correlation coefficient of two batches of variables. Starting from the circle with the red contour, which represents the coefficient produced by the `CorrCoef` (indicated on the circle's label), we can trace backwards the dependencies on variable 1 (produced by the `Source1` component) and variable 5 (produced by the `Source5` component). Variable 5 is traced though a *StateDerivation* and a *FlowDerivation*. The former is established between the correlation value and a stateful provenance assertion that was explicitly included in the *StateCollection* (light-blue circle) when receiving the variable 5 in a past invocation.

This is a grouped component on the *sampling-iteration* index, meaning that each instance receives all the batches produced for one or more iterations. At each invocation,

the operator adds the incoming batch to a list of variables' batches related to the input's iteration index, and it outputs all the correlation coefficient between the input and the elements already in the list. In such scenario, neither we can associate any common windowing function [207], nor we can rely on provenance types that take into account grouping rules, such as the `ASTGrouped`. The arrival of the batches is unpredictable and the derivations associated with each correlation coefficient should exclusively include the batches of the two variables, the one just read and the one that has been previously stored in the component's list.

Listing 4.3: Function in Python-style pseudocode computing the correlation of input batches for the `CorrCoef` streaming operator. The function receives `data` in input, which is a tuple of the kind: `(index,var,batch)`, where the `index` is the *sampling-iteration* index. This is used to compute the coefficient `rho` with all the other batches (line 10). Then the input is added to the list associated with its iteration index (line 17) and to the provenance state, as a reference to a new entity, which is described by a new dictionary of metadata (line 21).

```
1  def CorrCoef(data):
2
3   # For each new sampling iteration index create a new list of tuples
4   if data.index not in iterationList:
5       iterationList[data.index] = [ ]
6
7   # Computes correlation coefficient between the input
8   # and the elements in the list of the same sampling iteration
9   for tup in iterationList[data[0]]:
10    rho=compute_corrcoef(tup.batch,data.batch)
11
12   # Write data on the output port, add custom metadata and declares dependencies.
13   # method's use: write(port,data,metadata,dependencies)
14   write('output',rho,metadata={'iteration':data.index},dep=['var_'+tup.var+"_"+tup.index])
15
16  # Add the tuple to the list of the specific iterationIndex
17  iterationList[data.index].append(data)
18
19  # Create a new provenance entity in the StateCollection.
20  # method's use: update_prov_state(lookup-term,data,metadata)
21  update_prov_state('var_'+data.var+"_"+data.index,data,metadata={'batch':data.batch})
```

The `ASTGrouped` type, would capture provenance traces that will not always bear precise information because for each coefficient, besides the current sampling iteration, we need to take into account also the dependency on all the other batches in the list. Thus, to capture this behaviour a more detailed knowledge of the internal logic is

needed by the provenance system.

The *Active* provenance framework tries to mitigate these situations thanks to the flexible management of the *StateCollection*. The `updateProvState` method of the API combined with explicit provenance lookups can be used with little contribution from the developer to capture a precise provenance scenario for the component. Figure 4.6 illustrates the lineage trace for the `CorrCoef` component, whose implementation is shown in Listing 4.3.

The visualisation is produced with the Monitoring and Validation Visualiser tool (MVV) that will be presented in more detail in Section 5.6. In this example the workflow developer increases the level of precision by explicitly introducing a new element in the provenance state (`updateProvState`). She composes a lookup name and assigns it to an intermediate *Data* entity that represents and describes the input with a new metadata dictionary. When calling the `write` function, the right entity can be retrieved to describe the derivation that qualifies the *wasDerivedFrom* relationship of the new coefficient.

However, creating a new provenance entity for each incoming batch of variables can be avoided if no additional description of the input is needed. The developer can choose to instruct the active framework to assign the lookup term to the input just read, whose provenance entity was previously generated. This choice will not allow the developer to update the provenance entity of the input. This is enforced in order to maintain the consistency of the context expressed by each component, besides avoiding to handle potential synchronisation issues and expensive update operations on a provenance database, that receives lineage data at runtime. The effect of this choice on the trace is shown in Figure 4.8.

4.6.2 Correlation Matrix - `CorrMatrix`

The `ASTGrouped` becomes useful when we want to automatically capture the provenance traces for the `CorrMatrix` component, that produces the complete correlation matrix for each sampling iteration. This component can be parallelised by grouping, thereby, telling the computational workflow to send always to the same instance of the

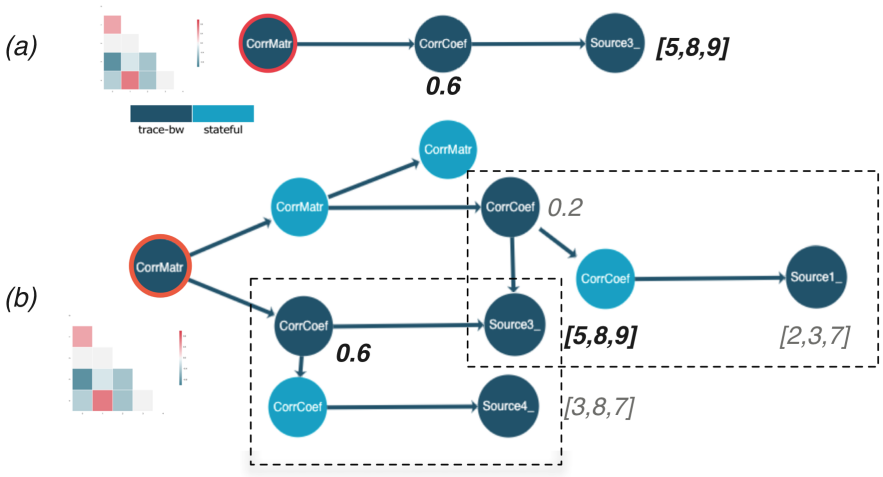


Figure 4.7: Correlation Matrix. Data derivation trace for the correlation matrix of a sampling iteration. Starting from the red and dark-blue circle, we can trace backwards the dependencies on all the correlation coefficients which are part of the matrix. (a) Shows a trace captured by considering all components stateless, while (b) show a trace produced when introducing *StateDerivations*. These are captured by the `ASTGrouped` type for the `CorrMatrix` and by explicit use of to the `updateProvState` method in the `CorrCoef`. The dotted lines enclose the provenance of two different correlations that used the same variable batch `[5,8,9]` (produced by the `Source3` component).

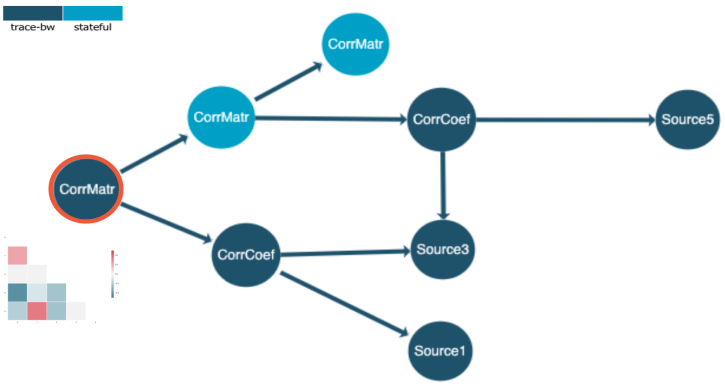


Figure 4.8: Correlation Matrix - Simplified data derivation trace for the correlation matrix. With respect to the trace displayed in Figure 4.7, here the derivations involved in the generation of the correlation coefficients directly link to the input data coming from the `Source` components.

component all the data related to the specific iterations. The matrix associated with a specific iteration is update sequentially. The provenance keeps track of the updates performed on each matrix, as shown in figures 4.7 and 4.8, taking into account the group to which the input data belongs. Listing 4.4 shows how the implementation of the `applyDerivationRule` for the `ASTGrouped` type.

Listing 4.4: Function `applyDerivationRule`, in Python-style pseudocode, for the `ASTGrouped` type. When writing the type establishes new *StateDerivations* between the current output and a new entity associated with the intermediate state. A hash function produces a lookup term based on the input data on `iport` and the GROUPING term. Before reading again, if the `end-invocation-event` occurs and no output was produced, the provenance state is updated with the data entity referring to an intermediate state for the specific sampling iteration. The new assertion will have *FlowDerivation* only with the input just received (`ignorePastFlow()`). If the `end-invocation-event` occurs and data was produced, all input dependencies are discarded (`discardInFlow()`). The state is traced by the `update_prov_state`.

```
1  def applyDerivationRule(event,void_invocation,data,metadata,iport,oport):
2   #Before writing
3   if (event=='write-event'):
4     lookup= hash(getGroupingProperty(iport,GROUPING))
5     setStateDerivation(lookup)
6
7   #Before reading and data was produced
8   if (event=='end-invocation-event' and void_invocation==False):
9     discardInFlow()
10    discardState()
11 #Before reading and no data was produced
12  if (event=='end-invocation-event' and void_invocation==True):
13    lookup=hash(getGroupingProperty(iport,GROUPING))
14    ignorePastFlow()
15    update_prov_state(lookup,data,dep=[lookup])
16    discardInFlow()
17    discardState()
```

4.6.3 Threshold Graph and Max Clique - `MaxClique`

Another interesting case is the application of rules regulating the dependencies within a single invocation. We can take as an explanatory example the `MaxClique` component.

As its name suggests, such component reads the correlation matrix and, given a correlation threshold, produces a graph and finds its max-cliques, that is, the subgraphs with maximum number of connected vertices. The graph and cliques are returned on different ports, respectively `graph` and `cliques`. For each sampling iteration the computation occurs within the same invocation. We do not enter into the detail of the way the max-clique is calculated, this being a well known NP-hard problem that goes beyond the scope of this section.

Looking at this component with a precise provenance eye, it appears sensible to capture the dependency between each clique and the pruned graph. Figure 4.9 shows the same provenance trace from two different directions; the derivation trace for a max-clique (a) and the data derived from the correlation matrix (b). The trace is produced by creating a stateful entity for the data delivered through the `graph` output and establishing a *StateDerivation* between this and the subsequent results that are written to the `cliques` port. In Table 4.2, we offer this behaviour as a reusable provenance pattern type `IntermediateStatefulOut`. Users and research-developers, for instance assisted by an interactive workflow dashboard system [147], indicate in the provenance configuration an output port to be considered as stateful, as shown in Listing 4.2 (`s-prov:stateful-port`). After each *end-invocation-event*, the state is cleared, preparing for the computation of the next clique. We show in Listing 4.5 the current implementation of the `applyDerivationRule` for the `IntermediateStatefulOut` pattern.

The support of this provenance pattern facilitates the immediate access and discovery of relevant dependencies in a validation and result management tool. For instance, if we would have used the `SingeInvocationFlow` the interactive navigation of the lineage graph would have not allowed us to reach the graph directly from the clique, as shown in Figure 4.9 (a).

We have demonstrated how the customisation of the `applyDerivationRule` method allows for the representation of reusable patterns. Though, this is not sufficient in more complex scenarios, where instead we adopted optionally embeddable methods and extended signatures of the provenance type. We will now continue illustrating the technique adopted for the dynamic attribution of the provenance types.

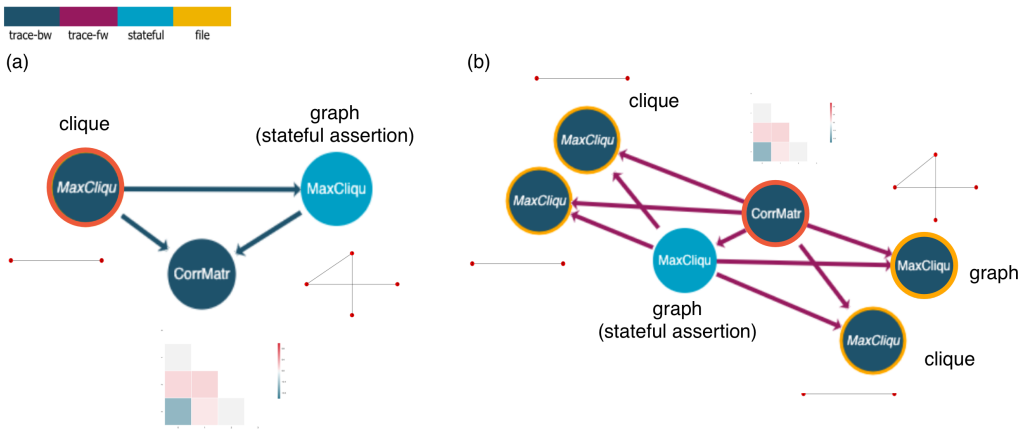


Figure 4.9: MaxClique and `IntermediateStatefulOut` pattern. Bidirectional lineage traces showing respectively (a) the derivations involved in the production of a clique (red-circle) and (b), starting from the correlation matrix (red-circle), a forward navigation of the dependency graph shows the set of all the cliques associated with the intermediate graph. Yellow circles state that the provenance entity links to a concrete data-resource, *i.e.* the files of the cliques' images. The visualisation of the traces is interactively produced with the `S-ProvFlow` system.

Listing 4.5: Function `applyDerivationRule`, in Python-style pseudocode, for the `IntermediateStatefulOut` type. The type has an internal variable indicating on which port stateful data is written (`STATEFUL_PORT`). The provenance state collection is then updated with the new assertion after each write event on the stateful port. A write event on a different port will be handled by ignoring old accumulated inputs and establishing a *StateDerivation* with the current state and a *FlowDerivation* with the input just received. At the end of the invocation if results have been produced the state and all the dependencies are discarded. For the remaining case all the dependencies are kept for the next invocation.

```
1  def applyDerivationRule(event,void_invocation,iport,oport,data,metadata):
2    #Before writing on a stateful port
3    if (event=='write-event' and oport == STATEFUL_PORT):
4      update_prov_state(STATEFUL_PORT,metadata)
5    #Before writing
6    if (event=='write-event' and oport != STATEFUL_PORT):
7      ignorePastFlow()
8    #Before reading and data is produced
9    if (event=='end-invocation-event' and void_invocation==False):
10     discardInFlow()
11     discardState()
```

4.7 Technical Considerations: Dynamic Types

When we introduced the concept of provenance types, we anticipated that these are wrappers around the components of the workflow. They do not exploit the workflow system's enactment units to produce lineage data, but leverage on the abstractions that are functional to the implementation of the processing elements. This allows to keep the independence from the resource mapping, the communication protocols and the underlying enactment engine. It facilitates the migration across development and production setup where different engine will be used for the same calculation. The example of Figure 4.10 shows, through an UML Class Diagram, how the application of the provenance configuration changes the class hierarchy of a component of the workflow through the dynamic redefinition of the component's class.

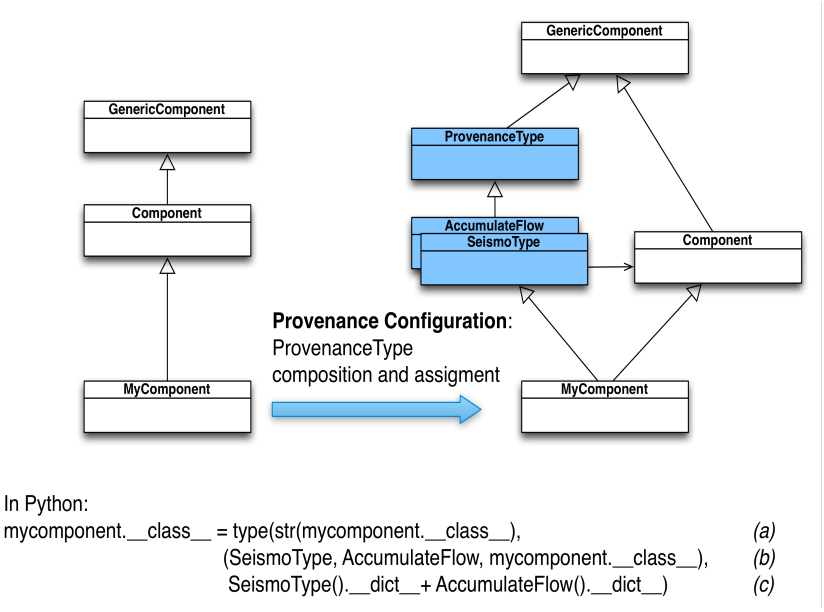


Figure 4.10: Dynamic class redefinition. The diagram shows how the framework realises the composition and attribution of two provenance types (*SeismoType, AccumulateFlow*) for a workflow component, during the application of the provenance configuration. It shows how the reshaping of the original class hierarchy of a component that adds the new types, is implemented through the dynamic form of class definition offered by Python, *i.e.* the `type` function. With this extension an object's class can be (a) renamed and (b) augmented with additional types, which add new methods and modifies the behaviour (c) of the existing ones.

The ability to generate classes programmatically, which is also known as meta-programming, is offered by many programming languages. For instance, in Python, which is the language used for our implementation, this is realised through the built-in function `type()` [55]. In the diagram the processing component is decorated with two provenance types; one deals with the extraction of metadata for seismic time-series, while the other establishes derivations between the output and the sequence of input data (`AccumulateFlow`). Finally, in the implementation for the `dispel4py` system, we apply the dynamic combination of **Contextualisation** and **Pattern** types, leaving the computational logic and the original specification of the workflow unchanged.

Listing 4.6: Log of the realisation of provenance typing in `dispel4py` during the profiling phase. The original types are combined with the provenance types. The `CorrMatrix` is extended by two provenance types that handles grouping (`ASTGrouped`) and another one that can extract stock market metadata (`StockType`). The latter will populate the extended attributes of the S-PROV *DataGranule* entity generated by the component.

```
Assigning Provenance Type to: Source
 Original base class: (<class 'dispel4py.core.GenericPE'>,)
 New type:       (<class 'dispel4py.prov.SingleInvocationFlow'>,
                  <class '__main__.Source'>)

Assigning Provenance Type to: CorrCoef
 Original base class: (<class 'dispel4py.core.GenericPE'>,)
 New type:       (<class 'dispel4py.prov.SingleInvocationFlow'>,
                  <class '__main__.CorrCoef'>)

Assigning Provenance Type to: CorrMatrix
Original base class: (<class 'dispel4py.core.GenericPE'>,)
New type:       (<class 'dispel4py.prov.ASTGrouped'>,
                 <class 'StockType'>,
                 <class '__main__.CorrMatrix'>)

Assigning Provenance Type to: MaxClique
Original base class: (<class 'dispel4py.core.GenericPE'>,)
New type:       (<class 'dispel4py.prov.IntermediateStatefulOut'>,
                 <class '__main__.MaxClique'>)
```

Type attribution is obtained with no impact on the original computational behaviour of `MyComponent`, which preserves the data handling abstractions and methods specified from its original class. In Listing 4.6 we show the output produced by the application of the configuration, where the native types of the components of the CAW are

changed into a combination of these and provenance types. Moreover, this dynamic approach can be applied at runtime for a group of components that have been already mapped onto distributed resources, thus enabling the flexible activation of different provenance recording strategies, depending on precision and metadata requirements. Such integration would support, for instance, short-term validation and monitoring use cases at a controllable overhead.

Finally, this solution plays in favour of more generality, encouraging the re-use of fundamental methods across disciplines, supporting different provenance profiles and exploitation use case. For instance, the CAW application could be used to monitor stock quotes coming from the financial markets, as well as for physical values in real-time. Both use cases adopt the same data structure, such as basic arrays of values, to transfer and process the information. Nonetheless these scenarios may use different **Contextualisation** types to inject into the lineage units of measurements and metadata specific for the domain, increasing the usability of the provenance. Types can be selected from a library and suggested to the user, thereby delivering the portability and the adaptability of the data-intensive application to different contexts and provenance requirements.

4.8 Provenance *Sensors*

In this section we present preliminary work that aims at enhancing the potential of our *Active* provenance framework, to declaratively delegate provenance related operations to dedicated extensions of a workflow. That is, users and developers can choose, as shown in Figure 4.11, to connect groups (or clusters) of processing elements to provenance *sensors*. The approach presented here is to be considered as future work. It is located in this chapter because of its relevance to the overall design of the configuration and exploitation modes, (see Figure 4.1) and its technical integration. Early work [212] reported some of the preliminary concepts and here we suggest an update and a few adoption scenarios. However, the design of a library of useful sensors will require future work to investigate more use cases and how to guide users when making sensible choices for their adoption.

4.8.1 Managing The Transfer of Provenance and Data

Our approach to tuneable provenance brings benefits to data-driven research by offering a framework which also supports customisable runtime monitoring and data-management tasks (Chapter 5 and Chapter 6) that rely on the provenance information produced by the computation. In an *Agile* system, these functionalities need to be guaranteed *by-design*. That is, the combination of the computational and provenance mechanisms have to adapt to the characteristics of the hosting infrastructure and have to be independent from the enactment technology. We have previously introduced how workflow users can selectively instruct the provenance types to store provenance within files, by accessing a local file-system, or to send provenance documents to an external service. The latter can reduce performances due to bandwidth limits and latency. This may depend on several factors, such as network configuration, remote-service responsiveness or a limitation on the number of allowed connections. Allowing each *ComponentInstance* to physically store the provenance can also cause bottlenecks, which may affect the overall performance significantly in near real-time computations. The proposed design allows the inclusion within the workflow of *sensors* that ingest provenance documents and can be tuned to overcome the aforementioned limitations. These can be configured to buffer, queue and directly store provenance messages coming from other processing elements. This approach avoids the need to have third-party software tools installed within extremely controlled infrastructures such as HPC infrastructures.

A `ProvenanceSensor`, in combination with the selective transfer instructions introduced in Section 4.5, can deal with the direct transfer of intermediate results. For instance, if we consider a component that incrementally accumulates information into one or more files, these could be periodically substituted by new files, the old ones closed and transferred to external resources. This can be beneficial for a) avoiding exceeding disk quota on a computational cluster, b) periodically staging the intermediate file to infrastructures where the data can be immediately used for complementary analysis and presented to enable steering decisions. In these circumstances, one or more `DataTransferProvenanceSensors` can independently deal with reading and storing the provenance messages. The transfer should be triggered for the data whose prove-

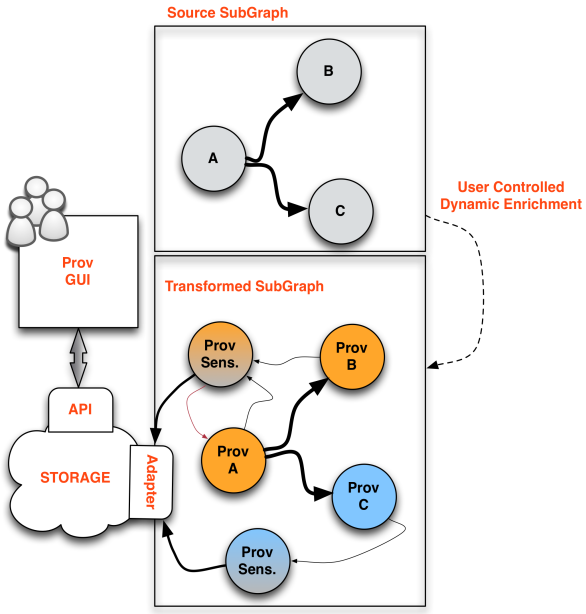


Figure 4.11: Provenance injection and sensors enrichment. A workflow graph is transformed to become provenance-aware. Components are organised into logical clusters and connected to provenance *sensors* for in-workflow rapid analysis and metadata-driven operations. The provenance data is then explored via a GUI, which connects to the provenance store via a web API.

nance entities present information about the current location and that are tagged as *immediate-access* by the transfer rule. Subsequently the sensor removes the file from the file-system, leaving the computational components to their main role of continuing processing tasks over the rapidly incoming data-stream. Configurable and provenance-driven data-movements have been implemented for an HPC seismic simulation workflow, (Chapter 6) showing immediate benefits during training sessions, where trainees could receive rapid feedback from long lasting simulations within an interactive environment.

4.8.2 Clustering and Short-feedback Loops

Another interesting possibility envisaged by this framework, comes with the opportunity of grouping the processing components into *provenance-clusters*. A *provnance-*

cluster can host disjoint groups of components, the smallest cluster is the one associated with a single *Component*. Sensors within clusters can be used to aggregate provenance messages. They could remove redundant information at runtime, and then producing summaries or provide feedback about specific situations before the provenance is stored. More specifically, we foresee that *feedback-loops* would allow the `ProvenanceSensor` to react to specific situations, broadcasting messages back to the *ComponentInstances*. These messages can be used to trigger the re-parametrisation of the components or even the complete replacement of their internal computational method. The S-PROV model captures such runtime changes through the *Change* class and the associated *wasChangedWith* relationship between the *ComponentIntance* an the changeable entities, such as a *ComponentParameters*, *Implementation* and *Location*. For instance, in the CAW, a `ProvenanceSensor` can be associated with a cluster that includes a number of `Source` components and the `MaxClique`. The sensor can then be implemented to re-tune at runtime the sampling-rate of the acquisition of the values to be correlated, to verify whether the length of the sampling intervals has any influence on the occurrence of the variables in the same *clique*. Similarly, a cluster can contain the `CorrCoef` components for a number of variables and the `ProvenanceSensor` can, based on the observation of the resulting *clique*, trigger changes to the length of the batches used for the correlation. In both cases the aim is to find out via the rapid and reactive analysis of a portion of the provenance, the impact of the re-parametrisation on the level of correlation among the observed variables. These are known issues, for instance, in high-frequency stock market data [202, 205, 144].

In our design and preliminary implementation of this experimental feature, that we would like to explore further in future usage studies, the `ProvenanceType` offers the possibility for the component to react and trace a feedback through a dedicated method, `process_feedback`. Ultimately, the implementation and setup of a `ProvenanceSensor` may change depending on the use case and should be included within a library of ready to use *sensors*. This reactive approach to clustering may be compared with others that consider aggregations of processing components as opaque virtual zones of the workflow [187]. This is useful for those workflow authors who do not have access to the code of the computational elements they use. Instead, in-workflow reactive loops could be explored to analyse the behaviour of the computational elements within

a testing and reactive environment, where changes depend on the observation of the outcome produced by a determined portion of the graph at runtime.

4.9 Conclusions

In this chapter we presented an approach that balances between a very practical, focused and science-driven way of setting-up ways of observing a data-intensive experiments via provenance gathering mechanisms. We discussed the means whereby users could investigate, better understand and validate the ways in which those results where produced, while leaving the experiment unperturbed. The comprehensive framework presented takes into account components that work on data-streams and that can be parallelised. Their behaviour can not always be represented by stateless or windowed operations since they use intermediate stateful data and generate outputs that may be characterised by unbounded dependencies. To produce provenance statements for these components that are precise and relevant to the application domain, we introduced a conceptual design and an implementation that is based on composable *Provenance Types* and provenance *Configuration*.

Provenance types are combined to match provenance dependencies patterns and to extract metadata according to standard vocabularies or custom terms. To make this distinction clearer we introduced two classes of types, respectively referred as **Pattern** and **Contextualisation** types. A **Pattern** type captures lineage by manipulating provenance assertions about input data and stateful products. This is realised via the definition of rules associated with events such as the writing of output data and the conclusion of an invocation. A **Contextualisation** type instead populates the lineage with metadata specific for the domain that are extracted from the output data-stream. The chapter presents a library of types and explains how they are implemented.

Users specify the attribution of provenance types to components during the *Configuration* phase, to prepare a workflow for its provenance-aware execution. They can declare semantic clusters and selectivity rules as conditions for provenance production based on domain metadata, or to trigger the transfer of an element to external

resources. The latter is used to to feed complementary processes for storage or visualisation tasks which are typically hosted by external systems. Overall, the approach aims at stimulating a more versatile and *Active* role of the lineage information, providing its tuneable precision and, through improved usability, fostering its exploitation early in the research life-cycle.

A prototypical implementation has been developed for the `dispel4py` computational library as a decoupled provenance module. If needed by the users and workflow developers the module can be imported and offers a framework that allows the creation of new types besides exposing a library of predefined ones. It implements the automatic application of the provenance configuration by means of metaprogramming techniques that extend the workflow components, and therefore their parallel instances, with the properties of the provenance types. To demonstrate the concepts and the technical features of the proposed approach, the chapter introduces a workflow for correlation analysis (CAW), where components can be parallelised and where they have different provenance requirements.

Finally, we have introduced the concept of provenance *sensors* to suggest a way to enable a versatile role of the lineage information in the research life-cycle, where short *feedback-loops* can steer changes into the workflow parametrisation and behaviour of the single components. We want to pursue a sound implementation and evaluation of this design in future work. We foresee the possibility for integration within the sensors of reactive machine-learning tools [222], for in-workflow rapid adaptation.

Allowing for standardisation, custom annotations, tuneable precision and granularity of the provenance in the early phases of the investigation, enables the productive exploitation of the provenance during the early phases of an investigation, towards publishable and repeatable data-products. Figure 4.12 summaries the concept of an extended *Active* provenance framework. From the development and execution of the workflow, to a runtime analysis and feedback-loop, involving workflow developers, users and intelligent systems. As Myers *et al.* [195] describe, this provides immediate benefits to users, such as support for method improvements brought by an evolving documentation of the experimental process characterised by precise traces rich with domain and experimental metadata. These can be accessed from tools that automate

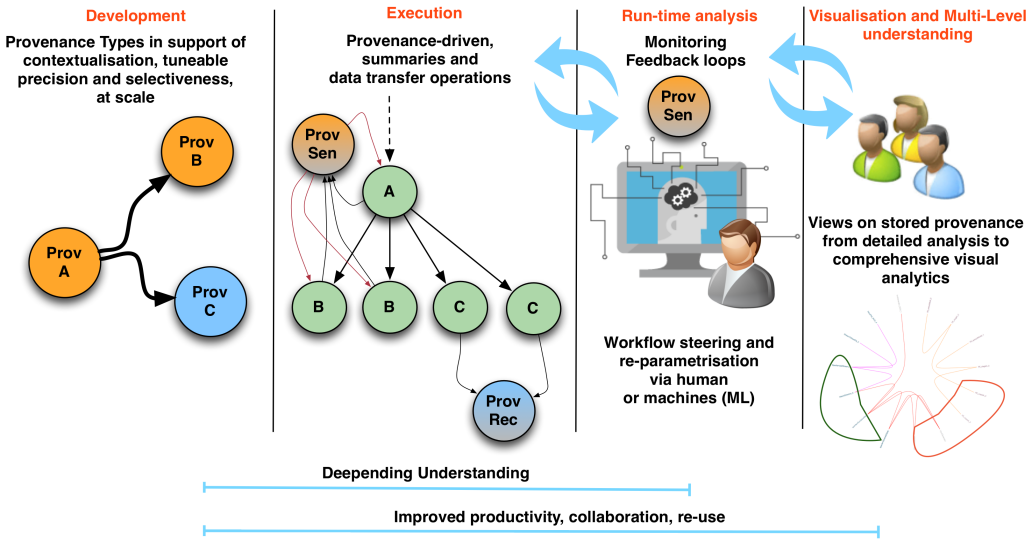


Figure 4.12: Active provenance and positive feedback. From the development and the execution of the workflow, allowing contextualisation, tuning, monitoring and feedback-loops that involve users and intelligent systems early in the process. This enables the reactive analysis of the provenance information, which can then be used for steering the computation. By working with this information many categories of experts gain productivity and understanding.

routine tasks, such as monitoring, result management and validation (see Chapter 5), and thereby showing evidence of a rapid return obtained from the investment in producing provenance data.

We address one of the most important limitations of grids and computational infrastructures identified by Mattmann [182], which is metadata management. For instance, in the case of HPC applications, data-intensive jobs are not closed systems, but rather active entities which can trigger behaviours and state changes in external services while they run. This results in combining tuneable production and runtime access to provenance with selective data movements and monitoring through a unified model and computational framework. This concept is further extended in the next chapter by introducing the `S-ProvFlow` system, an integrated collection of services and graphical tools that assist in the storage and exploitation of the provenance data.

Chapter 5

Access and Visualisation as a Service

We have described in the previous chapter an approach to the specification of the provenance characteristics of a workflow application. This involves different phases. From the contextualisation of the domain metadata associated with the workflow's components, to the possibility of tuning the granularity and the precision of the lineage generated by each component. This is achieved by the definition and application of reusable provenance types, and the possibility of easily applying additional adaptations, thanks to the direct manipulation of the provenance state. Finally, we have shown how to apply these characteristics by the specification of a provenance configuration for a particular run. In this chapter we will cover storage and interactive access and visualisation of the provenance information produced. These help the domain scientist understand the processes, investigate potential flaws and validate critical aspects of their work. Technology specialists may use the same facilities to investigate their concerns. We will address the following thesis contributions:

(C-3) Scale of the provenance records: Provenance information needs to be handled at scale. This concerns the size of the generated information and the flexibility of its content. It affects its acquisition and its manipulation via interactive tools. We present a representation of the provenance information of the S-PROV model and arrange its storage for data-intensive provenance scenarios.

(C-4) Support for multiple levels of understanding: Provenance and data-lineage information offer insights that interest different roles, from the domain-scientists to the community manager. In data-intensive computations this information can be overwhelming, hiding latent but significant evidence of a method's effectiveness and efficiency towards meeting required goals. This chapter investigates ways of interactively accessing and visualising the provenance, offering detailed interactive navigation of single executions, as well as tuneable perspectives involving data, people and infrastructures across multiple workflow runs.

(C-5) Integration with tooling: During the course of the PhD a comprehensive system including interactive tools, services and a database have been designed and implemented, offering access and visualisation of the provenance as as a service. The system, `S-ProvFlow` introduces technical solutions for provenance persistence and approaches to visualisation that rely on a high-level and combinable query interfaces and interactive tools.

5.1 Motivation

As introduced in Chapter 2, solutions for querying and visualising provenance traces have been investigated in the literature [158, 121, 106]. Some address particular scenarios or specific workflow systems, some others offer general functionalities, such as *ProvStore* [160]. However, to realise many of the use cases illustrated in this and the next chapter, the existing solutions entailed substantial effort to be extended and integrated in our context to reach the desired goals. We adopted general purpose toolkits when needed, especially for the manipulation and automatic serialisation of PROV documents [3].

Thus, we designed the features we wanted to focus on, and developed a system that exploits the potential of the S-PROV model and the *Active* framework. It highlights aspects of distribution and delegation of the processing components, shows their stateful behaviour and deals with volatile and concrete data products. These are all described by a variety of discoverable and actionable metadata terms and parameters. The latter

gained the users' attention, by delivering data discoverability in an exploratory space and visual analytics relevant to each user's interests. This is packaged as two provenance exploration tools built on top of a provenance store and a web API.

As a short anticipation, the system's features are used in two scientific contexts at different stages of their maturity in their design of methods and standardisation of working practises. The computational seismology community exploiting a novel simulation platform, was the first challenging test case. Agile collaboration in requirements capture, design and prototyping led to a highly valued and sustained research environment that met their goals and supported their long-running investigations. The next challenging test case had to support the work of climate scientists as they computed and presented the impact of climate change in specific geographic areas. The resulting integrated set of components continue to be developed and enhanced. In this chapter we report the latest progress, while new research projects will deliver further advances and larger scale applications in these domains.

5.2 Reproducibility Challenges

The reproducibility of scientific findings is essential to improve the quality and the dissemination of modern data-driven research, especially with the ever growing variety of data and the advances in computational software libraries. The principal investigator or a third party should be able to access the same materials and methods to confirm a prior result [100]. It requires that the final and the intermediate results can be verified. In many disciplines this is proving difficult to achieve, raising the question as to whether modern research is heading towards a serious reproducibility crisis [99, 192, 120, 113] which would undermine trust in the implications of that research. In computational research, effectiveness of the reproducibility practices are affected by external and imposed changes, circumstances and limitations.

Changes are attributed to data, its location, access rules, format and updates, and to all the other digital dependencies that characterise the methods implemented through a workflow tool. A similar challenge arise from the changes in the context, such as, soft-

ware library updates, new communication protocols, revised system software and hardware changes. Moreover, we need to consider all the changes introduced by humans which may occur offline or at runtime. For instance, in near real-time applications the observation of specific properties associated with the progress of the computation may trigger parametrisation revisions.

Finally, in large computations the drive for rapid results on low-cost production systems often overrides the time and resources needed to achieve reproducible research. The necessary information is often not available or those responsible for the services and systems do not produce it in a usable form. Practitioners moreover, ignore the need for methods by which the validation of an experiment can be made more reliable and certain.

5.3 Supporting the Computational Research Cycles

The impact of the reproducibility challenges can be mitigated rather than avoided. Mitigation can be achieved by putting more emphasis on the provision of technologies, methods and working practices to gather and exploit the necessary information in support of validation and traceability, throughout all stages of a data-driven method development and use. This is in line with the principles defining a research-object [100], where research artifacts should be traceable, besides being repeatable.

Hence, as discussed in the previous chapter, it is important to offer to users tuneable provenance-aware computational libraries that can help developers and users in the implementation and execution of experiments that can be automatically traced at different levels of detail. Details should be captured within an holistic formal model, as shown in Chapter 3 with S-PROV, that offers different layers of granularity. The representation of the model needs to be effectively combined with services offering provenance acquisition and exploration, anticipating the adoption of reproducible frameworks and practices early in the research life cycle. Increasing the participation of users will contribute to the better quality of the provenance. This depends on demonstrating the benefits of its rapid availability [118, 97].

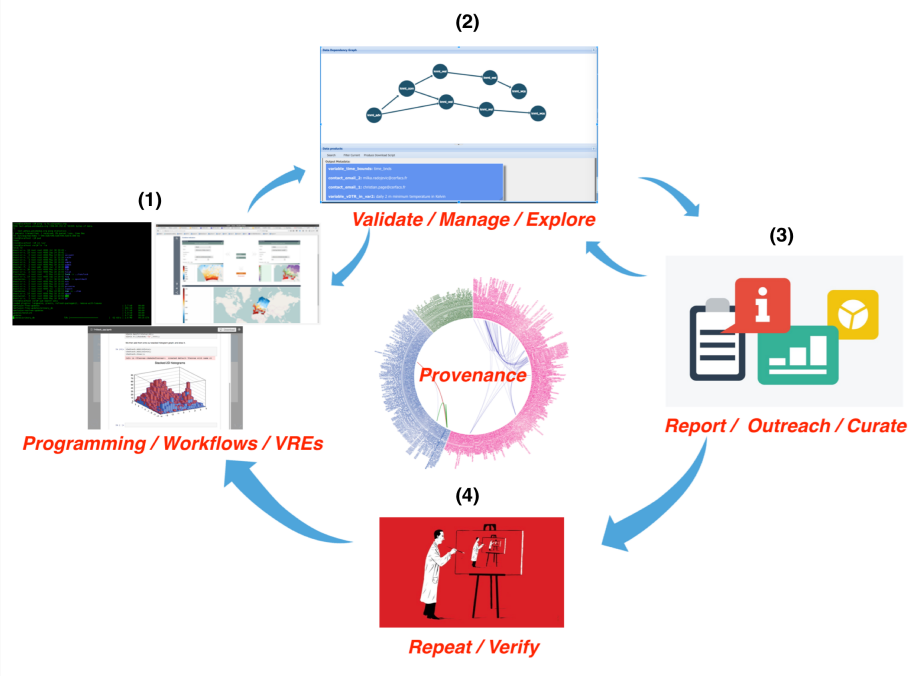


Figure 5.1: The Computational Research Cycles. (1) The design, testing and development of experiments. (2) The analysis of preliminary results to evaluate the methods. (3) The sustained use of the method in production to obtain sufficient result with sufficient quality leading to publication of results. (4) The use of data-lineage records combined with mechanisms that enable the re-enactment of the experiment to revisit process and test the validity of the evidence they produce.

In Figure 5.1 we depict how in computational sciences several intermediate cycles are required before obtaining a final product that can be communicated and the associated experiment repeated. We report in this chapter a set of services on top of the holistic provenance representation to bridge between these phases, with tools that facilitate the understanding of the computational processes. Making provenance information representable by the combination of different, and preferably standard, ontologies and metadata schema [102] is fundamental to support interoperability across contexts, assuring a shared understanding of the relationships among different players accessing and producing a scientific result. For this reason, the proposed system allows selected subsets of provenance to be exported as a semantic-web representation for ingestion by third party services and semantic reasoning. This makes possible the use of tools and quality assurance methods that have been developed for those standards.

The exploration tools presented in this chapter are designed for a range of experts. Once the data-lineage has been produced, at the desired level of detail, we enable the understanding of the "live" processes, fostering computational steering, sharing and reuse of data and methods. We will demonstrated the value of provenance exploitation to monitor experiment progress and to review relationships between large numbers of runs.

Overall, we aim at improving the synergy between different categories of expert. These include the domain-scientist using the analysis tool, the research-developer and data-scientist creating and refining the software and the methods, and the operation teams managing the resources. For instance, system engineers who deliver services for applications that handle a growing scale of data and run on distributed e-Infrastructures, can use these tools to gain insights into systems interactions and data transport in the context of the data-streaming methods their user communities are using.

The `S-ProvFlow`[1] package presented in this chapter combines a set of components that support provenance acquisition and exploration. It includes a database, a webservice layer and two complementary interactive tools. The whole system is delivered as a composition of different *Docker* [18] containers for an "easy" and decoupled installation of its components. This facilitates ways to make it accessible to research-developers and administrators that wants to explore or keep the the system updated. The realisation of the containers has been implemented with the collaboration of Andrej Mihajlovski and André Gemünd from the KNMI and Fraunhofer SCAI institutions.

One of the tools, the *Monitoring and Validation Visualiser* (MVV), assists the users in the fine-grain interpretation of the provenance records in order to understand dependencies; it allows them to select and configure viewpoints by specifiable searches over domain metadata values-ranges, previews, navigation of data dependency graph, within and across runs, data download and staging. It offers detailed runtime diagnostics also differentiating between stateless and stateful operations. A graphical tool, the *Bulk Dependency Visualiser* (BDV) offers broader perspectives on computational

[1] `https://github.com/aspinuso/s-provenance`

characteristics as well as collaborative behaviour via customisable radial diagrams. It adopts hierarchical edge bundle techniques and configurable grouping. It allows its users to dynamically adjust viewing and clustering controls to uncover aspects of the distribution of the processing for large single runs and data-reuse between different worklfows' executions and users.

5.4 System Design and User Engagement

The work pursued on the user interface, especially concerning the MVV tool (Monitoring and Validation Visualiser, see Section 5.6), required several iterations and the observation of people's reaction to the interface. Initially, and especially for the more experienced researchers, it was very interesting to notice that it was challenging to propose an innovative way to access their results, which did not require them to memorise file-system structures and file-name patterns. It seemed like an important role is played by the psychological perception, probably driven by habit, of loss of control when doing otherwise. The introduction and improvements of visual aids to navigate the data dependencies, editing of runs' annotations and search capabilities, especially facilitating transfer operations of specific subsets of data, helped in gaining more trust and positive feedback. Still we experienced the need for frequent communication adopting terminology closer to the domain expert and organised in short-cycled development and evaluation steps, This is crucial to smooth the intellectual ramp. However, we could also observe a rapid gain of confidence in the seismology students joining the training sessions. After the first explanations by the trainers, we noticed that they were approaching the interactive features offered by the tools more naturally. Especially features like runtime monitoring, dependency navigation and rapid access to intermediate results where fundamental to keep the lesson more dynamic during the submission of the simulation.

The BDV (Bulk Dependencies Visualiser, see Section 5.7) was appreciated by senior researchers involved in teaching activities (Personal communication with Prof. Heiner Igel, from the University of Munich), especially when showing the dynamics occurring within their computational methods. Managers of research infrastructures, were

impressed to see in a comprehensive view that their clusters were interacting with so many different workflows, users and similar external infrastructures offered by other project partners (Personal communication with Anton Frank, Research Coordinator at LRZ [71]). However, we should not underestimate privacy issues associated with the public disclosure of such information in a real-life service.

Thereby, we pursued our requirements elicitation process iteratively, in collaboration with a core set of experienced users. These were constantly involved in design choices, with evolving refinements and that were regularly exposed to update and prototypical versions. Among the functional requirements, we had to guarantee that users were able to manage, search and reuse the results of a large number of experiments. As the lineage of a streaming computational model was our underlying setting, we had to handle the difference between volatile and concrete entities, in order to provide a exhaustive validation of the run, while offering rapid access to its concrete output results often stored as multiple files and online resources. Moreover, users that were exposed to long-running computations, especially within HPC infrastructures, required to monitor the execution in detail and in the context of their application domain. Therefore, our design had to support runtime feedback with substantial scientific and technical information, differentiating between the variety of provenance elements offering different interactive modes. This required us to experiment and explore solutions, which resulted in tools and methods that in this context were demonstrably preferable to the existing approaches. Non-functional requirements instead were dictated by the need to overcome limitations in setting up provenance acquisition and management services embedded within the computation infrastructure, and by the need for a flexible and performant system that can be easily tuned to support evolving scenarios. These concern the support of adaptable indexing strategies, ease of formats manipulation and a design that decouples tooling, method abstraction and the implementation of the underlying model.

We have presented our approach to user engagement and the overview of the most relevant requirements. In the following section we proceed by discussing the details of the design and technological choices for the storage and access of data-intensive lineage.

5.5 Storage and Access

We present in this section our approach to the storage, access and visualisation of the provenance information. The data is captured according to the S-PROV vocabulary and concepts introduced in Chapter 3. By investigating the use of a document-oriented database based on JSON representation, we explored ways to benefit from the simplicity of implementation and the flexibility of integration for which such technologies are gaining momentum. On the practical side, as many systems that exchange metadata over a network, the JSON format has advantages due to the wide availability of software libraries delivering efficient manipulation and visualisation. It facilitates the adoption of the same format across all layers of a system reducing the necessity for mappings and complex format conversions. Finally, thanks to a shared representation of the information, it allows the rapid implementation and validation of new features, fostering the evolution of the software involved into the different components.

Despite document stores are not commonly associated with semantic-web scenarios, the success gained by the format and the supporting technologies motivated the semantic-web community to deliver a new standard, JSON-LD, to support the inclusion of *Linked Data* information. The capability of representing RDF triples facilitates the interoperability among different classes of systems that can be interchanged depending on scale, access mode and type of analysis. Suggesting future research towards hybrid storage architectures for provenance management.

Hence, for the representation and storage of the provenance we adopt a JSON format and a NoSQL document-oriented database, implemented using MongoDB [35]. On top of the repository, we had developed a service layer based on a web API (Section 5.5.2), which offers the possibility to query and update the provenance using a set of high-level methods designed to support different exploitation scenarios and interactive visualisations [213]. These are iteratively refined according to new emerging requirements and use cases, gathered in dedicated validation sessions with expert domain scientists, and by observing users behaviour during official trainings. Figure 5.2 shows an overview of the system and its interaction with computational, data and provenance resources. We proceed by explaining the characteristics and rationale of the system

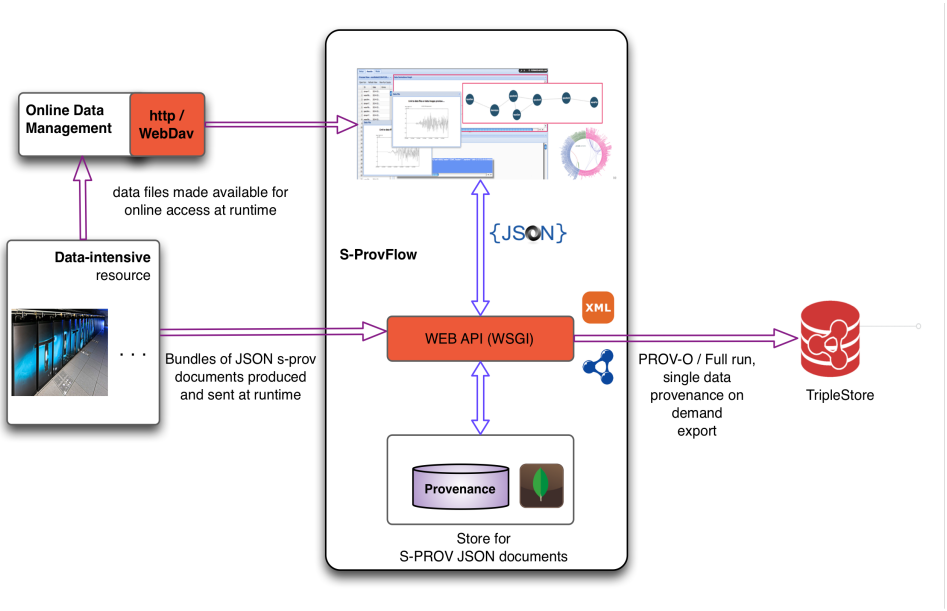
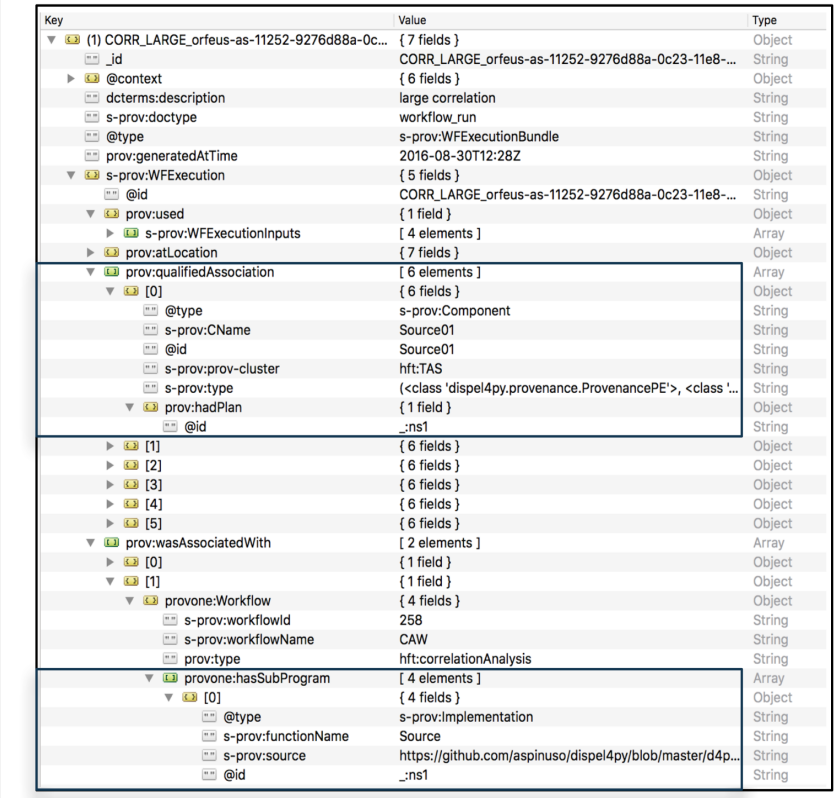


Figure 5.2: Schematic architecture exploiting the `S-ProvFlow` system for acquisition, visualisation, data access and provenance export services.

and its functionalities.

5.5.1 Representation of the Trace

The choice of a document based approach is motivated by the nature of the provenance information produced by our target class of systems. The provenance relationships occurring when a bulk of data items are written to an output port or saved within the provenance state of a component are easily representable within a self-contained document. This is characterised by an internal structure that links the output data with its derivations, metadata and relationships with the workflow's processes and abstractions. We encode the S-PROV model as two JSON documents describing respectively the setup of a workflow run and the associated lineage. In Chapter 3, Figure 3.8, we have already introduced a representation of a S-PROV lineage document in JSON-LD. In Figure 5.3, we present instead the document describing the run. We call this bundle document, as analogy to the PROV bundle. In literature a general serialisation of PROV to JSON-LD [159] is presented, making the data easily exportable to serve semantic-web class of services. However, the current implementation in our target



Key	Value	Type
(1) CORR_LARGE_orfeus-as-11252-9276d88a-0c...	{ 7 fields }	Object
_id	CORR_LARGE_orfeus-as-11252-9276d88a-0c23-11e8-...	String
@context	{ 6 fields }	Object
dcterms:description	large correlation	String
s-prov:doctype	workflow_run	String
@type	s-prov:WFExecutionBundle	String
prov:generatedAtTime	2016-08-30T12:28Z	String
s-prov:WFExecution	{ 5 fields }	Object
@id	CORR_LARGE_orfeus-as-11252-9276d88a-0c23-11e8-...	String
prov:used	{ 1 field }	Object
s-prov:WFExecutionInputs	[4 elements]	Array
prov:atLocation	{ 7 fields }	Object
prov:qualifiedAssociation	[6 elements]	Array
[0]	{ 6 fields }	Object
@type	s-prov:Component	String
s-prov:CName	Source01	String
@id	Source01	String
s-prov:prov-cluster	hft:TAS	String
s-prov:type	(<class 'dispel4py.provenance.ProvenancePE'>, <class '...	String
prov:hadPlan	{ 1 field }	Object
@id	_:ns1	String
[1]	{ 6 fields }	Object
[2]	{ 6 fields }	Object
[3]	{ 6 fields }	Object
[4]	{ 6 fields }	Object
[5]	{ 6 fields }	Object
prov:wasAssociatedWith	[2 elements]	Array
[0]	{ 1 field }	Object
[1]	{ 1 field }	Object
provone:Workflow	{ 4 fields }	Object
s-prov:workflowId	258	String
s-prov:workflowName	CAW	String
prov:type	hft:correlationAnalysis	String
provone:hasSubProgram	[4 elements]	Array
[0]	{ 4 fields }	Object
@type	s-prov:Implementation	String
s-prov:functionName	Source	String
s-prov:source	https://github.com/aspinuso/dispel4py/blob/master/d4p...	String
@id	_:ns1	String

Figure 5.3: S-PROV properties describing a *WFExecution* expressed in JSON-LD. The squares highlight the association between the workflow *Component* (`Source01`) and its *Implementation*.

document store adopts a more concise format, facilitating the ingestion and runtime access of the information produced by the *Active* framework. The support for more general representations is envisaged.

In Figure 5.4 we present a schematic view to explain how the provenance dependencies are associated with internal document's data structures and how they are connected within and between the documents. Each lineage document presents a list with the *Data* and *DataGranule* entities produced by a write operation (or state update) and a list with their derivations. The derivations are used to link to the data dependencies, which are described in other documents. Occurrence of changes to an instance can also be described in a lineage document, allowing to trace the association with the first

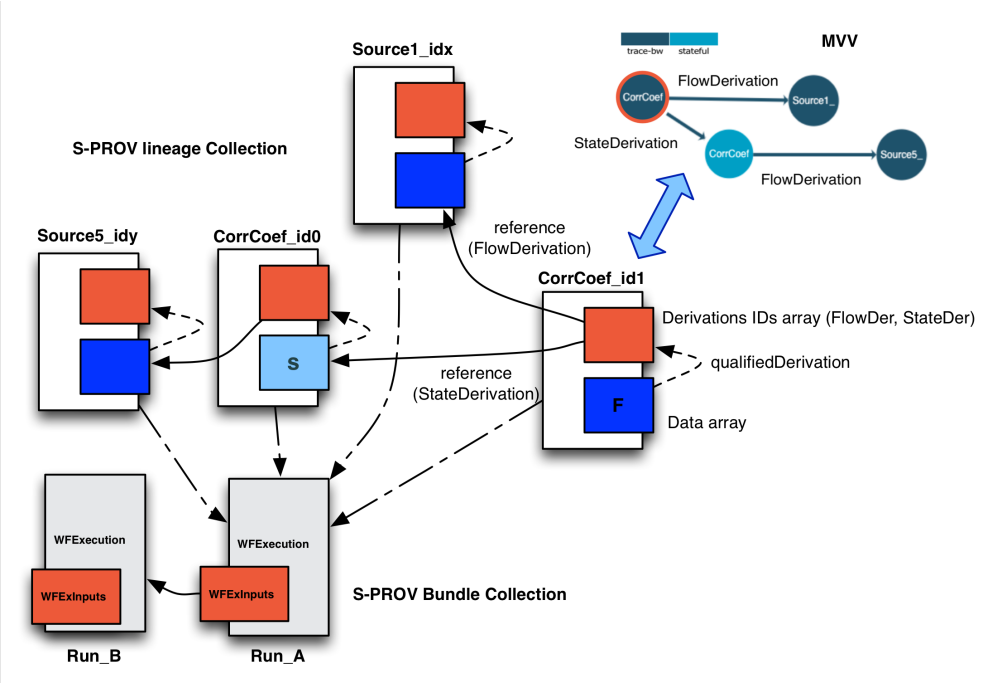


Figure 5.4: Document store, lineage and bundle collections: Lineage documents' dark-blue and light-blue (stateful) boxes represent the list of *Data* elements produced as output; red-boxes are the *qualifiedDerivations* that contributed to the output. The latter references the provenance assertions described within other lineage documents. Each document contains additional information about the invocation generating the data. They all refer to the bundle document of their run, which may link to other bundles to indicate data-reuse across executions

invocation that acknowledged the change.

Moreover, by adopting such schema for the lineage documents, we take advantage of denormalisation. This design choice allows for redundant copies of targeted information and it is recommended by document stores in order to use them efficiently and gain performance. This approach is beneficial especially when the data is rarely updated but frequently read, favouring scalability in terms of both size and representation. Hence, because of the append-only update pattern adopted in our implementation and the limited redundant information associated with the events occurring at runtime, denormalisation can be afforded. Repeated data are mostly related to some of the *ComponentInstance*'s properties and relationships, such as its parametrisation and location, the workflow's abstract *Component* and the implementing function.

We instrument these features within a NoSQL technology, MongoDB. It relies exclusively on the JSON format that is commonly used to reduce the complexity of the mappings of the representation of the information from the application logic, (from the producer side as well as from the consumer) to its storage. The documents stored into the system include further adaptations that exploit the possibilities offered by database technology to create custom indexes on the document's properties, to benefit the performance of various queries, especially those performed on domain metadata and parameters. The importance of the management of domain and experiment specific terms and values as part of the provenance is considered extremely relevant in literature [124, 139]. This is the case for the properties describing the *DataGranule* and *ComponentParameters*. Though, we have to take into account that in large collections, when there is no fixed vocabulary associated with a dictionary, the choice of which index to generate in order to rapidly perform searches on these terms presents several challenges, which also include the maximum number of indexes supported by the underlying technology. In Section 5.5.2 we will briefly describe the adaptations and procedures that enabled to address this and other exploitation use cases based on domain and user-defined metadata.

For the sake of portability the `S-ProvFlow` system can export complete workflow's execution provenance traces, and the lineage of a single *Data* entity in PROV compliant formats (PROV-XML and RDF-Turtle), facilitating interoperability with general purposes provenance archives, such as *ProvStore* [53]. This will foster compatibility with other provenance exploration tools and use cases that require semantic reasoning. Moreover, it guarantees the link to the fundamental ontologies (PROV-O, ProvONE, S-PROV) and to those imported by the users, for instance, through the *Active* framework, by applying a specific provenance configuration. The export formats will be useful for the publication of final research products to institutional and multidisciplinary data archives, where provenance may be included in their data-curation processes, in support of reproducibility and validation by peers. Information describing the details of users, affiliation and infrastructures is beyond the scope of this work, assuming our system to be complementary to third parties repositories that take care of more organisational information, for instance, such as the DCAT-AP [138]. Such catalogues can link or embed provenance information according to the format and policies adopted by

the specific profile.

As a concluding note, we run several instance of the database for tests and in production deployments, as we will show in the Chapter 6, managing millions of documents. We have served a number of trainings in seismology [81], with more users performing simulations and storing provenance documents concurrently. Large production deployments of the database technology in other contexts exist. The manufacturers report that performance is be mostly affected by the combination of indexing and update strategies and document schemas [46].

5.5.2 Access Methods

In order to facilitate the realisation of interactive tools that exploit provenance data, the `S-ProvFlow` system exposes a collection of methods through a web API, which is built on top of the storage backend. The methods were initially designed to meet the requirements collected and discussed during the realisation of a real computational service for seismology [97]. In that application, the provenance information had to address use cases such as runtime monitoring, detailed dependency exploration, visual summarisation and integrated data-discovery. The latter is used in support of different workspaces to perform automatic data selection and integration, see Chapter 6. The description of the API's methods is provided in Table B.1 of the Appendix B and the source code is available as part of the `S-ProvFlow` master repository[2] in GitHub.

We proceed with explaining the rationale of some of the use cases and high-level methods of the API, referring in some specific cases to their implementation to show how we exploit the database query interface. The specific examples that will be mentioned are expressed in pseudocode, and reported in the Appendix B. As a short background technical information, MongoDB offers a collection of query functions, that are extensively described in the official manual [75], of which the most relevant for our implementation are `db.collection.find()` and `db.collection.aggregate()`. They select and project documents in a collection and calculate aggregates based on the documents fields and values. Moreover, we will introduce in Section 5.6 a postpro-

[2] `https://github.com/aspinuso/s-provenance`

cessing task that exploits the map-reduce functionalities offered by MongoDB for the realisation of a particular use case.

Layered Activity: the execution of a workflow can be monitored at different level of details. From the high-level classification of the components introduced by the users, such as semantic clusters, to the invocation of one of the instances. The listings B.1 and B.2 show respectively how we use the aggregation framework offered by MongoDB to produce multilayered views on a run and the resulting response. The denormalised approach to the storage of the lineage documents allows us to perform queries that obtain automatically complete processing and attribution information. Details such as location, amount of data produced, execution time and software agents are immediately collected and aggregated without joins to follow references. The method of the API offering such functionality is number (5) of Table B.1. We will make use of this numbering schema to reference the API's methods throughout the chapter.

Search: the validation and traceability query methods of the API, referred as (4), (9), (10), (15) in Table B.1 perform searches on concept and terms defined by the S-PROV model and vocabulary, and on the terms associated with the properties of the *DataGranules* and *ComponentParameters*. These are used in combination with their values-ranges to search for data and workflows' executions.

To serve these class of methods the current implementation extends each *Data* object of a lineage document with a list [`indexedMeta`] containing dictionaries of the form {`key:<term>, value:<val>`} and adopts a compound index [76] on the two properties. This approach allows us to perform fast searches on terms which are dynamically added to the provenance collection. Otherwise, indexes would have to be generated ad-hoc depending on predefined vocabularies, which besides being a less flexible approach, it may reach the limit of the maximum number of indexes allowed by the database. Terms to be included in the indexed list are selected based on specific characteristics. For instance, we do not include properties with very large textual descriptions, since it is very unlikely to have values-ranges queries on these sort of strings. However, use cases that require full text searching could be obtained by combining the document store with

other technologies, based on ElasticSearch [36], which better serve this sort of functionalities. A sample query that extracts a document based on the values of the metadata occurring in the *DataGranule* is reported in the Listing B.5. We omit for simplicity the complete implementation of the method to focus on the actual query that exploits the representation of the provenance.

Metadata indexing and other aspects (that will be explicitly mentioned in this chapter) have been addressed in collaboration with the Bachelor of Science project of Thomas Kok, who investigated the possibilities offered by MongoDB to apply different indexing strategies and procedures, aiming at serving the methods of the API efficiently.

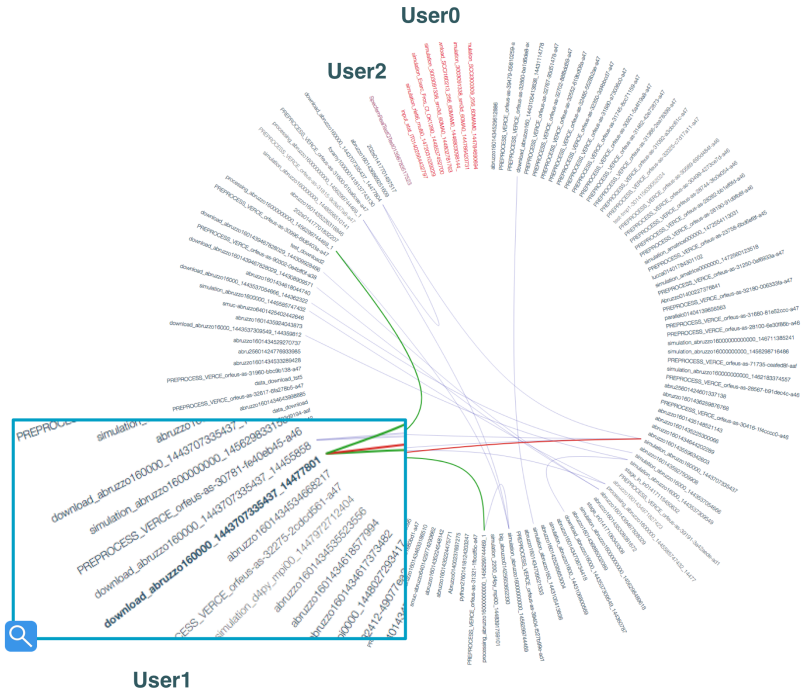


Figure 5.5: Radial visualisation involving three users (colour coded sections of the graph) and their runs (vertices) producing and reusing data (green, red edges) that involve `stations` codes CERA and CAFR. The diagram relates to time-series analysis workflows for seismological data

Data Lineage: the methods of the API `data.derivedData`, `data.wasDerivedFrom`, referred as (11) (12) in Table B.1, allow users to navigate the data derivation graph interactively by specifying how much depth should be visualised at each

step. This is used to build a visual and interactive representation of the trace, as we will describe when we will present the MVV tool. For instance, a request to the `data.wasDerivedFrom` will be served by the database query and algorithm, whose pseudocode is provided in Listing B.3, and returns a document of the form shown in Listing B.4. Also in this case, the denormalised approach to the storage allows us to easily obtain information about the processes and agents involved into the generation of the data. Another method that combines graph traversals at a configurable depth with queries on metadata values-ranges is the `data.filterByAncestor` (10). It receives a list of data `ids` and applies a filter excluding those whose ancestors' properties do not match the query parameters.

Aggregations: the API provides two high-level summary methods to extract comprehensive information. One of the methods (14) covers processing dynamics, such as data transfer between the components and their concrete instances, indicating additional details, such as the computational nodes and execution modes, depending on the chosen enactment. Another method instead (15) reveals collaborative dynamics, such as data-reuse between people, workflows and infrastructures. These are built interactively by specifying properties of the data produced by the users' runs.

Figure 5.5 shows an example of a comprehensive summary produced from a collection of seismology workflows. The view is grouped by `User`, thus putting in the forefront collaborations between researchers who exchange data that present specific characteristics. The implementation of this type of methods requires to access the database by executing queries that aggregate and group documents across the two collections. The queries that contribute to the generation of Figure 5.5 is shown in Listing B.6.

We will present in the next sections the two visual tools built on top of the `S-ProvFlow` API, for the interactive exploitation of the provenance traces.

5.6 Visualisation: Monitoring and Validation Visualiser

The `S-ProvFlow` system offers a visual tool (Monitoring and Validation Visualiser - MVV), that enables different sorts of operations through the interactive access and manipulation of the provenance information. These include monitoring of the progress of the execution, discovery of data and runs, filtering, data preview, download and staging. Below we cover each of these aspects separately. The visual components of the tool introduced in each of the following sections are shown in Figure 5.6.

5.6.1 Monitoring

The *Runtime Monitor*, displays the activity of the workflow after it has been mapped to the target resource. The view can be updated at runtime and its content is dynamically fetched from the API according to the user's position within the scrollable area. This allows users to easily browse through the items. Although the interface considers the components' instances as the default level of monitoring, other levels can easily be supported thanks to the layered activity service offered by the API (5).

The list shows the timestamp of the most recent invocation, the count of the data produced, occurrence of runtime messages, such as errors, warnings or special textual annotations coming from the live computation. Runtime changes affecting the instances are highlighted reporting their total count. Clicking on one of the instances loads the provenance information of the generated data in the *Data Products and Metadata* panel, for visualisation and further operations, as we will describe in the following paragraphs.

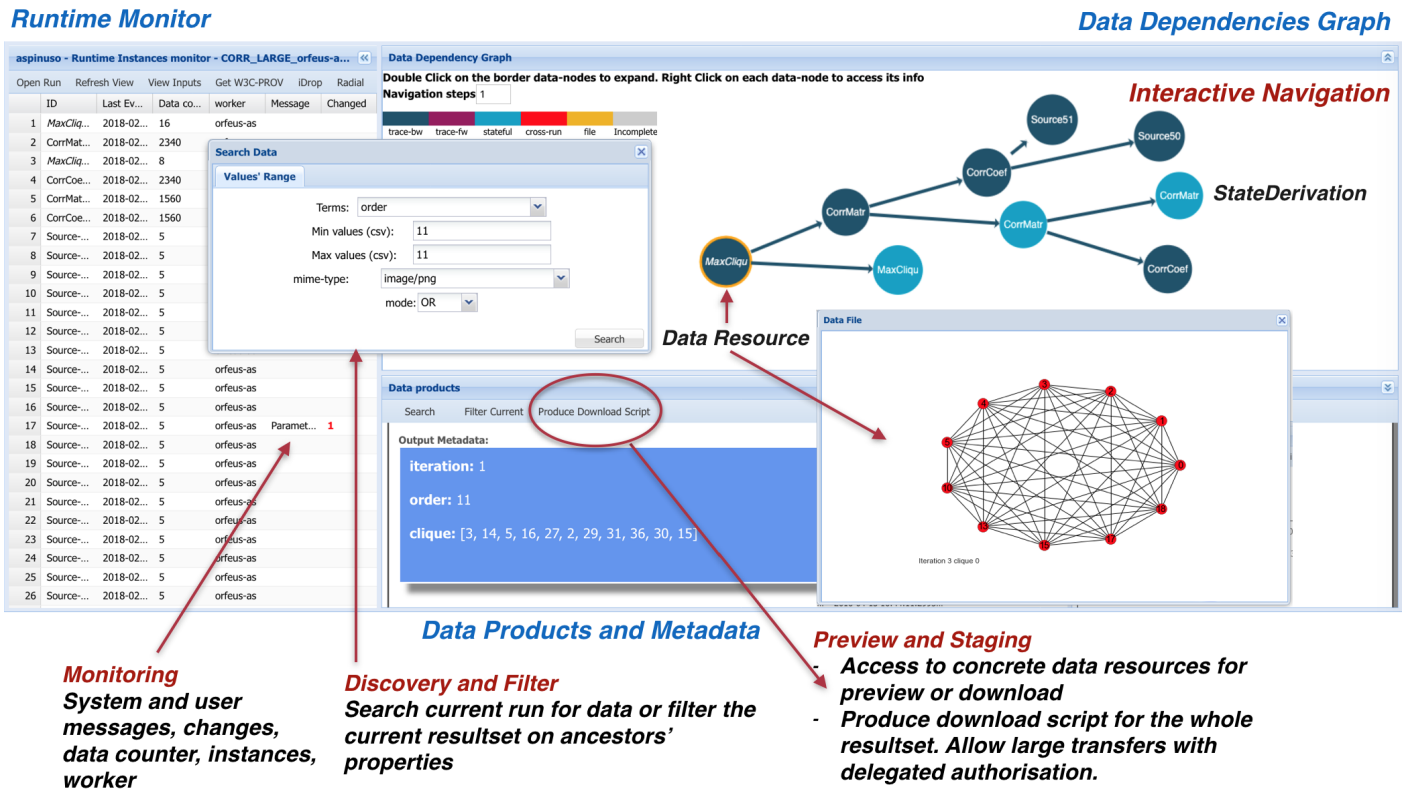


Figure 5.6: MVV: Navigable graphical representation and combined access to data products and metadata. This example represents the interaction with the provenance information of a run of the CAW application. The *Runtime Monitor* shows the list of active instances. It reports the quantity of data produced, feedback messages and the changes occurred, when this applies. In the *Data Dependencies* graph the yellow circle indicates that the provenance entity links to a concrete data resource, represented by an image of a large clique in this specific example. Metadata are visible in the *Data Product and Metadata* panel. The interface also allows the selective export of the lineage to PROV formats

The list is also capable of notify the availability of data-products for immediate download, visualisation or staging, through a *bold-Italic* style in the label of the instance. The visual aid complements what have been described in Section 4.5, to support the functionality of the *Active* provenance framework that triggers selective transfers of intermediate results, for instance, to location that are readily accessible to the user. As Myers *et al*. [195] have observed, the early engagement of the scientists is a crucial step in gaining appreciation of the value of provenance collection and preservation, which should incrementally lead to improvement in the trustworthiness of results and the quality of the science.

5.6.2 Discovery of Experiments and Data

Users can search for workflow executions and data elements adopting terms which refer to standard vocabularies, as well as experimental terms introduced by specific application's and researchers' requirements. Searching for workflows' executions allows them to configure the MVV tool for the exploration of a specific workflow run, see Figure 5.8.

Once the run is selected and the MVV prepared to access its provenance, users can search for data in the *Data Products and Metadata* panel or apply filters on the data already listed (*i.e.* on the properties of their ancestors (10)). Here, each data product is described by its metadata and the information about its generating process. The selection of the terms for the searching is assisted through hints. These are suggested among the terms introduced by the user's runs attributed to the S-PROV entities such as *ComponentParameters* and *DataGranule*. The hints are presented to the user by accessing an additional database collection, the `terms_summaries`, that is regularly updated via the incremental analysis of the whole provenance archive. The update is performed offline by a batch job that, by executing two map-reduce processes on the lineage documents and the `terms_summaries` itself, emits and updates statistics for all thyellowe terms introduced by the users' experiments.

The workflow implemented by the batch job is described in Figure 5.7. This new collection is queried by the `terms` method of the API (13), which returns use (metadata

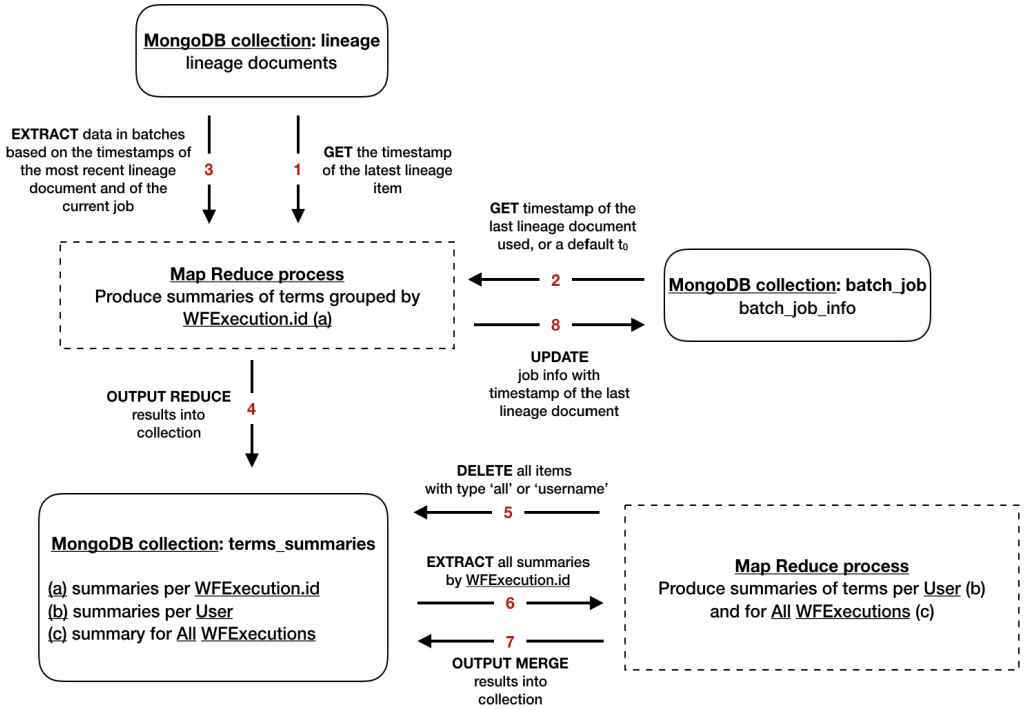


Figure 5.7: The image shows the workflow that produces the summaries about the terms that are introduced by the users' run in the provenance archive. The summaries describe the use of the term (metadata or parameters), their type (string or numerical) and statistics (count, max and min values). Three summaries are produced: for single runs (a), users (b) and for the full collection of workflows' executions (c). These are exposed by the `terms` method of the API. Image kindly provided by Tomas Kok.

or parameter), type, *min* and *max* values, when they have a numerical type, and their number of occurrences within the scope of the search. They may be associated with namespaces prefixes referring to controlled vocabularies or new and experimental. As for the metadata indexing strategy, also this work has been conducted in collaboration with the Bachelor of Science project of Thomas Kok.

5.6.3 Data Dependencies Navigation

The *Data* elements returned by a search or a filter are examined within the *Data Products and Metadata* panel. Each data element can be analysed in detail, including the possibility to start the interactive exploration of the data dependencies, which is then

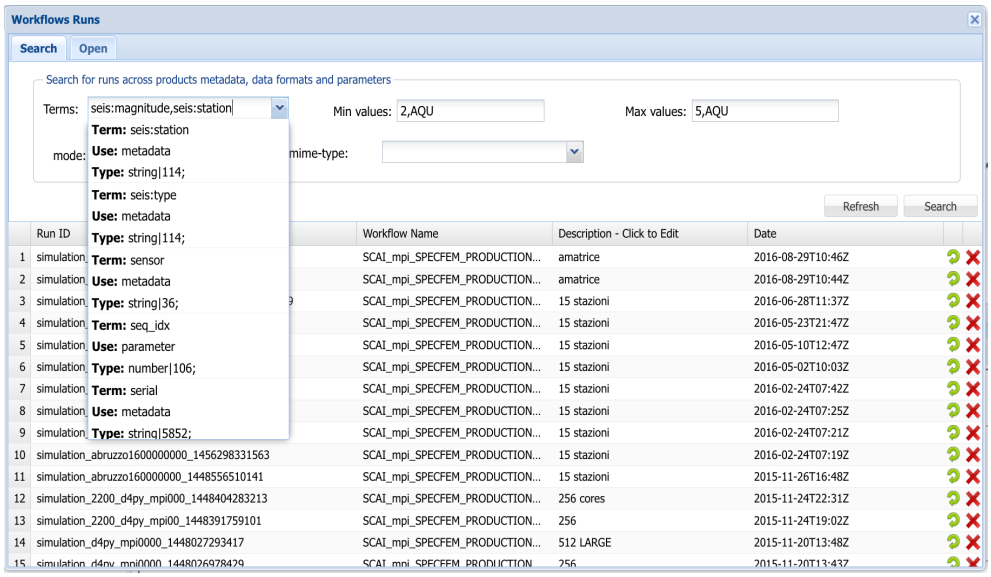


Figure 5.8: MVV: Workflow Execution search panel. This panel is used to discover runs of interest based on parameters and metadata values-ranges. The list present the user's runs that match the searching parameters. The suggested terms are the ones introduced by the user's workflow executions. As shown in the dropdown list, these can be either qualified (`seis:station`, `seis:magnitude`) or experimental (`seq_idx`). The terms are further described by their *Use* (parameters or metadata) and their *Type*. The total count of their occurrence is also reported. A run can be then selected for its interactive evaluation. Additional search items may be included, (*e.g.* component's implementation, workflow type, data-format).

performed within the *Data Dependency Graph* panel. The navigation is controlled by clicking on each data node and users can configure the depth of each step. The reason for allowing tuneable exploration depth is to support localised and detailed evaluation of the data dependencies within an otherwise large trace. We reduced the verbosity in the graph representation by excluding the explicit rendition of the processing nodes, to focus on incremental data traceability with comprehension of relevant process details.

By iteratively formulating and discussing the functional requirements with domain scientists, we envisaged a scenario where users start searching for data with certain properties and subsequently explore interactively its lineage. The dependency graph can be expanded bi-directionally with the methods (11) and (12) of the API, and can span across multiple runs. This is visually represented by *red* arrows connecting the

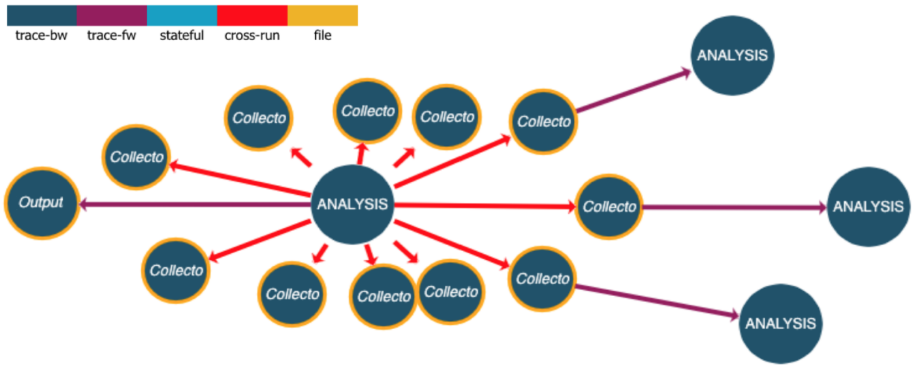


Figure 5.9: MVV. Forward provenance navigation across data contributions over multiple runs. The specific example represents a very simple analysis workflow. The workflow is made of three steps `Collector` → `ANALYSIS` → `Output`. Purple arrows shows dependencies occurring within the same workflow execution, while red arrows highlight interaction and data-reuse among different runs. The image produced with the MVV tool shows that the data-product in the centre of the graph, produced by the `ANALYSIS` component, has been read and stored by the `Output` component of the same run. The data then was read again by the `Collectors` of other executions (red arrows) to perform further analysis (purple arrow towards `ANALYSIS` components). The yellow circle indicates to the user that the provenance links to an actual data resource

.

two data vertices of the dependency graph, as shown in Figure 5.9. Setting clear visual boundaries between workflow executions, where the data has been produced and reused, may suggest further examination of the different computational methods, also considering the temporal gap between the production and the re-use of the data. For instance, in a relatively long time-frame the data could have interacted with other systems producing relevant new products or an updated version of the data itself, that may be used by the current computation. This inter-workflow dependencies can be captured when the *ids* of the data are reused across runs. For instance, we have shown how the `ProvenanceType`, presented in Section 4.3, allows researchers to reuse the persistent identifier when it can be obtained directly from the data format. (*e.g.* as implemented in the `NetCDFType`, see Table 4.1). We will come back to cross-runs scenarios in the next chapter.

5.6.4 Preview and Staging

The *Data Products and Metadata* panel allows users to gain detailed information and invoke operations on the data and its provenance. For stream-based data-intensive analysis, which is the main focus of this work, the data does not always correspond to a concrete file or resource. Especially in the intermediate phases of a computation, data is volatile in most of the cases, therefore only described by its domain and processing metadata. However, when the data is materialised transfer operations across infrastructures are facilitated by generating staging scripts based on the data products location. This can be applied to all data listed in the *Data Products and Metadata* panel. Once the script is produced its execution can be performed remotely (currently in GridFTP [26]), by referencing the active user certificate which enables delegation of data transfer services, according to the granted authorisation credentials. We foresee for this task to support also more generic clients, such as the tool delivered by the EUDAT project B2STAGE [23].

Through the MVV users also have access to the data used to initialise the workflow's execution (*WFExecutionInput*). These are stored in the workflow collection document and may also include references to the executions of other workflows whose data is reused in the current run. The MVV tool allows users to directly open the contributing runs for further analysis.

Figure 5.6 illustrates a user session showing how the different functionalities above are combined. The user interface facilitates such combination enabling a fluent user-directed exploration and use of the provenance data, fostering an increased engagement in the generation of usable provenance information during the early phases of the computational research practice. The MVV was developed as a web application by combining the *Sencha-ExtJS* and the *Arbor.js* [68, 6] toolkit. The next section will introduce the *Bulk Dependencies Visualiser* tool, which allows users to interactively control a visual analytic interface serving two comprehensive provenance exploration use cases related to single-runs and collaborative interactions.

5.7 Visualisation: Bulk Dependencies Visualiser

The interactive visualisation features presented by the MVV in the previous section are designed for in-depth analysis of large workflow runs. To obtain comprehensive views for a single workflow execution or involving many runs and users, we explore an approach to visual analytics of the information captured by the S-PROV model, that exploits radial diagrams combined with the Edge Bundles [157] technique. These show comprehensive views of the provenance repository at multiple levels of granularity and for different kinds of expertise and roles. It offers facilities to tune and organise the views.

We consider two classes of usage. In the first scenario we explain how the visual analytics approach can produce views related to a single run, while in a second scenario, users can visually explore the interactions and data-reuse between users workflows and the computational resources involved, in the context of configurable domain metadata values. In both cases we use provenance traces produced by the data-intensive workflows of the VERCE platform [97] and stored in the `s-ProvFlow` repository according to the the S-PROV model. In Chapter 6 we will cover more details about these workflows and the adoption the *Active* framework and the `S-ProvFlow` system. The implementation of the BDV was realised using *D3.js* [14], a programmable visualsation toolkit.

5.7.1 Single-run Visual Analytics

To explain this kind of visualisation, we take as example the *Earthquake Simulation* workflow implemented for the VERCE computational platform [97]. This can be divided into three phases, chronologically: data and models stage-in, input preparation and simulation, postprocessing and visualisation. Figures 5.10 and 5.11 show a series of visualisations produced by interactively querying the S-PROV model via the web API's method (14). The provenance data belongs to the execution of a simulation workflow using 512 CPUs of the Drachen HPC Cluster at SCAI Fraunhofer [65].

Figure 5.10 provides a prospective view of the workflow, which is reconstructed

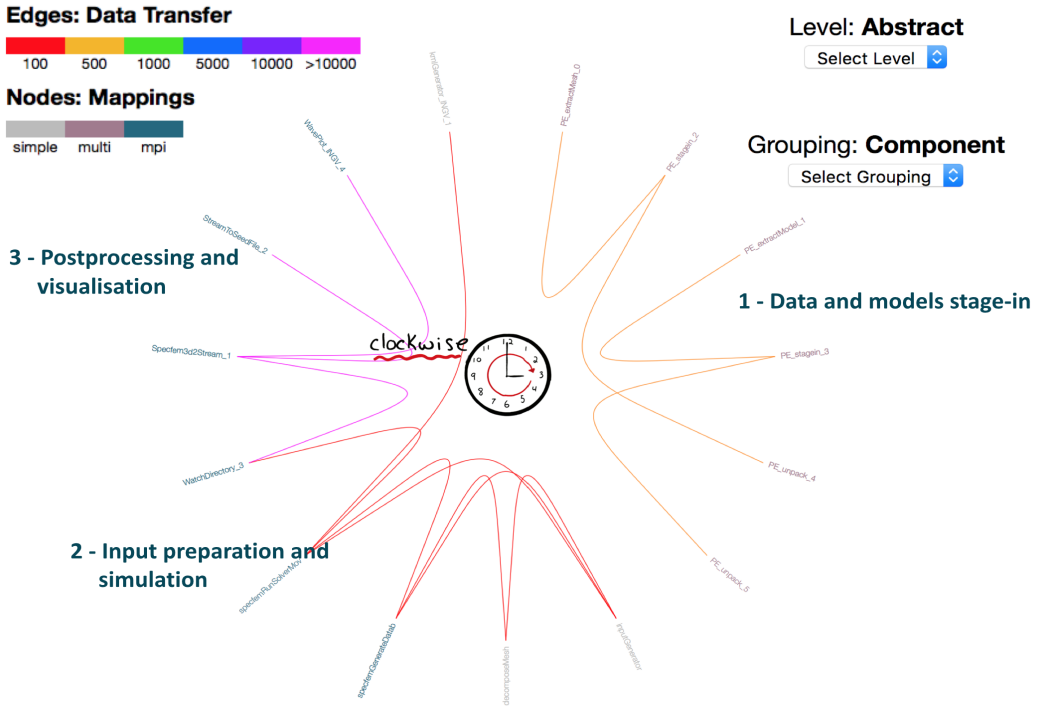


Figure 5.10: BDV. Radial edge bundles perspective of an earthquake simulation workflow. This summary shows a coarse grain view on how the worklfow's components have interacted during a workflow's execution. Users can select the depth of the perspective and the grouping rule. In this view, the nodes of the diagram on the circumference of the circle represent the workflow's *Components*, while the edges indicate connections and intensity of the data exchange. The two colour-coded legends describe respectively the amount of data transferred across *Components* and the type of enactment.

from the retrospective analysis of the provenance traces. The vertices represent the *Components* of the workflow and are placed around the circumference in the order of their activation. This is a very high-level view on the S-PROV model, which explains how the computational method was designed, besides offering coarse-grain information about its execution. The two color-coded legends for edges and vertices, indicate respectively the amount of data transferred and the type of enactment mapping (single-process, multiprocessing [56], MPI [37]).

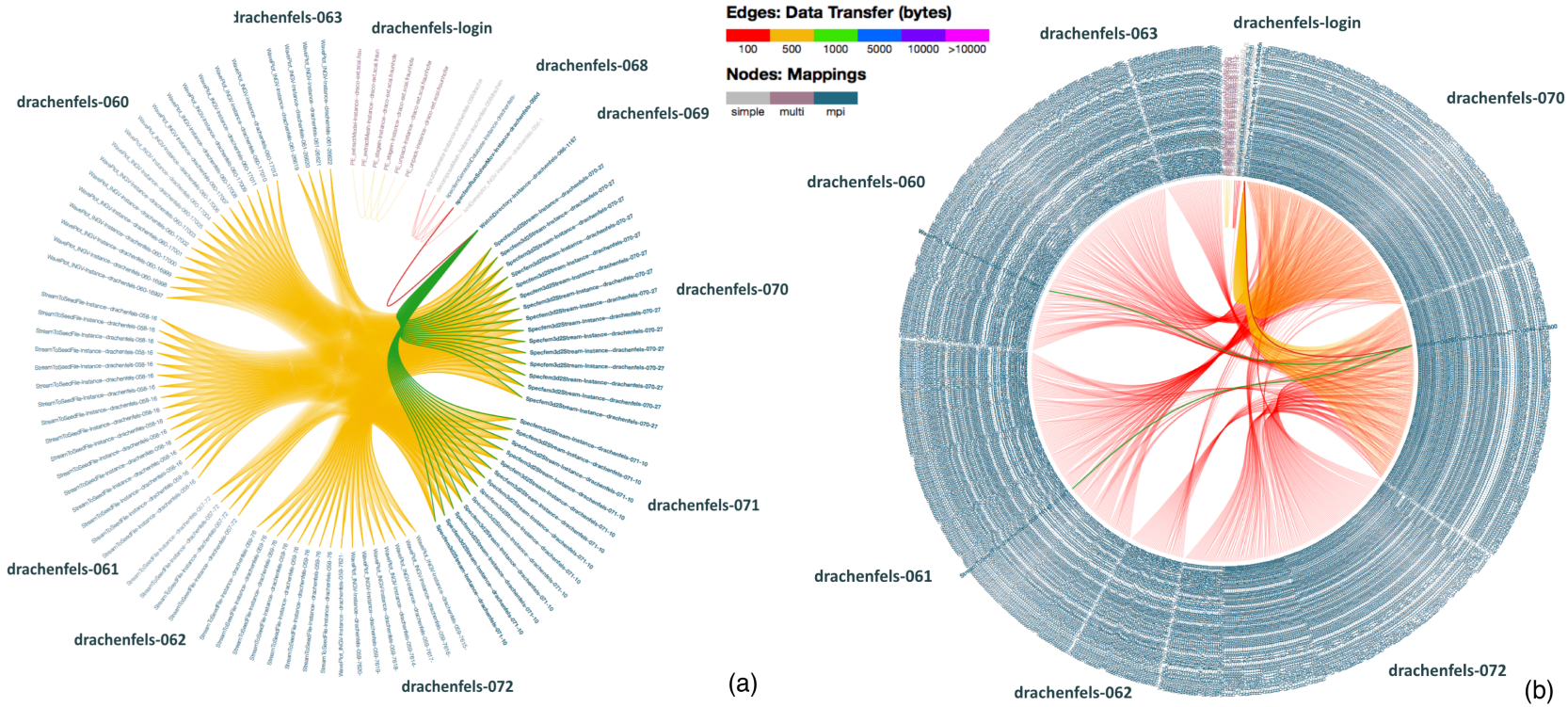


Figure 5.11: BDV: fine-grain radial edge bundles perspectives on the S-PROV model for an earthquake simulation workflow. The colour-code follows the same schema as Figure 5.10. The diagrams indicate data transfer between (a) *ComponentInstances* and (b) *Invocations* for the same workflow's execution. The vertices are labeled respectively with instances and invocations `ids`. By hovering on the nodes, incoming (red) and outgoing (green) streams are highlighted.

Figure 5.11 goes more into the details of the distribution of the actual computation. The first visualisation (a) shows how the *ComponentInstances* are exchanging data across the computing nodes of the HPC cluster, highlighting distribution patterns and intensity of the transferred information. The second (b) goes into a deeper detail, showing the actual transfers associated with each *Invocation*. Following the colour-coding, we can appreciate the difference in the data transfer shown by all the three images. The first indicates a higher values (purple color in the *Processing and Visualisation* phase), while it decreases to yellow and red in the following diagrams, showing a deeper perspective that aggregates fewer data streams.

The visualisation is implemented as a browser-based interactive tool where users can choose a grouping rule to organise the radial diagram according to their interests. Rules can be defined according to any of the properties associated with the computational metadata, which includes attribution, location, component names, *etc.* For instance, the example provided in Figure 5.11 shows a grouping by worker-node. Such views suggests that when technical experts such as research developers or advanced computational engineers, are interested in obtaining more insights about the low-level system's processes, they could group the instances in radiants associated with different details about the hosting operating system. For instance, they may produce interactively views grouped by processes' PIDs, to analyse the quantity of *ComponentInstances* running within the same process, increasing the level of detail until they the single *Invocations*. Though, as shown in Figure 5.11 (b), *Invocations* could be extremely large in number, producing very dense diagrams with the consequent loss of information. To mitigate this side effect, we allow users to restrict the visualisation to a subset of invocations by specifying *min* and *max* indexes and a starting time. These parameters will be computed respectively against the *invocationIndex* that counts the number of subsequent invocations of an instance, and their *startTime*, in a way to interactively slide through time within a range of invocations.

Views generated by this sort of query suggest ways to help a data-architect or a research-developer visually reveal exploitation patterns by recognising variations in the intensity. This is achieved by unifying aspects related to the exploitation of the resources and the logical structure of the workflow, putting the technical aspects in

the context of the scientific method and vice-versa. Overall, this may improve the users' interpretation of the provenance according to conceptual and technical interests, fostering the realisation of the positive feedback-loop summarised in Figure 4.12.

5.7.2 Collaborative Interactions

We have discussed how visual aids can support different classes of users analysing single runs. We consider now scenarios involving collaborative interactions through the reuse of data and methods. To start with, Figure 5.12 (a-b) shows how data has been reused across different users' runs. These views are interactively produced by using the API's method (15) by entering terms' values-ranges and a list of user-names. The vertices of the returned radial diagrams represent workflow executions, while the edges indicate that data produced by a workflow has been reused. As before, this is interactively indicated by the red and green colouring of the edges (input and output contribution) showing upon hovering on the vertices. The two images display different grouping rules, respectively by user-name and by workflow-name. In both groupings, each vertex gets assigned a colour to represent the runs' attribution, that is, the user. In the next chapter we will also explain views by workflow-type, in the context of the use of provenance within a more complex seismology use case involving different types of workflows implemented for the VERCE computational platform.

These options allow the personalisation of the analysis, to visually perceive what data produced by which method has been reused, but also what has not, in the context of a specific set of metadata. This may suggest the adoption of the results obtained by other workflows or, depending on the circumstance, the discard of those result that have never been reused. The combination of connections and colours make all these different interactions evident. Another interesting grouping rule is shown instead in Figure 5.13. Here the runs are grouped by the computational infrastructure where they were executed. The clusters shown in the figure were made available by the partners of the VERCE project [80].

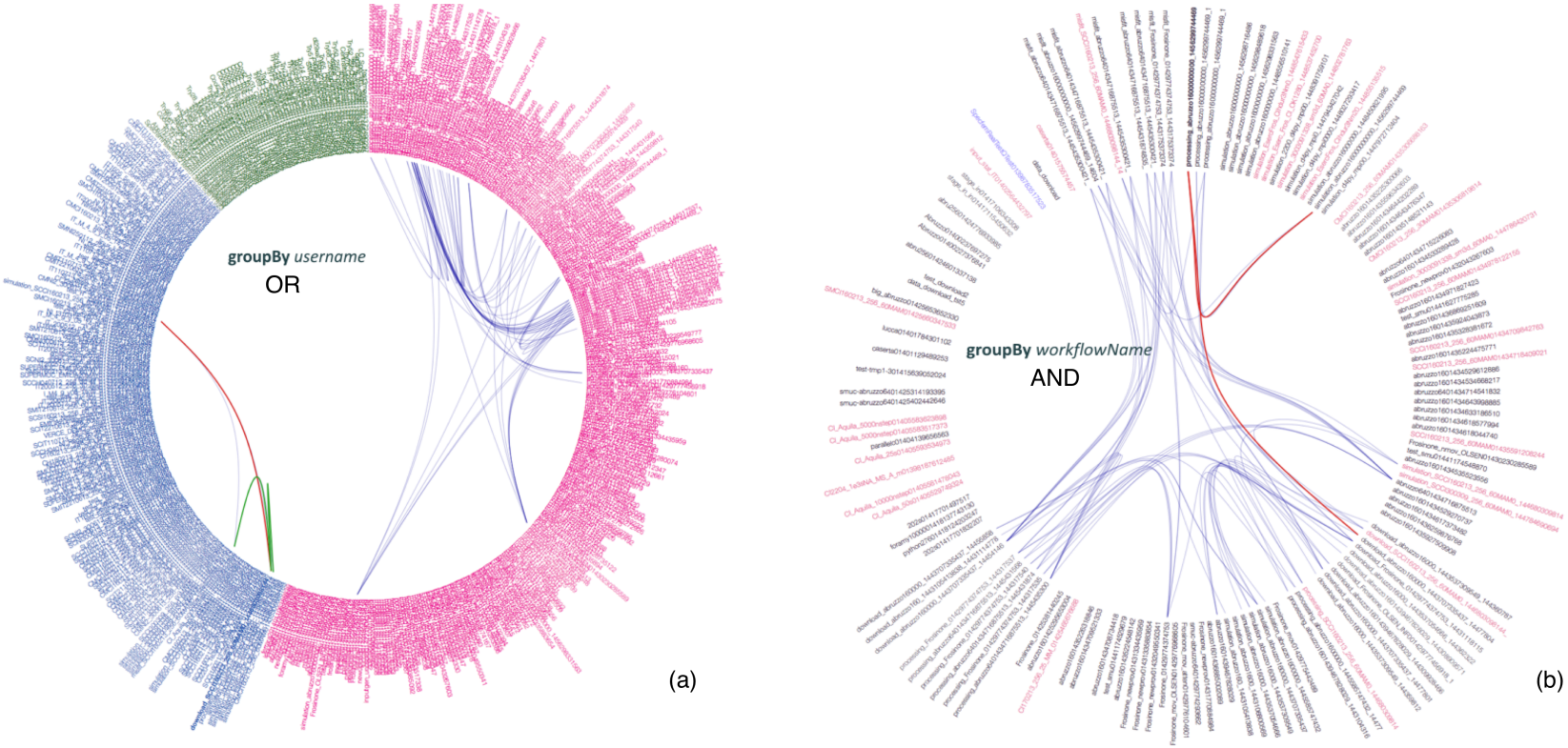


Figure 5.12: BDV: collaborative interactions among users and workflows. The diagrams are obtained by searching for runs that involved data that present metadata within specific values-ranges and applying two different grouping rules: (a) by user-name and (b) by workflow-name. Vertices represent workflows' run `ids` and edges indicate whether data has been reused by the target run. Vertices' colour indicates a specific users the run executes on behalf of. The image (b) contains fewer nodes, as result of requesting the logical conjunction (AND) of metadata values-ranges.

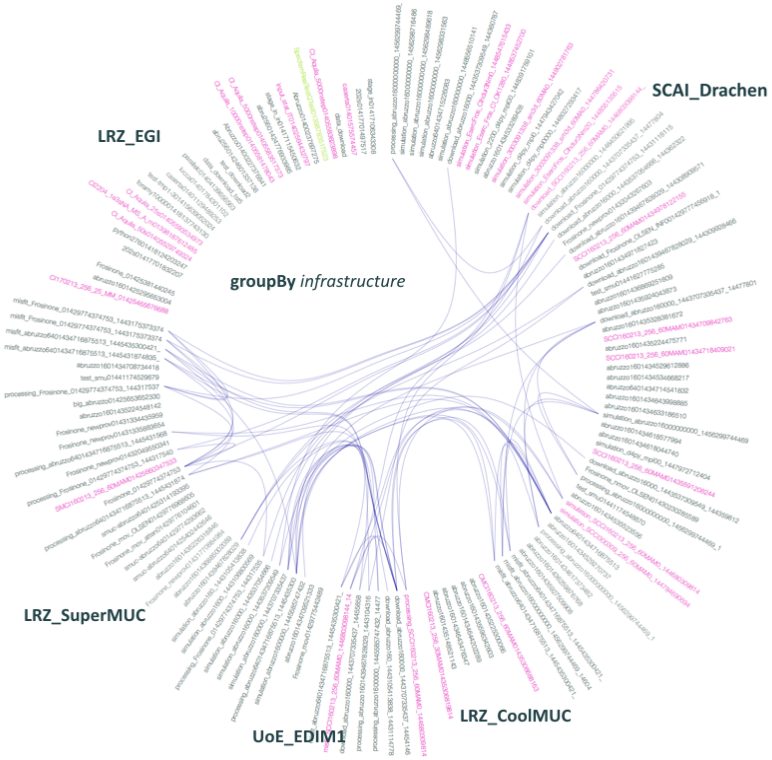


Figure 5.13: BDV: the same data as Figure 5.12 (b) is organised in this diagram to show exploitation of the participating sites in the context of the interactions among users, workflows and metadata values. The sites are named in the figure by combining the names of the organisations and their clusters.

Such visualisation suggests ways to help communities and research managers of computational infrastructures to obtain an immediate overview of the interactions across different sites. These sort of comprehensive diagrams may be used for public outreach and within official reports, providing a sensible and tangible perception of the actual exploitation of a distributed data-intensive platform, serving yet another category of consumers through the same underlying provenance model.

5.8 Conclusions

The combination of consistent provenance streaming with powerful tools delivers a smooth path between different levels of expertise by providing instruments that can

be used to explore the workflow's outcome as needed, supporting interactive access, data preview, searches over metadata and dependencies, explicit identification of data sources and diagnostics.

This is achieved through the S-ProvFlow system that manages the acquisition and dissemination of provenance data in a single exploratory space. It consists of a repository based on a document store which has been tuned and indexed with special attention to the requirements of provenance queries that access a large quantity of traces and metadata terms. The document store and the methods are exposed by a web API that allows users and interfaces easy access to common provenance interrogation use cases. We envisage future work in technical implementation of the API method, through the integration of graph databases such [39], aiming at combining flexible indexing possibilities of MongoDB with the graph traversals offered by Neo4j. Full text searching use cases could be also enhanced by experimenting the adoption of suitable technologies, such as ElasticSearch. Overall, the experimentation of polyglot solutions should be motivated by further research in provenance exploitation scenarios. Moreover, we would like to align the methods of the API with the PROV-AQ [48] standard for provenance access and query services, thereby improving interoperability by providing a service description and URI templates according to the specification.

The visualisation tools of S-ProvFlow add data-driven exploration of the provenance graph, highlighting characteristics of the computational method and its components at different levels of detail, relevant to a user's role, expertise and scientific domain. We described and provided examples of how these functionalities have been implemented in two different tools. One allows monitoring and in-depth analysis (*Monitoring and Validation Visualiser*), supporting validation, discovery, data-preview, download and staging, while the other (*Bulk Dependencies Visualiser*) offers comprehensive perspectives across large computations and collaborative interactions involving users, workflows and infrastructures. The latter explores the possibilities offered by interactive grouping combined with the radial diagrams exploiting the visual power of the Edge Bundles technique.

In both classes of tools, context specific helpers can be a shortcut to foster the uptake of these new classes of services. We have started the investigation of such functionali-

ties by designing methods, through the analysis of the provenance, that can offer hints on metadata terms to be used for discovery that might be relevant in the context of many users and experiments. We aim at adaptive interfaces, for the exploration of the computational processes that give users the potential to more easily exploit their own research in a much bigger space of interconnected systems, processes and methods, even when processes fail but yield rapid feedback.

The *Agile* framework and the `s-ProvFlow` system are used in combination to facilitate the capture, management and exploitation of the provenance information bearing S-PROV semantics. We have been inspired by the use cases envisaged for the realisation of modern computational platforms that serve different research communities with HPC and data intensive workflows and adopt standard processing services and data-format conventions. Here, the provenance mechanisms need to be integrated within complex and distributed DCIs (Distributed Computing infrastructures). We will discuss in the next chapter, how the proposed model has been concretely adopted in a number of different scenarios for the realisation of domain specific tools, to monitor workflow's executions and organise the obtained results, thus assisting users in connecting and setting-up experiments according to the specific scientific aim and terminology.

Chapter 6

Adoption and Evaluation

The progress of the work presented by this thesis was constantly stimulated by real challenges brought by the requirements of two different scientific communities in the field of climate and solid-Earth science. These communities have undertaken long-term investments in the realisation of data processing applications which are in many cases exposed through web-portals and web-services, that access different classes of computational infrastructure.

This chapter reports the direct experience from integrating the provenance solutions proposed in this work, with their working practices and scientific methods. It highlights the benefits of taking this approach. The main contributions of this chapter are as follows:

(C-4) Support for multiple levels of understanding: Each community approached the provenance information with different perspectives. Users of the climate services wanted to have the documented provenance trail embedded into the data-files, thereby prioritising the access of provenance information from the data. The seismological use cases covered a much wider spectrum of adoption, giving priority instead to the use of provenance information as a means for the management of their operations. This showed a point-of-view that considers the collection of experiments as the main entry-point providing access to the many results.

(C-5) Integration with tooling: Several technical challenges were tackled when integrating provenance extraction and management mechanisms. In some cases we had to find solutions to enable provenance recording in compliance with the high security standards imposed by the computational services, or with metadata and identifiers schemas that had to be consistently represented within the lineage. Moreover, the processing services offered by the platforms where the provenance system had to be integrated, were of quite different scale in terms of size and computational needs. These went from simple and rapid data processing operations on small portions of a dataset, to large simulations and postprocessing tasks involving many files of different formats that required long runs of multiple interconnected workflows.

6.1 Integration in Virtual Research Environments

In this section we present the direct experience of adopting the *Active* provenance framework and the `S-ProvFlow` system within existing VREs (Virtual Research Environments). We will cover two different community platforms: the VERCE Science Gateway, offering services for computational seismologists, and the CLPIC Portal, which addresses the specific use case of combining climate impact indicators coming from many sources. The two portals use different technologies to implement and expose their computational tools and they also present different levels of maturity in terms of adoption of metadata standards and provenance requirements. The challenges brought by the two communities, motivated the exploration of concepts enabling flexibility in the capture of a variety of provenance patterns and usage scenarios. Moreover the implementation required technical solutions that balanced between experimental techniques and the adoption of standards and well-established software. In this context the *Active* provenance framework, enabled the rapid and incremental refinement of the lineage contents, while the `S-ProvFlow` facilitated its representation and exposure through a programmable API. This allowed the developers of the different systems to rapidly prototype and deliver new provenance-driven functionalities.

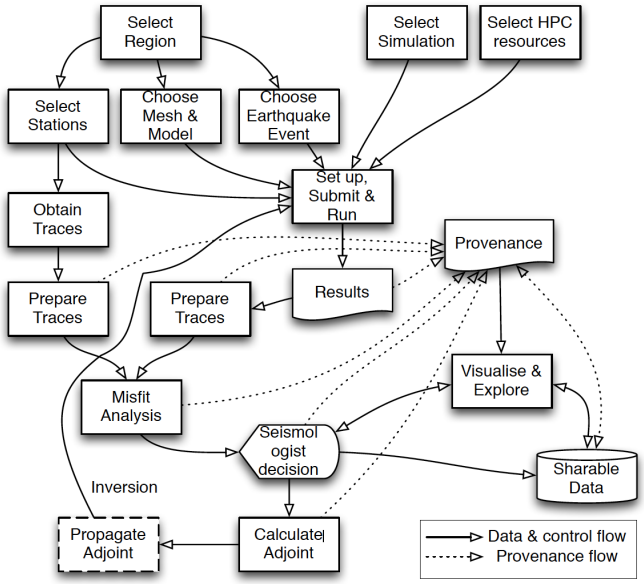


Figure 6.1: Central role of provenance information in the forward wave propagation, misfit analysis and inversion.

6.1.1 Computational Seismology

The VERCE project has pioneered an e-Infrastructure to support researchers using established simulation codes on high-performance computers in conjunction with multiple sources of observational data [97, 164]. The project's requirements for the realisation of provenance-aware seismological workflows and the exploitation of HPC infrastructures inspired preliminary work [212, 114], which saw major conceptual improvements and implementations, largely supported by the work presented in this thesis.

The VERCE e-Infrastructure is accessed and organised via the VERCE Science Gateway that makes it convenient for seismologists to use these resources from any location via the Internet. Their data handling is made flexible and scalable by two Python libraries, *ObsPy* [43, 104] and `dispel4py` and by data services delivered by the *International Federation of Digital Seismograph Networks* (FDSN) [25]. In this context, provenance-driven tools enable rapid exploration of results and of the relationships between data, which accelerates understanding, method improvements and semi-automatic configuration of interdependent workflows and workspaces, see Fig-

ure 6.1 and Figure 6.2. We introduce below the use cases supported by the VERCE Science Gateway and how provenance enables their realisation, enhancing usability.

***Earthquake Simulation*:** Users setup simulation workflows that produce synthetic seismograms using publicly available and customised Earth models and parameters from earthquake events services. The simulations are performed via the execution of HPC codes, called solvers, such as *Specfem3D Cartesian* and *Specfem Globe* [200, 70, 69].

***Raw Data Acquisition and Misfit*:** The synthetic data is compared with real observations adopting data-intensive methods. This type of analysis involves three phases: download and archive of observed data, preprocessing, and misfit calculations. Each phase can be performed separately and the results are reused across phases, for instance to test different methods. Data from seismometers is accessed from federated community services, such as the FDSN network, and stored within a intermediate data management layer, together with their processed versions, the synthetic seismograms and the analytic results.

6.1.1.1 Use of S-ProvFlow for the HPC Simulation

This specific use case involves the production and rapid extraction of provenance information about data-intensive workflows and simulations executed within HPC infrastructures offered by different service providers in Europe (LRZ, SCAI-Fraunhofer) [71, 65]. Provenance is used to monitor and to describe the phases of the simulation, from the staging of the models, the production of intermediate simulation outputs, until the postprocessing phase delivering derived products, such as images, videos and packages (*i.e.* packages containing images and *kml* files [150]) for visualisation and sharing, see Figure 6.3.

The first part of the computation is implemented through the execution of batch-processes instrumented through the WS-PGRADE workflow system [165, 164] (*i.e.* model decomposition, generation of the intermediate grid-database and simulation through *Specfem3D*). The postprocessing phase instead is delegated by WS-PGRADE to the `dispel4py` data-intensive tool. Both batch-processes and data-intensive work-

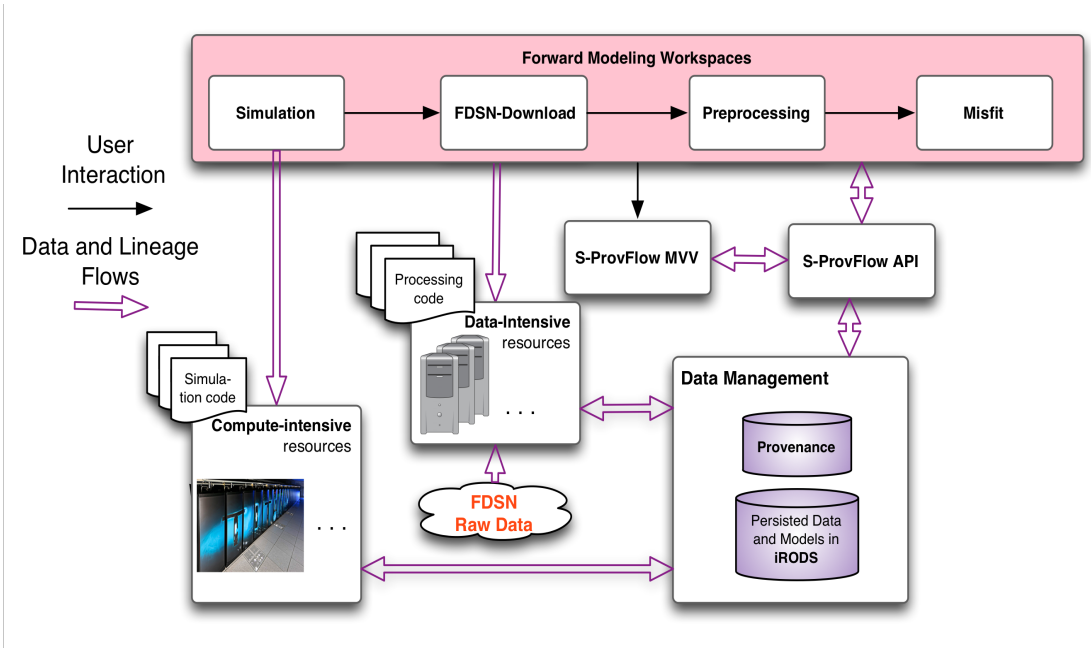


Figure 6.2: Simulation and Analysis Platform. Schematic representation depicting the *User*s workspaces and their interaction with the *System* components through exchange of data an lineage. All the workspaces, from the simulation to the misfit are controlled by the users that access interactively the provenance services to discover and combine the data produced by the previous phases (or by previous runs of the same phase), and that will be involved in the configuration and the input of the next workflow. All workspaces can be operated independently.

flows generate lineage documents that are ingested by `S-ProvFlow` and linked in a coherent trace as shown in Figure 6.4.

Given the technical and security constraints imposed by the HPC infrastructures that limit external connectivity from the computing nodes, the lineage is initially stored as files. A fork process running in parallel to the computation on a staging-node, thereby, connected to the external network, reads and transferrs the files in batches to an external deployment of the `S-ProvFlow`'s API. This allows users to monitor the execution. Moreover, the continuous and incremental access of up-to-date provenance information is combined with the characteristics of the S-PROV model and the *Active* provenance framework of sending signals to instruct selective data movements to remote resources for further analysis (Section 4.5). Thanks to this solution the users of the VERCE Science Gateway can rapidly access and evaluate the intermediate data

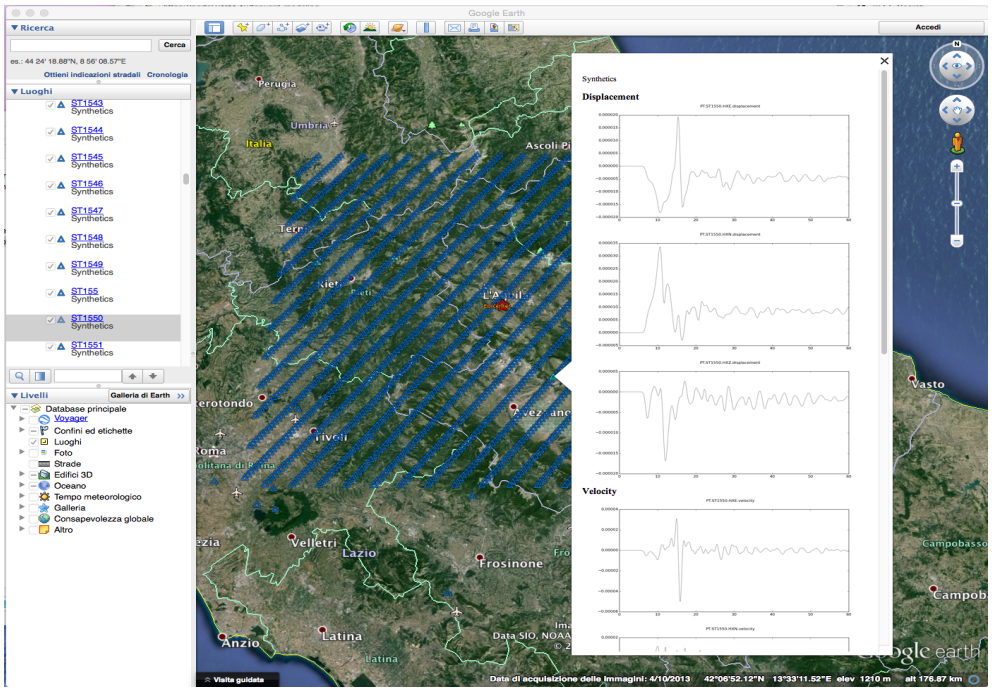


Figure 6.3: Google Earth visualisation of the results obtained from a simulation workflow for an earthquake in the region of Central Italy. The postprocessing phase of the workflow ingests and transforms synthetic data producing time-series and images for a grid of 2,212 points (19,908 single data channels). Depending on the configuration, these are typical simulations that produce approximately 10 Gigabytes of data and more than 30,000 lineage documents, the size of which is kept small and manageable by our system (around 70 Megabytes). Scientists expect to perform much larger runs to produce ensembles that will contribute to localised seismic assessments, as we will anticipate in Chapter 7. Image kindly provided by Federica Magnoni, INGV.

produced within the HPC cluster, even though it protects itself from user interaction. They can then choose whether to let the computation progress or interrupt to release expensive resources. This enables a dynamic and responsive user experience.

As experienced during training events, this allowed to have more interactive hands-on sessions, where trainees could to start looking at important outputs of the different phases of the workflow during long-lasting runs. Another feature that has been actively used by seismologists in the context of a PRACE grant [115], it is the possibility offered by the MVV of generating staging scripts based on the data products

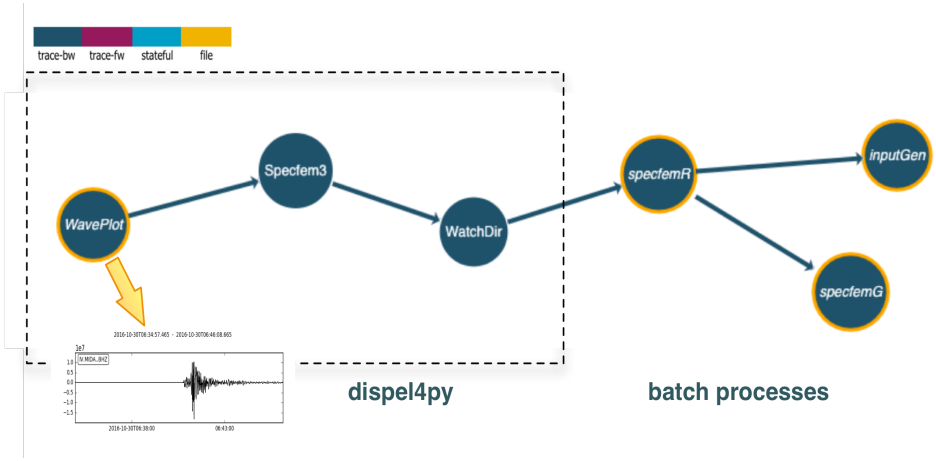


Figure 6.4: Lineage trace of a synthetic seismogram's image. The section in the box is performed by a data-intensive workflow, while the rest is implemented via batch processes. All generate lineage documents that are received by `S-ProvFlow` asynchronously at runtime and combined in a coherent visualisation.

visualised within the *Data Products and Metadata* Panel. By combining provenance searches for materialised data products and their transfers towards dedicated computational resources, they were able to reuse the data and the configuration files produced interactively through the VERCE gateway. As an example, we show below the script produced on-demand by this functionality for the secure transfer of time-series from the VERCE's iRODS-based [31] data store, to a location where the user's has certificate credentials. Paths are omitted for brevity.

```
1 globus-url-copy -cred $X509_USER_PROXY gsiftp://dir-irods.epcc.ed.ac.uk/~/<path>/IV.
      CERA.HXN.synthetic.seed ./
2
3 globus-url-copy -cred $X509_USER_PROXY gsiftp://dir-irods.epcc.ed.ac.uk/~/<run_id>/IV
      .CERA.HXZ.synthetic.seed ./
```

Moreover, through the user interface users can interactively reload the configuration of old simulations. This is possible thanks to the provenance API, which provides detailed information on the input files, their location, and the parameterisation adopted for the different phases of each simulation. Thus, by accessing the same knowledge base, users can prepare additional experiments by changing only specific details, suggesting the possibility to use the lineage data to bring improvements to their productivity. The S-PROV model in combination with the *Active* provenance framework and

the S-ProvFlow management system demonstrated to be flexible enough to represent and manage fine-grain as well as coarse-grain information, and to contribute to deliver interactive functionalities adapting to and overcoming the infrastructure's constraints.

6.1.1.2 Use of S-ProvFlow and Provenance Types for the Misfit

As we have already described, the Misfit analysis is characterised by three different phases: *Download*, *Preprocessing* and *Misfit*. The provenance data exposed through the API is used during each of these phases to gather relevant information to setup the inputs and the parameters associated with the workflows. Figure 6.2 shows a schematic representation of the flows characterising the user interaction across the workspaces controlling the different phases of the analysis, and the data-flow exchange between these and the infrastructure's components. The API component feeds the interactive workspaces that can be accessed independently to prepare and control the phases of the analysis simultaneously. Below, we describe for each workspace how the users were assisted by the provenance information to achieve their tasks.

***FDSN-Download*:** information such as stations, extent of the region of interest and time-window are extracted from the lineage of a simulation selected by the user. These details are then automatically used to configure the parameters needed to search and obtain data from the FDSN archives and pre-stage extracted data into the data management layer.

***Preprocessing*:** once the data is downloaded, its provenance is used to assist the interactive setup the input of the preprocessing workflow by matching simulation and observed data. Users select a simulation run from the GUI, displayed in Figure 6.5, and are presented with a list of potentially relevant download runs. This is based on the assumption that different simulations performed in a certain area and at a certain time may reuse the same observational dataset for the *Misfit* analysis. Moreover, data is not always provided with the same quality and availability from the providers, thus the access to the time-series may be performed multiple times and used for different evaluations and comparisons. The GUI shows the user's descriptions, which also contributes to give an im-

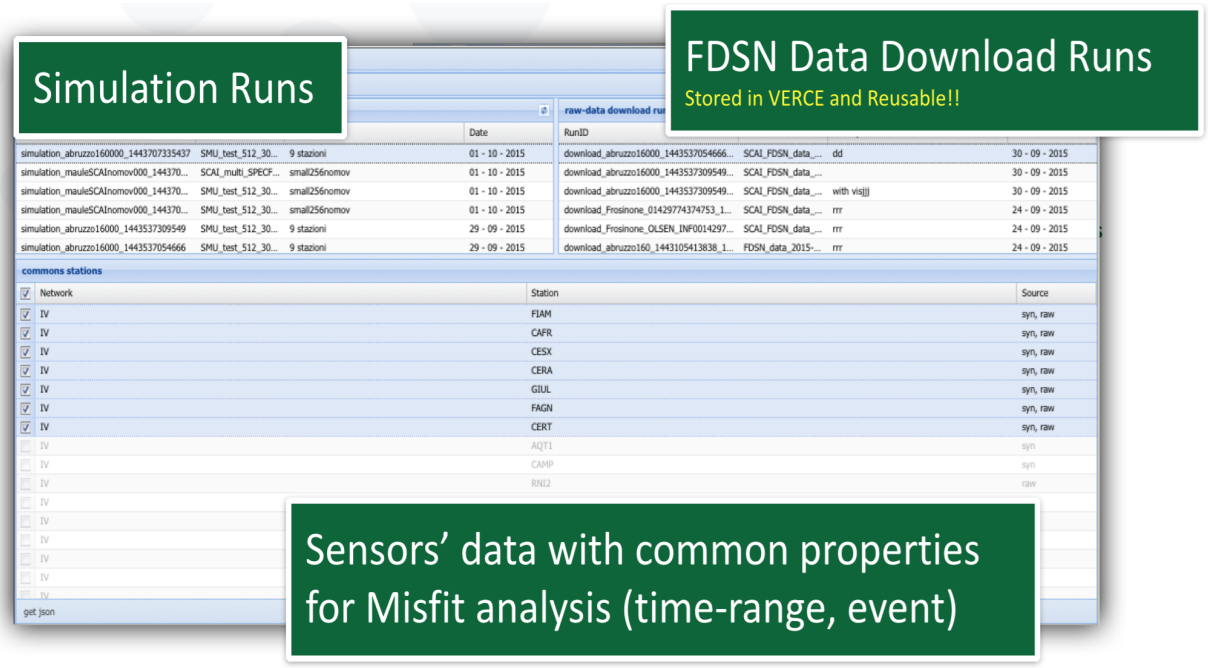


Figure 6.5: Combine Simulations and Observations. Part of the *Preprocessing* workspace GUI where the provenance API is queried to suggest to the users which data to select that have been previously obtained from simulations and FDSN-Download workflows. Good candidates are highlighted on the basis of their contextual metadata.

mediate indication about the results or the characteristics of the run. After the users selects the download run, the GUI queries the provenance again to automatically present a list of stations generated by the two workflows that are good candidates for misfit analysis, on the basis of their metadata. Thus, by explicitly suggesting the reuse of an existing pre-staged dataset, the tool exploits the provenance archive to contribute to the users' productivity, also trying to reduce large download operations from the data provider.

Once the data is selected, the interface allows the researchers to compose a processing pipeline, where they can checkmark each step to indicate whether the intermediate result has to be visualised or saved. Once the processing workflows are composed and configured the users submit and control the results at runtime by interacting with the provenance information. Figure 6.6 shows how the steps that produced the visualisation of the data are highlighted by the MVV.

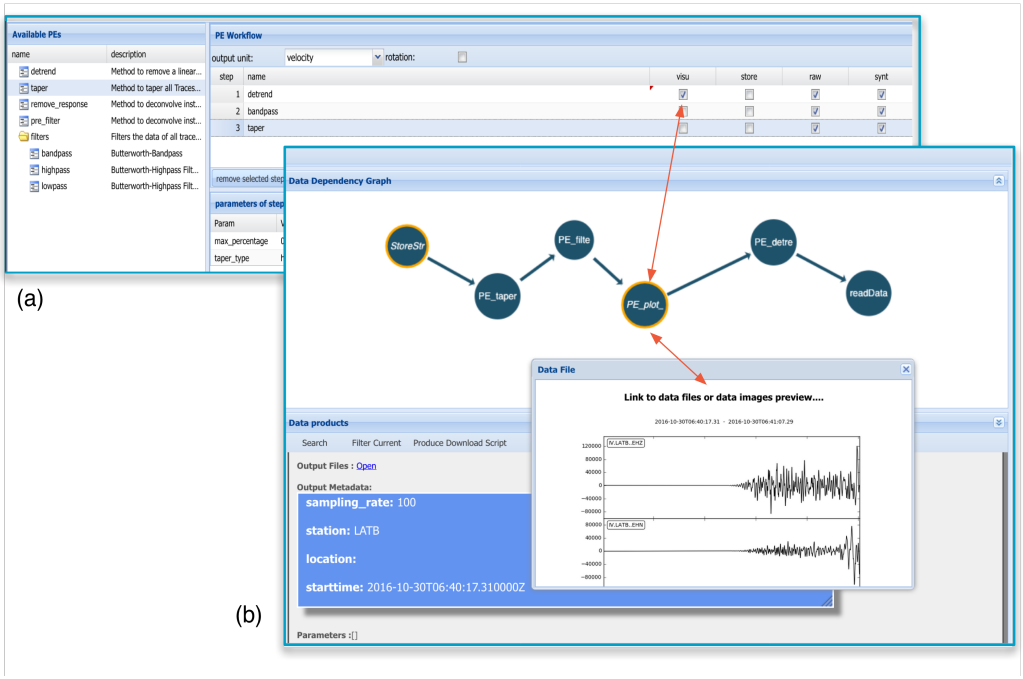


Figure 6.6: Misfit *Preprocessing*. Users compose their data processing pipeline specifying intermediate visualisation steps (a). The outcome is made discoverable and accessible by the MVV tool, which provides in a single view the image, the complete lineage and domain metadata (b).

Typical preprocessing operations include: patching small gaps, removing instrument response from the observed signals, normalising and band-pass filtering.

***Misfit*:** in the last phase of the analysis provenance is used to show to the users which data preprocessing pipelines have completed. They select one of these runs to automatically configure the input data for the misfit workflow and choose which type of misfit analysis should be executed. For instance, as a preliminary investigation, they may ask to calculate and visualise exclusively the time window on which the comparison should be computed.

Figure 6.7 shows the abstract workflow and the trace associated with this choice, indicating that this is translated into a specific routing of the data within the workflow. The MVV offers to the users the possibility to visualise the resulting image and its lineage and to access the provenance of those workflows that provided the input data for the current run (associated with the simulation, download

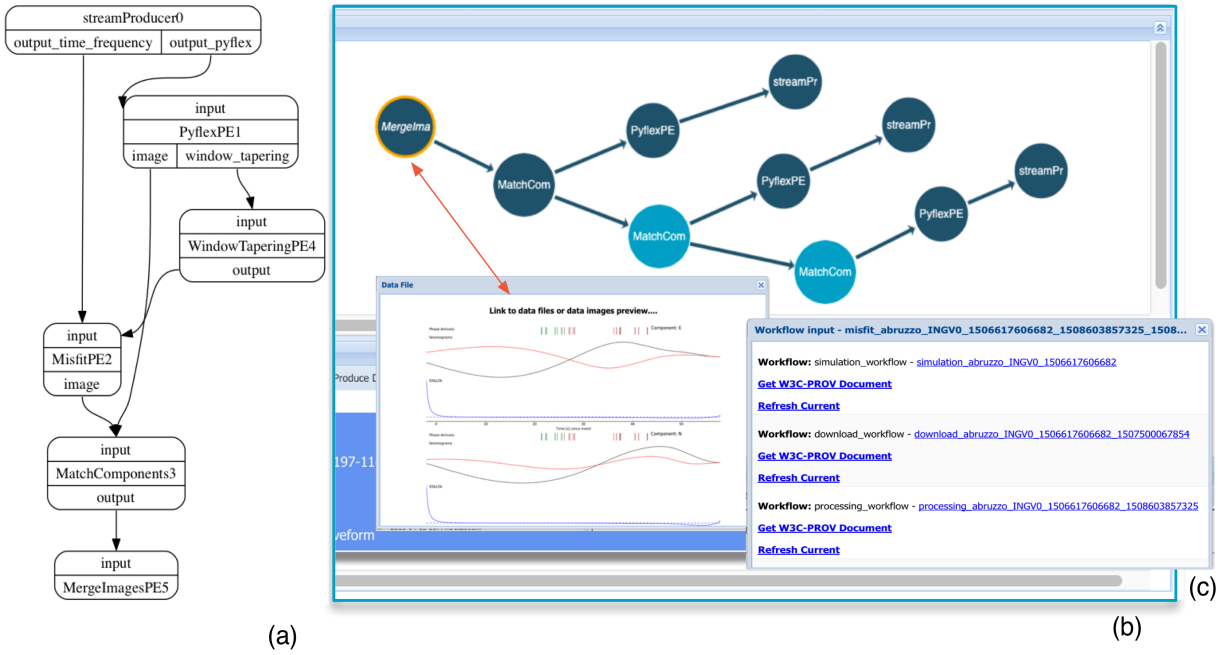


Figure 6.7: Misfit Results. Abstract Misfit workflow (a) and lineage of one of the resulting images (b) that summarise the comparison (*Pyflex*) of observed and simulated seismic channels on the computed time window. The input data and the access to the trace of the other phases of the misfit analysis are made available for further investigation (c), including the possibility to export their provenance in PROV format.

and preprocessing phases). These can be selectively opened by the researchers and analysed.

Going into more detail of the logic behind the Misfit workflow of Figure 6.7, each sensor is composed of three channels or components (Vertical, North and East). The `streamProducer` extract data from the local archive and otputs a stream of observational and synthetic channel paris. The `PyFlex` [54] operator receives the pairs and calculates the windows on which the comparison should be performed, including the associated image. These are then passed to the `MatchComponents`. This is a grouped operator on the image's properties, such as the sensors' `id` and `image-type` (allowing for workflows that produce more images per pair). It combines all the images of the same type for the same station `id` and finally sends them to the `MergeImages` operator, that generates the final

visualisation. Thus, in order to match the right images, arriving without any specific order, the `MatchComponents` preserves an internal state. This is a use case where the provenance type `ASTGrouped` (Accumulate State Trace Grouped, see Table 4.2) is used to capture the underlying lineage pattern, *i.e.*. dependencies are partitioned by groups of inputs, and traced as state updates, until the next output is produced.

Thanks to its generic functionalities, the MVV is adopted as a single results management tool for all the phases of the Forward Modelling analysis, allowing users to monitor and validate their workflows and to access the generated outputs through provenance exploration. We have exposed the system to users at different stages of its implementation, within informal sessions and during official training targeting students and researchers in seismology [81, 22]. These were able to use the computational platform independently and to conduct real research tasks. These concerned particularly studies of seismicity in regions such as Central Italy [115] (Figure 6.3), Maule region of Chile [142] and Nepal.

Moreover, through the BDV we produce comprehensive views about the activities of the Science Gateway highlighting the interactions between the different workflows of the Forward Modelling (Figure 6.8), involving users and computational infrastructures in the context of a specific combination of metadata terms and values. This allows researchers to obtain an overview of similar computations and their impact on the progress of the research across users. For instance revealing good datasets to be reused in their analysis.

To summarise, the flexibility in the provenance collection model based on self-contained lineage documents allowed users to obtain rapid feedback on the executions' progress. The adoption of provenance types helped in precisely representing the lineage information with no loss of detail and consistently with the behaviour of the stateful components. They have been used extensively also for the automatic extraction of seismic waveform metadata, according to the vocabulary and data-format adopted for the processing. However, these could be extended with additional and experimental terms according to the requirement of the different scientific workflows, *e.g.* to classify the different types of misfit with relevant metadata and to describe the

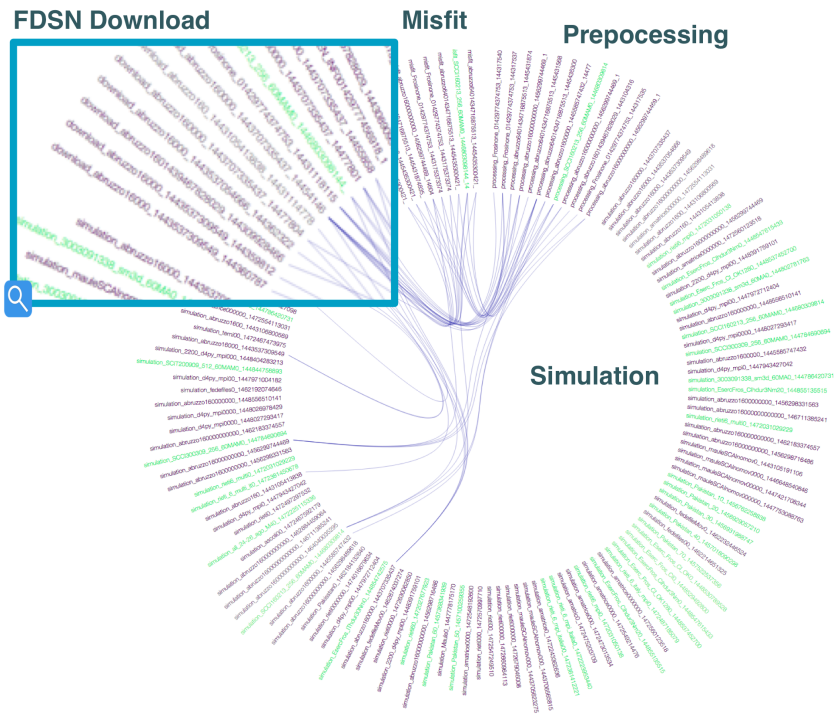


Figure 6.8: BDV. View of the interactions between different computational seismology workflows grouped by their type: *Simulation*, *FDSN-Download*, *Processing* and *Misfit*. The colour of the vertices indicate different users, while the magnifier shows a download run whose results have been reused by many preprocessing tasks, suggesting the presence of a good dataset or the target of a particular investigation.

parametrisation of each component.

In Table 6.1, we report usage statistics extracted from the production deployment of the `S-ProvFlow`' repository for the VERCE platform. This has also been used to store lineage produced by experimental workflows for realtime seismic correlation analysis, a service that is not currently exposed through the Science Gateway.

This experience suggested substantial benefits brought by designing a VRE around an holistic model of provenance, which is exposed by easy to use service abstractions. The latter allowed GUI developers to implement dedicated interactive workspaces as provenance-driven tools, while more generic interfaces, such as the MVV, proposed new ways of engaging users in monitoring, discovering and accessing the results, ob-

taining rapid access to the information concerning the reuse of data between the different phases of their analysis. Tuning the collection and the visualisation at different levels of granularity can serve different expertises and tasks depending on the role and responsibility in the VRE's operations. Researchers, developers and managers can use the model from different perspectives from the early stages of the platform's exploitation. Experimental data that can be manipulated, combined, reused and validated in the context of the users' domain and the specific computational task. Enabling "by-design" the early adoption of provenance information that integrates standards, as well as new concepts fosters the natural and incremental migration to the long-term curation of the results. This was achieved by adopting an intermediate repository of provenance as the core knowledge-base, whose content can be selectively extracted and summarised by dedicated operations, allowing users to add further annotations prior to the official publication. The design of these classes of operations are considered as future work and are already envisaged by projects such as SEAD [195]. However, the underlying compliance to the PROV concepts is a fundamental requirement.

Usage Statistics in the VERCE VRE	
Metric Description	**Quantity**
Total number of *WorkflowExecutions*	1,641
Total number of lineage documents	2,129,194
Total number of *Users*	49
Different types of workflows	6
Size of the Database	16 GB
Max number of lineage documents associated with a single run	185,328 (Correlation Analysis), 79,644 (Simulation)
Total number of *DataGranules* metadata terms	90

Table 6.1: Usage Statistics on the `S-ProvFlow` database deployed for the VERCE portal. The hosting and the resources required for the operations of the database are performed by SCAI-Fraunhofer. Statistics have been extracted with the kind support of André Gemünd on July 2017.

6.1.2 Climate Indicators Analysis

Modern analysis platforms for Climate studies provide generic and standardised ways of accessing data and processing services. These are typically supported by a wide range of formats and interfaces based on OGC [44] standards. However, the problem of instrumentally and consistently tracing the dependencies of the transformations occurring on a dataset remains an open challenge. It requires these standard-driven and interoperable services to facilitate the reproducibility and understanding of the output produced by simple as well as more complex tasks involving self-describing data formats.

The CLIPC portal [10] offers a climate impact toolkit to evaluate, rank and combine climate impact indicators. These synthesise effects of future climate change with relevance to a specific sector and business. Applications include research or decision-support in policy and practice. For instance, they are used to help governments to inform farmers in a particular area how to adapt their land management, or to help to shape a civil-engineering project to reduce flooding. Indicators are produced with data coming from several categories: satellite measurements, terrestrial observing systems, model projections and simulations and from re-analyses.

CLIPC allows users to *combine* indicators into a new indicator. For instance, they can add up climate impacts or create a difference map. In such a context, clarity of provenance is fundamental. Besides the documentation on the technical quality of the original data, on metrics related to scientific quality and on uncertainties and limitations of the data, the service required an instrumented and systematic provenance system that could capture the interactive execution of a *combine* function driven by the users. It had to be able to extract the relevant metadata and to link all the data products and the intermediate results that lead to the production of a new indicator [186].

This project belongs to the Copernicus Earth Observation Programme for Europe [13], which will deliver a new generation of environmental measurements of climate quality. It has a backbone combining OGC WPS (Web Processing Service) and *OPeNDAP* [199, 84] services. The climate toolkit is realised by the orchestration of a number of processes that ingest, normalise and combine *NetCDF* files. The WPS allowing this

specific computation is hosted by the *Climate4impact* portal [32], which is a generic climate data-access and processing service.

6.1.2.1 Adoption of Provenance Types and Export Service

In this context, guaranteeing traceability is a clearly stated requirement to improve the quality of the results obtained by the combined analysis. The contribution was made by developing a workflow that captures provenance assertions and data dependencies consistently with the adopted conventions and metadata. Hence, the workflow, that can be exposed as a WPS, delivers interoperable provenance reports as part of its *NetCDF* output files, which are the new impact indicators.

Listing 6.1: Configuration of the Combine Indicators Workflow. Both the components of the `clipc:Combiner` cluster have the `NetCDFType` to deal with metadata and the consistent referencing of the datasets' `ids`. The `COMBINE` also adopts the `Nby1Flow` pattern type to handle lockstep reads.

```
configuration = {
        "s-prov:username":    "aspinuso",
        "s-prov:description" : "provdemo combine",
        "s-prov:workflowName": "demo",
        "s-prov:workflowId"  : "workflow_combine1",
        "s-prov:save_mode"   : "service",
        # Define the provenance types
        # for the components
        "s-prov:componentsType" : {
         "COLLECTOR": {"s-prov:prov-cluster":"clipc:DataHandler"},
         "ANALYSIS":  {"s-prov:type":(NetCDFType,),
                       "s-prov:prov-cluster" :"clipc:Combiner"},
         "COMBINE":   {"s-prov:type":(NetCDFType, Nby1Flow,),
                       "s-prov:prov-cluster":"clipc:Combiner"},
         "STORE":     {"s-prov:prov-cluster":"clipc:DataHandler"}
        }}
```

The approach pursued was to develop, through the *Active* provenance framework a contextualisation type, `NetCDFType` (Table 4.1), by specialising the `extractDataSourceId`, `extractItemMetadata` and `makeUniqueId` methods, in order extract all the relevant metadata, including those of the new indicator, and to handle the identifiers of the input and output files. This was combined with the `Nby1Flow` (Table 4.2) provenance type, in order to handle streams of inputs to be combined in lock-step, for instance, in

a parallel execution of the workflow. All these information had to be consistently used in the lineage. Figure 6.9 depicts the trace of a new indicator file, while Listing 6.1 shows the configuration used, according to what we introduced in Section 4.5.

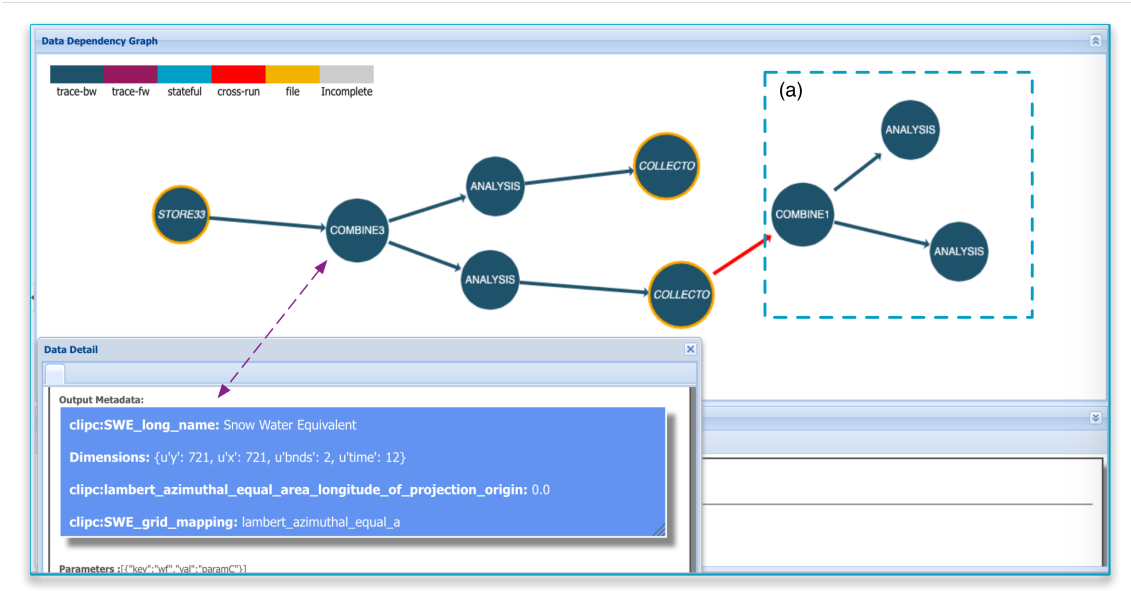


Figure 6.9: CLIPC. MVV visualisation of a portion of the lineage of a climate indicator obtained by the workflow implementing a *combine* function. The use of the provenance `NetCDFtype` and `Nby1Flow` for the `COMBINE` component guarantees the consistent representation of the lineage. The square (a) shows that an indicator produced in a previous run has been reused in the current computation.

Lineage documents are eventually sent to the `S-ProvFlow` repository. At the end of the computation the provenance trace is extracted from the API in PROV-XML format for its storage within a dedicated attribute of the *NetCDF* output. The trace can be accessed and visualised from the portal, see Figure 6.10. As suggested by one of the patterns proposed by the RDA provenance working group [61, 7], instead of embedding the trace into the file, it would also be possible to directly link to its URL as specified by the API.

Results' management and validation benefit from the customisable extraction and attribution of the `ids` to data resources. For instance, in Figure 6.9, while the provenance entity attributed to the collector already links to the actual data resource (yellow circle), its dependency refers to the combine stage where the indicator was actually

computed, providing immediate access to the processing and metadata details. This avoids the capture of redundant metadata by those components of the workflows that handle the files (`clipc:DataHandler`), producing informative traces.

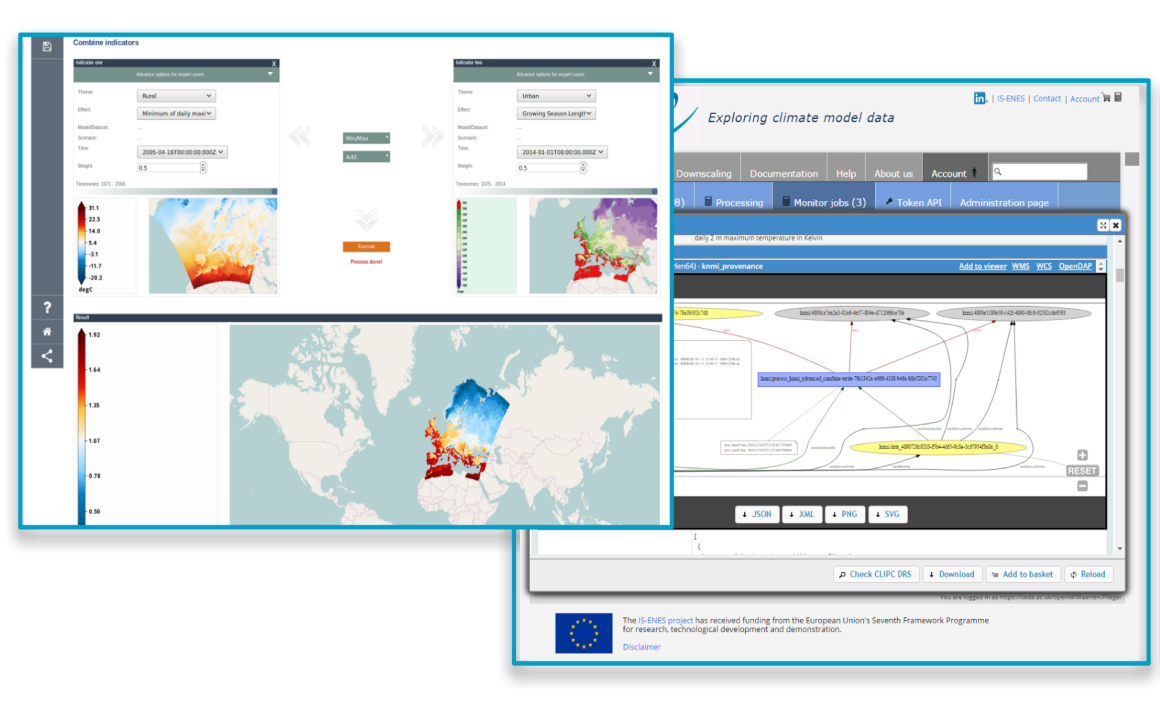


Figure 6.10: CLIPC. GUI for the *combine* function and a view of the provenance trace for the resulting climate indicator within the *Climate4impact* portal.

Overall, we have tackled the challenges posed, by offering as part of a generic computational and provenance-aware framework the consistency of the information stored in the lineage with the data-format's conventions. The workflow had to guarantee to capture the data-dependencies established by iterative executions of the *combine* function that produce multiple outputs. Eventually, results could be enriched upon request with their lineage traces, through the selective extraction from the provenance archive. Details of this implementation are included for demonstration within a *Jupyter Notebook* page available in GitHub[1].

As a more general consideration these two concrete experiences showed that in computational platforms serving a community of users, the provenance-enabled systems

[1] `https://github.com/aspinuso/dispel4py/blob/master/NetCDF-comb-Demo.ipynb`

are part of a much larger picture where innovative specialists in many domains need to be supported to recognise and reap the potential of their growing wealth of data. Data gets reused, moved and combined by using remote processing infrastructures and data-management services, therefore also common provenance related tasks need to be conducted by using remote services that offer flexible and high-level abstractions, and include standardised provenance data within their results.

6.2 Feedback Collection and Workshop

On the 19th of January 2017, we ran, on behalf of KNMI, a workshop on provenance requirements and practices for climate-data processing services. Participants were affiliated to organisation in the public and private sector. The list is reported in Table 6.2. We only mention the number of participants from each organisation for privacy reasons. They contributed by presenting their approach or by participating in the discussion on their expectations for the systematic collection of provenance, its coverage and exploitation use cases.

Workshop of 19th of January 2017 — Organisations		
Organisation	**Type**	**Participants**
BSC, `https://www.bsc.es`	Public	4
B-Open, `http://www.bopen.it`	Private	1
University of Cantabria, `https://web.unican.es`	Public	1
ECMWF, `http://ecmwf.int`	Public	2
KNMI, `http://www.knmi.nl`	Public	2
MeteoSwiss, `http://www.meteoswiss.admin.ch`	Public	1
Predictia, `http://predictia.es`	Private	1

Table 6.2: List of organisations attending the workshop, their type and the number of affiliated participants

The main scope of this event was to present and discuss the *Active* framework, and the `S-ProvFlow` tools that support it. The workshop emphasised practical aspects asso-

ciated with the wider application of the system. It demonstrated its current capabilities and introduced the S-PROV model at its foundation. During the event, the participants were encouraged to openly discuss the presented work, with the support of a SWOT matrix (strengths, weaknesses, opportunities, and threats) [132]. This had the advantage to connect, during the informal evaluation, factors that are internal and external to the specific community, taking into account priority of requirements, intellectual ramp and potential for adoption, considering basic and more advanced use cases. After the workshop, informal email communications addressed the details of the model representation and implementation. Some organisation also recorded and returned their feedback in the form of the SWOT matrix. These are BSC, ECMWF and Predictia.

6.2.1 Demonstration Setup

Practical demos were performed by means of a *Jupyter Notebook* page, shown in Figure 6.11 and whose updated version was previously mentioned in Section 6.1.2. The session exposed participants to the features of the *Active* provenance framework, the underlying model and the visualisation tools. The examples used in the demonstration ingest data in *NetCDF* format from an external *OPeNDAP* service, perform the data analysis and store the results in a file on the local file system. In a second phase the same workflows could be setup to reuse the file just produced.

Through the notebook, inputs and provenance configuration could be easily changed to apply different setups to show the effect of no-provenance, basic provenance capture and the additional functionalities offered by the typed approach.

1. the lineage trace generated with the `NetCDFType` produced a richer (and more precise) provenance content that included the domain metadata automatically extracted from the *NetCDF* attributes;
2. the lineage associated with the output produced by iterative executions of the workflow on the previous results showed a longer trace, backtracking until the the original data accessed from the *OPeNDAP* URL;
3. the possibility of obtaining (and visualising) the provenance in PROV format.

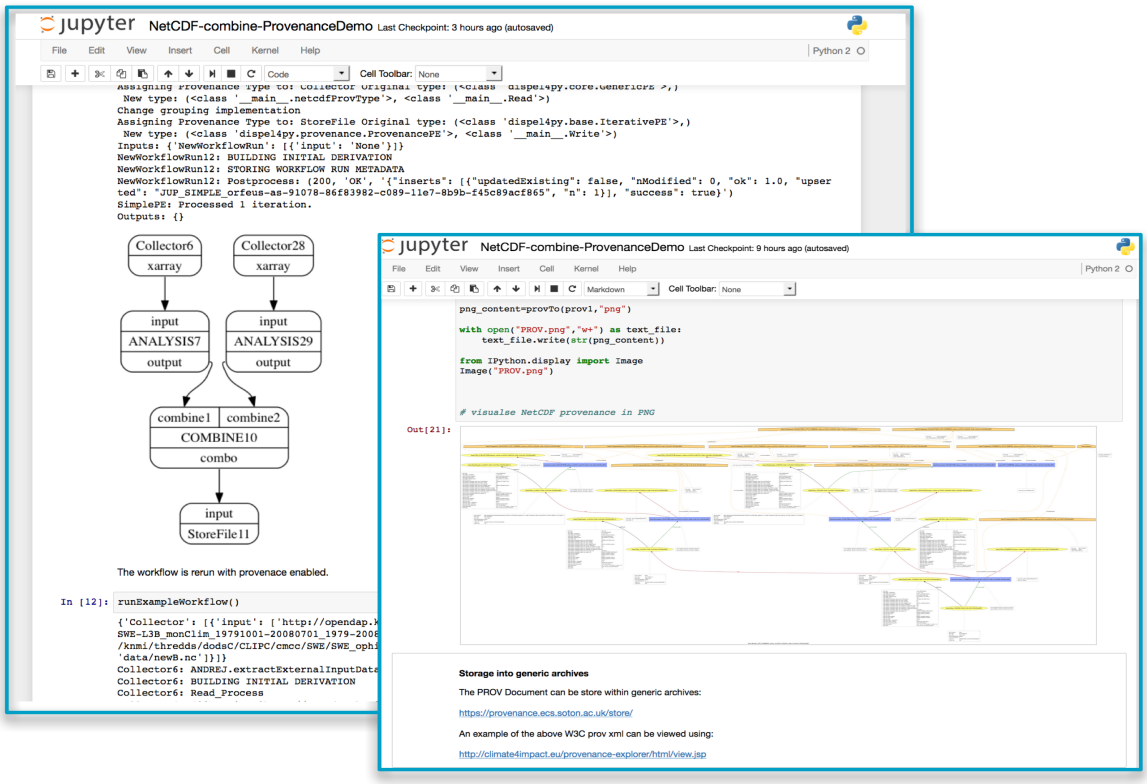


Figure 6.11: Jupyter Notebook pages used to illustrate the use of the *Active* provenance framework and the `S-ProvFlow`'s services.

Finally, after showing the visualisation of PROV documents according the the typical representation offered by a generic toolkit [3], the results were also accessed within the MVV, to demonstrate its interactive approach to the exploration of the provenance information that exploits the variety of methods of the API and the characteristics of the underlying model.

6.2.2 Evaluation and Discussion

We requested the attendees to provide feedback about aspects of the *Active* provenance framework, the model (S-PROV) and the visual-tools. We find it convenient to use the SWOT classification to interpret and summarise their comments and discussions. Table 6.3 shows the sample topics associated with each element of the matrix.

SWOT Matrix — Topics	
Strengths - What is done well? - What advantages are provided? - Strong research and development capabilities?	**Weaknesses** - What areas need improvement to accomplish the objectives? - What does the system lack?
Opportunities - What opportunities exist in our environment that can benefit from the system? - Is the perception of the system positive? - Has there been recent growth in interests?	**Threats** - Who are the existing or potential competitors? - What could prevent the system from being used?

Table 6.3: SWOT Matrix and suggested topics.

Strengths: The *Active* framework was considered powerful, flexible and supported by significant in-depth understanding of the provenance challenges. The attendees appreciated the possibility offered by the visual tools to explore the details of the results, which also allows them to export the lineage selectively in an interoperable representation. The experience in integrating `S-ProvFlow` technology within real domain platforms was also considered inspiring and worthwhile. The latter fostered their awareness about the possible benefits of provenance, as a sophisticated knowledge base that helps VREs to put users in control of their results and operations, exposing if they require relevant technical details.

Weaknesses: The *Active* framework may leave too control to users' or research developers' choices. They should be trusted in the way they describe and set the detail and precision of the provenance data. Its current implementation targeting the `dispel4py` library should scale to other technologies to satisfy the technical requirements of an open and multi-community system. The support for a domain metadata schema which is characterised by a dictionary of properties that does not include an internal hierarchy might not satisfy more complex scenarios.

Opportunities: Data provenance is complex to manage in all research areas and com-

putational scenarios, the framework successfully shows a way towards a more comprehensive solution. The presented setup could be proposed as a guide to discuss domain-specific data provenance structures and management. Possibilities of cooperation with other similar initiatives are envisaged, for instance with the work pursued by the QA4Seas [57] project (Quality Assurance for Multi-model Seasonal Forecast Products).

Threats: Some of the participants raised the issue that it could be hard to persuade domain-experts that some of the low-level provenance (related to distribution and processing details) data is useful for their work, making the abstractions less relevant to domain scientists.

Among the weakness we discussed the risks associated with enabling the users to setup their own provenance configurations, questioning whether a completely predetermined approach would be easier to adopt. However the participants acknowledged that a fully automated solution would only be possible for very specific applications, leading to an implementation that may hardly be reusable in other contexts. We know from the literature that in a generic computational framework, the automation of the provenance collection bears the risk of producing less precise, thereby less usable traces. For instance, to improve how the workflow associates results with the right input parameters [91], it is required to identify potential provenance issues in the workflow structure, thus asking users or developers to revise their implementations.

As a reaction to this possible weakness, we have produced an extended library of provenance types, by using the methods illustrated in Section 4.3, and including the possibility for precise refinements through explicit management of the provenance state. We foresee that new types will be added during the further exploitations of the *Active* system in upcoming and running projects (*i.e.*, the DARE H2020 project [15]), identifying new general patterns and increasing the variety of combinations of pattern and contextualisation types. For what interests instead the support of hierarchical domain metadata models, we should mention that the storage system can potentially ingest more complex structures. Though, the current API should be extended to allow expert users to directly query these particular schemas.

Concerning the issue mentioned in the threats, the role of the human expert is con-

sidered extremely relevant in our approach to provenance capture and representation. Experts' contributions are facilitated by the *Active* framework by the possibility of extending the metadata with new experimental properties and semantic characterisation at run-time, aiming at making the core concept as close as possible to their understanding. This goes along with the possibility offered by the model of explicitly representing those aspects that are related to low-level processing details, such as distribution and delegation, allowing multi-layered views of the computation. For instance, from the discussions that took place during the workshop, the flexible combination of scientific and computational information would be beneficial for the *Copernicus Climate Data Store Toolbox* (CDS Toolbox) [175], which will be offered to experts as an online service to execute methods on a large and constantly growing amount of data products stored in the CDS.

This event inspired a number of followup workshops in Toulouse and Barcelona (March and June - 2017), where the collection and definition of provenance use cases was progressed. These targeted quality control aspects of climate products, as well as more general use cases related to the *Copernicus Climate Change Service* (C3S) [112]. Further events are planned for the 2018, where the participation of representatives of the RDA Provenance Patterns Working Group will be strongly encouraged. To conclude, all the participants at the workshop acknowledge the importance of providing provenance-driven interactive tools to smooth the intellectual and technical ramps experienced by the communities approaching new classes of computational services. These insights were suggested by showcasing our general solution in the context of a deployment within real infrastructures serving specific scientific use cases.

We pursued and still engage in intensive interactions with the scientific and technical experts implementing modern climate services. Here provenance matters are discussed by bringing in the experience matured with the work presented here and the cooperation will continue beyond the finalisation of this thesis, with more results pursued during the next concrete initiatives. We observe that whenever the rationale behind our approach for a flexible management of data-intensive provenance was communicated, either officially or informally, this led into further engagement. This is confirmed by the continuation of the discussions, workshops' followup and, according to projects'

resources and planning, by the adoption of the proposed technical solutions. For instance, after the initial experience of CLIPC, which is a C3S pre-operational initiative, evaluation for the adoption and further extension of `S-ProvFlow` are taking part in the context of the MAGIC project [34]. This specific initiative aims to deliver an operational data analysis service for climate-models information.

6.3 Conclusions

The evaluation of the work presented by this thesis focussed on the application and adoption of the concepts and their technical implementation within existing online computational platforms, serving seismology and climate-impact studies. In both contexts, to achieve engagement and mutual understanding, the system was exposed regularly to scientists and developers, bringing evidence of the benefits and the breadth of the use cases. The seismological service provided the most challenging and interesting scenarios where the *Active* framework and `S-ProvFlow` could be evaluated in almost every respect. The system is used to manage the results contributing to the experimental evaluation of solid-Earth structural and velocity models. This is obtained by executing large-scale simulations of earthquakes and then comparing the synthetic data with real observations (Forward Modelling and Misfit). The scientific task is decomposed into more workflows that adopt HPC and data-intensive technologies, relying on heterogeneous types of computational and data infrastructures. We have explained how the lineage was collected for batch processes and for streaming data-intensive workflows, capturing the behaviour of stateful and parallel streaming operators. We illustrated how the lineage is collected and stored at runtime within the `S-ProvFlow` system, and how it is accessed from the API for the discovery of the workflows' results needed for the preparation of the new phases of the analysis. Finally, we also demonstrated how the interactive tools could facilitate, in a single exploratory space, to monitor, validate and manage the experiments, offering users the means to visualise and trigger the transfer of the results of interest. Thus, the facilities enabled by *Active* provenance and `S-ProvFlow` yield two significant benefits:

1. The seismologists were able to control and manipulate their work, to examine

the quality of their methods and the validity of their results by adopting a generic provenance framework.

2. They found that the tools and visualisation provided by `S-ProvFlow` automated or simplified tasks they would otherwise have to undertake; thereby delivering early benefits from using provenance, as advocated by Myers *et al.* [195]

The other direct experience addressed climate processing applications associated with the Copernicus Climate Change Services. Here, the main challenge was to deal with requirements that put self-describing formats as the main resource of data, metadata and lineage information. Thus, conventions had to be taken consistently into account when integrating provenance-awareness, starting from simple processing tasks to more complex scenarios involving the reuse of results. We demonstrated how the *Active* framework could adapt to these requirements in the context of the combined analysis of climate-impact indicators that ingest *NetCDF* data-sources. This was demonstrated by discussing the adoption of contextual and provenance patterns types and by showing the capability of the `S-ProvFlow` system to extract a lineage trace as a PROV document, thus facilitating its inclusion within the *NetCDF* container. Although we argue that this not an ideal solution given the resulting limited use of the lineage and its potentially large size, we could deal with this requirement to address this specific community's priority. Alternative solutions decoupling the provenance from the data have been discussed and are currently addressed within dedicated working groups. We demonstrated the system and encouraged discussions in a dedicated workshop where engineers, scientists and managers of climate services took part and provided useful feedback. The workshop was supported by training and explanatory material in the form of *Jupyter Notebook* pages, showing the components of the framework in action. By providing this interactive documentation, we could concretely show the possibility offered by the framework, in combination with the `dispel4py` computational library, that allows users to design their methods while being facilitated in taking responsibility for the quality of the provenance produced for their results.

As already discussed in Section 5.4, co-design sessions with the involvement of domain experts and GUI developers were favoured to a formal usability evaluation. Though, it would be valuable and potentially applicable during the ongoing initiatives

to perform a more quantitative analysis. We promoted frequent participation of expert users in the refinement of the service, and pursued the realisation of a provenance framework that was actually used to drive the daily experimental practices. This experience suggests new ways to scale to a wider variety of modern computational portals, contributing "by-design" to an assisted long-term reproducibility and understanding of the methods. Finally, combining experts' contributions with outreach events and trainings with new users, exposed our system to the improvements of the underlying model, toolset and implementations. It will continue to be shaped by more advanced requirements, further comparing with other concrete approaches and supporting their technical realisation in future work. The established trust and evidence of the benefits perceived by the early adopters, made it possible to involve the aforementioned communities in followup events and in the acquisition of new related projects. We will briefly cover new upcoming initiatives in the next and last chapter of this thesis, in relation to the envisaged future work.

Chapter 7

Conclusions and Future Work

In this chapter we present the summary, the concluding remarks and the lessons learned. These have matured during the years of study and the implementation of a comprehensive provenance framework tuned for data-intensive research methods. We have been through several iterations and improvements mainly driven by the agile co-design and development with users and research developers. This established the conceptual elements through experimentation, while pioneering a new technical background. The resulting framework is able to accommodate current and future evolutions in the way provenance is integrated, represented and exploited, from the early stages of the computational research life-cycle onwards, see Figure 5.1. We investigated a wide range of scenarios, from tuning and short-loops supporting rapid validation and steering by users and developers, to the long-term curation of the scientific results and the analysis of the interactions between users' methods, data and computational resources. We had to be capable of offering to researchers sufficient details on their method's behaviours. This captured information relevant to the scientific domain and the scale of their workflows that are transparently mapped onto a larger network of infrastructures. All these resources are exposed through virtual environments to allow the configuration and control of scientific tasks, as well as the development of completely new methods. This required high-level use cases to be identified and implemented through interactive tools and service interfaces.

7.1 Computational Seismology Test Case

The motivation for the work presented in this thesis was triggered by the need to develop a new computational platform that had to support: a) the management of user's workflows and experiments; b) the provision of a metadata catalogue that, by showing the relationships between methods and results, could be interrogated to configure different stages of the scientific investigation; c) the flexibility to support evolving experimental methods, products and software. These facilities fostered learning behaviours and improved productivity when researchers exploited the resources offered by complex and distributed e-infrastructures. Our approach matured during a six-year community effort, which is still continuing. It aimed at delivering a platform for computational seismology studies (VERCE) [97]. This was followed by a preliminary evaluation and experimentation with a different class of online computational services addressing climate studies.

In VERCE, we had to help users execute and validate results obtained by experimental simulation and analysis code, where neither metadata standards and data-formats, nor the choice for a single underlying computational technology was expected to be consistently chosen throughout the development of the project. However, we understood the underlying computational model advocated by the computational experts, its distributed nature and the variety of configurations. We recognised the generic nature of such challenges to support both rapid innovation and production runs for research that uses both compute-intensive and data-driven methods. To achieve this, the services should be able to capture and communicate effectively the characteristics and the conditions of the systems when and where the data was produced. This information can be represented by standard provenance models [188], such as PROV[83]. Thus we had to ensure libraries and system used delivered provenance to these standards. This is necessary in a variety of user context including the rapid prototyping, validation of scientific methods and the associated interactive tools. These tools, which exploited the provenance data, were eventually integrated within the platform and exposed as a service for wider use.

In such a context, and during the lifetime of the project, scientists in seismology

started to produce new algorithms and batch jobs, while data architects and HPC experts developed a new data-intensive streaming library [134]. The library allowed applications to scale-up and apply different resource mapping modes. It presented very few constraints on how users could ingest, group and manipulate data, and offered no support for provenance and lineage capture, annotations and metadata. This posed the challenge of a very interactive and dynamic environment combining many skills, backgrounds and interests, with many challenges for an effective integration of scientifically relevant and reusable provenance information within sophisticated technical solutions. Moreover, we had to take into account infrastructure constraints in security and protocols, especially when rapid and detailed feedback and computational steering started to turn from a secondary requirement to a crucial demand. This provided more evidence that provenance information could act as the spinal cord of a nervous system providing impulses to adaptive computational and validation tools. It offered an approach to the management of the results which allows for incremental refinements, as envisaged by projects such as SEAD [195], one of the major motivators for our work.

Eventually, by facilitating provenance practices "by-design", we contribute to the interoperable traceability and reproducibility of the results, one of major challenges which is not yet considered fully addressed in the management of scientific research [99, 192, 120, 113]. Provenance should be helpful for the automated construction of required metadata and provision of interworking. However, domain researchers need to control selection, timing and quality of results that are offered for "FAIR" [221] access. We wanted to foster reproducibility on the conceptual and technical level, considering the delivery of products, whose is favoured by a platform that can integrate methods where the control from the developers, the core researchers and the end-users is assisted by a comprehensive and holistic provenance model and management framework. The VERCE scenario is typical of many emerging application domains, thus we recognise and consider the potential for a wide adoption of the approach presented in this thesis, which is already being exploited within new initiatives.

7.2 Computational Models and Interoperability

In this thesis we address scenarios where online scientific gateways expose research tools that are implemented through technologies such as scientific workflows, with special attention to computational libraries that enable the distribution of data processing tasks across concurrent streaming operators. The provenance of such systems should be described by an holistic model offering multiple levels of detail to serve different exploitation use cases, depending on users' expertise and their role within a larger infrastructure.

The investigation of a model of provenance suitable for our objectives was presented in Chapter 3. It started by taking into account the abstraction offered by the PROV and ProvONE models. We then developed more detail concerning the underlying computational model and the characteristics of the abstract and concrete operators of the system that we want to capture. This helped define the semantics of the *observables* that should be included in the data streaming model of provenance. This was then formalised by extending and re-using when possible the classes and relationships of PROV and ProvONE. This lead to the definition of S-PROV, which addresses aspects associated with delegation of the abstract workflow components, seen as agents, to their running instances (actors). We included in the model elements to represent and manage the actors' internal state. These extensions covered data resources that are updated and reused across multiple invocations. Moreover, we considered dynamic scenarios were a distributed workflow may change the behaviour of its components at runtime, for instance by means of re-location, re-implementation and re-parametrisation. The application of which should be further investigated. A set of operations defined on the model were described. These support common provenance patterns, with special attention to stateful scenarios. As an example of the application of some of the properties of the S-PROV model, we present retrospective analysis, diagnostic and stream-reordering use cases. The latter addressed the data processed within concurrent and asynchronous processing pipelines.

Comprehensive provenance and the complexity of its integration and adoption had to be addressed by offering a framework characterised by a type and a profiling system

offering reusable patterns and customisation. This was made usable via management system and a set of exploitation tools, so that developers could easily choose the appropriate trade off for their current work.

7.3 *Active* Provenance for Assisted Usability

In Chapter 4 we explored ways of offering semi-automated mechanisms that help those introducing these practices from the early stages of the creation and evaluation of their methods. This enabled contextualisation and selectivity of the scope and the nature of the lineage which users considered relevant. It allowed them to tune the precision of the capture of the data dependencies within complex and stream-based computational operators. We call this approach an *Active* provenance framework for data-intensive computations. It presents a conceptual framework based on developer-defined *ProvenanceTypes* and user *Configurations*.

Research developers are encouraged to create libraries of re-usable provenance types that capture the behaviour of basic and complex components, as well as extracting the metadata associated with the produced data. Advanced and more specific uses of the framework enable their libraries to inject additional runtime annotations and model more complex dependencies, accommodating to different scenarios by using simple extensions. The framework hides all the complexity of dealing with the dependencies and of consistently recording attribution to the delegated operators. It offers an easy way to access and link to the provenance assertions related to stateful data resources. We have presented an implementation for an existing computational library, `dispel4py` [134]. We discussed an engineering approach that makes use of user-configured profiles with semantic annotations applied through dynamic typing techniques. The former facilitates the integration and use of the provenance type system, with less impact on the computational library and the operators. The latter allows to group and map the workflow's components to concepts associated with the user's universe of discourse, as well as to external ontologies specific of the domain of application. The characteristics of the framework were demonstrated using a workflow for correlation analysis (CAW). The type system empowers research developers with

tuneable and precise provenance production and use. Domain scientists use the provenance to validate each phase of their analysis and to discover the products needed for the preparation of the subsequent tasks. Given the suitable tools this has proved to be effective, especially when adopted via a virtual research environment to assist researchers in meeting their needs incrementally and efficiently.

The selective and precision aspects of the *Active* framework were justified to match the scale and relevance of the lineage to the users' interests and the progress of their experimental investigations. We illustrated how the run-time analysis of the metadata can enable operations on the data, such as transfer to other target resources, as part of the fundamental services of a provenance-aware system. We believe that this aligns well with the objective of a data fabric infrastructure, which has among its requirements to support unification and to accommodate new standards (metadata standards in our case) as they become relevant and to move the data to the right application or service at the right time. In that respect, Mattmann [182] identifies imitations in the case of HPC applications for data-intensive computing. These should become active entities which can trigger behaviours and state changes in external services while they run. We addressed this challenge by enabling selective data movements and exploring ways of integrating runtime metadata analysis through a computational framework that adopts a unified model of provenance.

7.4 Managing and Exploiting Lineage Collections

The integrated provenance management and toolset, `S-ProvFlow`, was introduced in Chapter 5. This is a system that aims at facilitating users in the management and exploration of their experimental results, in relation to their computational methods. Through its API it fosters the implementation of new domain specific tools to improve the overall effectiveness of the virtual laboratory by enabling humans and the software agents, acting on their behalf, to exploit information associated with their research activity communicated as provenance streams.

The API exposes the underlying database which has been implemented with a document-

store (MongoDB). We have experienced the advantages of such technology, which thanks to its flexible representation of the information in JSON format, allowed us to focus on the implementation and evaluation of new features in short development cycles. However, with the growth of the provenance collection and the variety of metadata and values dynamically entered to the store, we experienced a loss in the performance, especially on those methods that query on domain and user-defined metadata. This suggested the need for further work (in collaboration with a student enrolling in a technical internship) to improve indexing strategies in combination with adjustments to the internal representation of the lineage. This delivered to the users a more responsive access to the underlying information. Query simplification was brought by the denormalised approach encouraged by the document-store technology, which through embedded map-reduce postprocessing possibilities allowed us to create new aggregates addressing specific use cases. For instance, as one of the outcomes obtained by supervising the aforementioned student project, the API was extended with the possibility of suggesting terms, values and their use, that are relevant to the experiments attributed to a single user or to a research team, facilitating collaborative efforts. This was motivated by the desired flexibility of the `S-ProvFlow` system that has to automatically accommodate new metadata vocabularies and experimental terms. Further investigation on combined polyglot solutions is envisaged, as we describe in the next section.

We would like to pursue further work on these (and other) visual-analytics techniques, especially to improve the interaction between the BVD and MVV in support of decision processes related to the reuse of the data and its deletion. We assume the lineage data associated with a workflow's execution as immutable. Though, the API supports the deletion of the provenance of entire experiments. We should consider that in some scenarios this may affect the provenance of those runs that show dependencies with the ones deleted. This suggests a use case that considers the possibility of summarising provenance traces before deletion, so the references to the deleted runs can reach "tombstones" with summary information. We need to take into account that within collaborative virtual laboratories, the experimental results could have storage requirements which may not be affordable in the long-term. However their potential of showing evidence of their relevance should be protected.

The S-ProvFlow is thereby offering new opportunities for the management and exploration of the provenance information. Tools can be developed around its API, which should be extended to support additional high-level abstractions for annotation and direct comparison of workflow runs. It could provide metrics to feed into cost functions for optimisation or into tools for planning the provisioning of resources from the enactment platform. The existing tools already show how computations can be monitored and evaluated interactively at different levels of detail, visually combining computation and scientific metadata with users' processes. The database technology and the adopted representation proved sufficiently flexible to accommodate the rapid implementation of the use cases. It also facilitated the interconnection, through the API's methods, with external semantic services and metadata catalogues.

7.5 Current Outreach and Future Contributions

The possibilities and potential of *Active* provenance have been incrementally refined and demonstrated through implementations for specific application domains. We foresee the need to incorporate these extensions into standards through a larger community effort, to build a powerful library of types and patterns and to explore the use of other platforms. The supporting framework and the integration of tools deserves further development and optimisation.

The experience of deploying provenance practices in support of computational services for these initial communities gave many insights. It was very important to put to the fore-front the provenance information that is relevant to a specific user role, in order to foster engagement and to discuss new adoption scenarios. By creating further abstractions, such as additional provenance patterns to be included as reusable provenance types, we could reduce the need for the research-developer to implement them. Explicit inline dependencies, established through provenance state instructions, were still useful when evaluating complex methods that required the inspection of relevant intermediate results. The approach aims at improving the synergy between different concerns and user groups, by delivering an innovation and production context that enables domain scientists and research developers to work more closely together. We

need to continue working on meaningful ways to map low-level and detailed lineage to higher level summaries. In that respect, linkage from and to external semantic catalogs is crucial to establish contextual mapping and to extend information about experimental methods to established descriptions of services and agents.

We foresee multiple phases of exploitation, which will support the development of the methods first, sharing of intermediate results, finally towards the exhaustive and assisted packaging of self-contained research objects [102]. Provenance services should provide hints about how to query the provenance itself by anticipating the interest of the end-user. This applies to annotations and experimental terms, but also to the linkage between multiple experiments, which may have exchanged relevant results overlooked by the user. Visual analytics techniques of large traces should include further possibilities for customisation, from the perspective of the users' interests. We envisage a provenance annotation workspace supported by an extended section of the current API. Here, interactive tools enable users to annotate large sections of the provenance collection to analyse execution patterns, thereby identifying and classify interactions between methods, data and user groups. Tuning and incremental refinement of the gathering of the lineage should be supported by interactive environments. We believe that the static analysis of the workflow and its single operators [91], should consider the role of the expert, who may want to further annotate, enrich or selectively ignore dependencies at runtime.

These, among others, will be part of the challenges investigated in the project DARE [15], inspired by the work presented in this thesis, and that as an outcome sees the continuing engagement with research teams in seismological and climate communities. The project aims to provide researchers with a unifying platform for user-friendly and reproducible progress of huge data-driven experiments, that benefit from rapid prototyping and evaluation. DARE wants to support teams of research developers and scientists, who work at the intersection of software engineering and scientific domains. It will serve communities dealing with climate and seismological modelling addressing specific use cases. Climatologists will generate a multi-model, multi-scenario, time-series average datasets of the Earth's surface temperatures for Western-Europe, using CMIP5 [11] data. They will process high-resolution dataset with daily temperature

for a time-span of five to ten years, from 1950 to 2100. Input files are stored on distributed nodes of the ESGF federation [19] and each model produces 5-years files of approximately 1 Gigabyte. We have estimated an input data volume of 300 Gigabyte to be processed in stages of location-dependent workflows, accessing close-to-data computational facilities for data-reduction tasks, as well as intermediate and on-demand resources for the generation of the derived products. Eventually, data-reduction will be very substantial. For instance, final reports about the anomaly of the temperature in a specific region between two periods and across tens of models, may consist of a single file of one Megabyte. Compared to CMIP5, which counts in total 1.8 Petabytes involving 4.3 million files distributed in 23 ESGF nodes, the first phase of CMIP6 simulations will increase the scale by a factor of 20, with 36 Petabytes in 86 million files (simulations will begin in 2018). These figures have been kindly provided by Christian Pagé, CERFACS [8].

Seismologists, instead, will develop methods for the Estimations and Rapid Assessment of strong ground motion (ES - RA) for emergency response. On demand ensemble simulations that require 20 to 200 million CPU-hours will produce tens of Terabytes of data per run. This will allow the rapid characterisation of the Seismic Source (SS) in order to support decision-makers in localised hazard assessments. In the former use case, provenance mechanisms are required to produce metadata-rich traces for the tailored data-products, which are linked to the data life-cycle across the workflows located in the different sites dedicated to the data-reduction and processing. The seismologists demands for robust provenance-driven tools to organise, explore and reuse of the results derived by the ensembles and the rapid assessment analysis, with flexible management of metadata for detailed and ad-hoc validation of their methods. The holistic system will facilitate comparative studies and will complement the rapid response to societal demands with trustworthy evidence and advice. Details about this use cases have been kindly provided by Federica Mangnoni, INGV [29].

Considering these numbers, in DARE the management of large-scale lineage information will be among the main objectives. These challenging requirements will motivate further improvements to the underlying technical solution. We will pursue the development of solutions that guarantee a good balance between scalability and

ease of use of the adopted format, with the support of effective storage and query techniques to serve the existing and new functionalities offered to the end-users. We foresee that new provenance exploitation scenarios, will benefit from the experimentation of polyglot database solutions. For instance, in order to deliver personalised recommendations when searching or trying to reuse results and methods, we could extend the postprocessing provenance capabilities of `S-ProvFlow` with the combination of document-store and graph databases [39], as well as exploring RDF based solutions. Especially the latter, is envisaged in DARE, when we will start exploring the use of *SemaGrow* [67], to integrate multiple provenance resources delivering combined views and summaries. Visual-analytics tools will benefit from this integration and will be further extended to implement new use cases. They will produce responsive and interactive summarisations of interconnected roles and experiments performed by peers. Evidence of their influence and impact on the delivered products will be tailored on the target scientific community, offering insisghts on the exploitation of the underlying resources besides the in-depth exploration of the fine-grained dependencies. This will progress together with further improvements of the implementation of the *Active* provenance framework, in terms of performance and usability. We will pursue future research to extend its adoption to other computational tools. Possible new targets could be the `EXAREME` [24] dataflow processing system and the *Climate Toolbox*. The latter, briefly introduced in Section 6.2.2.

We envisage the integration of interactive tooling to achieve FAIRness incrementally and in a developer-friendly framework. Thus, facilitating productivity, reproducibility and the exchange of ideas. Moreover, by further investigating the architectural solution proposed in Section 4.8, which foresees in-workflow provenance sensors, we would like to add new operational features exploiting run-time provenance analysis. This may involve reactive statistical methods in support of computational steering use cases. This could bring to the improvements of the workflow settings in a DCI at runtime, motivating further investigation on the traceability of the occurring changes. Finally, as illustrated in Figure 4.12, each aspect of the envisaged future work should contribute to the exploitation of provenance models throughout all phases of the digital investigation, achieving the concrete realisation of a *positive-feedback loop*, where users and intelligent systems cooperate in the progress of the computational research

cycle. We will continue the collaboration with running projects such as EOSC-pilot [21] and will contribute to the use cases and objectives of the Research Data Alliance (RDA) [61].

7.6 Provenance-powered Futures

The power of detailed and controllable provenance recording has been demonstrated. It yields substantial benefits for researchers who became actively engaged and for developers who can implement highly productive patterns. Operations and events occurring within computational systems that support adaptive deployment and runtime steering can be traced thereby recording the relationships with the information pertaining to the scientific context. This provides the underpinning communication channels to improve collaboration between different categories of experts across diverse operational platforms. As today's challenges demand increased performance to support more sophisticated methods, provenance-powered data-intensive platforms will grow in importance and be widely applied, especially in combination with collaborative VREs and web-based development tools.

Publications

Parts of this thesis have contributed to the following publications, abstracts and presentations in international conferences and workshops.

1. **A. Spinuso**, J. Cheney, and M. Atkinson. *Provenance for seismological processing pipelines in a distributed streaming workflow.* Proceedings of the Joint EDBT/ICDT 2013 Workshops, 2013.

2. S. Gesing, M. Atkinson, R. Filgueira, I. Taylor, A. Jones, V. Stankovski, C. S. Liew, **A. Spinuso**, G. Terstyanszky, and P. Kacsuk. *Workflows in a dashboard: a new generation of usability.* In Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science, pages 82-93. IEEE Press, 2014.

3. R. Filgueira, A. Krause, M. Atkinson, I. Klampanos, **A. Spinuso**, and S. Sanchez-Exposito. dispel4py: *An agile framework for data-intensive escience*. In 11th IEEE International Conference on e-Science, pages 454-464. IEEE, 2015.

4. M. Atkinson, M. Carpené, E. Casarotti, S. Claus, R. Filgueira, A. Frank, M. Galea, T. Garth, A. Gemü nd, H. Igel, I. Klampanos, A. Krause, L. Krischer, S. H. Leong, F. Magnoni, J. Matser, A. Michelini, A. Rietbrock, H. Schwichtenberg, **A. Spinuso**, and J. P. Vilotte. *VERCE delivers a productive e-science environment for seismology research.* In 2015 IEEE 11th International Conference on e-Science, pages 224-236, Aug 2015.

5. T. Garth, A. Rietbrock, S. Hicks, A. Fuenzalida Velasco, E. Casarotti, and **A. Spinuso**. *Full waveform modelling using the VERCE platform-application to aftershock seismicity in the chile subduction zone*. In EGU General Assembly, Conference Abstracts, volume 17, 2015.

6. **A. Spinuso**, R. Filgueira, M. Atkinson, and A. Gemünd. *Visualisation methods for large provenance collections in data-intensive collaborative platforms.* In EGU General Assembly Conference Abstracts, volume 18, 2016.

7. T. Kiss, P. Kacsuk, R. Lovas, A. Balaskó, **A. Spinuso**, M. Atkinson, D. D'Agostino, E. Danovaro, and M. Schiffers.*WS-PGRADE/gUSE in European projects*. In Kacsuk [162], pages 235-254.

8. A. Mihajlovski, **A. Spinuso**, M. Plieger, and W. Som de Cerff. *Enabling datadriven provenance in NetCDF, via OGC WPS operations. climate analysis services use case*. In AGU Fall Meeting Abstracts, 2016.

Appendix A

S-PROV Classes and Properties

In this appendix we describe in tabular form the classes and properties of the S-PROV model that are referred throughout the thesis. S-PROV is also available as an ontology in OWL[1] format for its experimental adoption[2].

S-PROV: Description of Entities, Agents and Activities	
Class	**Description**
WFExecutionBundle	A *prov:Bundle* providing a complete view on all the entities, agents, activities involved in a specific *WFExecution* and their relationships.
WFDataTraceBundle	A *prov:Bundle* presenting all the entities, agents, activities (and their relationships) involved in the generation of a specific *Data* element.
WFExecution	A *provone:Execution* describing a workflow run. From this activity qualified associations to agents such as users and logical components are established. The class links also to high-level details about the locations where the workflow was deployed (*CResource*).

[1] https://www.w3.org/OWL/

[2] https://github.com/aspinuso/s-provenance/blob/master/resources/s-prov-o.owl

S-PROV: Description of Entities, Agents and Activities	
Class	**Description**
WFExecutionInputs	A *prov:Collection* of entities used for the general parametrisation of a *WFExecution*. The collection contains entities of different kinds; *e.g.* data and configuration files, archive files containing the workflow implementation scripts, references to other workflow executions, etc. The latter indicates the exploitation of information produced by other workflows' runs, allowing to represent, at a coarse-grain detail, inter-workflow information exchange and reuse.
Component	A *prov:Agent* responsible for performing a task in the workflow. It represents an abstract structural element of the workflow and delegates to its instances the actual concurrent computation on the incoming data stream. *Component* agents cooperate in a *WFExecution* and are further described by their `ProvenanceType`, see Chapter 4.
Implementation	This is a *provone:Program* representing an atomic operation constituting a workflow component. It is described by its source code or by a link or to an external versioning system. As the *provone:Program* it has input and outport ports and allows for the traceability of its evolution.
ComponentInstance	A *prov:SoftwareAgent* that acts on behalf of a logical *Component*. In a parallel execution multiple *ComponentInstance* act on behalf of the same *Component* by performing the same *Implementation*.
Invocation	The *prov:Activity* performed by a *ComponentInstance* over incoming data. Each *Invocation* can produce more *Data*. It uses a collection of *ComponentParameters*.
ComponentParameters	Each *ComponentInstance* is configured with a set of initial parameters that are used by their invocations. This class is thereby an *prov:Entity* that is described by the parameters' terms and their values. Values can be literals or referring to other entities. Reconfigurations may occur at runtime and propagated to all the instances.

S-PROV: Description of Entities, Agents and Activities	
Class	**Description**
StateCollection	The internal state of a *ComponentInstance*. A *prov:Collection* that contains references to *Data* entities that get updated by an *Invocation* during stateful operations.
StreamOut	A *prov:Collection* used to characterise a set of *Data* elements, written by a streaming operator to a *provone:Port*, that have the same derivations.
Data	We extend the *prov:Collection* considering as such each output produced or used by an *Invocation*. S-PROV treats data entities as characterised by a generic container of one or more *DataGranule*. This guarantees separation between a baseline of generic information, such as location, format, size and annotations and domain specific metadata. The latter are instead attributed to the granules.
DataGranule	A *DataGranule* is part of a *Data* collection. It extends a *provone:Data* and contains exclusively contextual metadata related to the specific research topic and domain. Metadata can be associated with community standards, as well as defined by the users. The latter fosters the adoption of provenance for the rapid cataloguing and exploration of experimental properties that are extracted from intermediate results.
SystemProcess	The location of a *ComponentInstance* in a runtime environment. It describes information about the computational worker node of a cluster and the operating system process *pid*. Multiple *ComponentInstance* can be allocated to the same worker and also to the same process.
CResource	The computational resource where the workflow or its programs are submitted and executed.

Table A.1: Overview and description of the provenance entities and activities defined by the S-PROV model.

In the table below we describe the most relevant properties of the S-PROV model grouped by their domain classes.

S-PROV: Properties and their Domain Classes		
Domain Class	**Property**	**Description**
Data	*wcount*	The production sequence number of the output data attributed to an instance.
	toStateCollection	This relationship indicates that a reference to the *Data* has been included or updated in the *StateCollection*.
	parameterName	A name can be associated with a *Data* entity, especially when included within the *WorkflowInputParameters*.
	location	The current location of the *Data* entity,
	immediate-access	A boolean tag that indicates that the data is transferred or needs to be transferred or copied as soon as possible to a target destination.
	first-known-destination	The URL that states the first known destination of a copy of the data. Useful in combination of the *immediate-access* property to trigger and record runtime transfer operations.
	size	The size of the data or an estimate.
	format	The format of the data, typically its *mime-type*.
	annotations	User annotations expressed according to an annotation model; *e.g.* Open Annotation[3].
provone:Workflow	*workflowId*	System *id* of the workflow attributed to a workflow; *e.g.* a code attributed by a workflow repository.
	workflowName	Common name of the *Workflow* plan.

[3] `http://www.openannotation.org/spec/core/`

S-PROV: Properties and their Domain Classes		
Domain Class	**Property**	**Description**
Component	*prov-cluster*	Components can be grouped in to clusters. Clusters refer to concepts belonging to external ontologies or defined by the user. A cluster can be associated with a *ProvenanceSensor* for runtime provenance analysis, as we introduced in Section 4.8. It correspond to the *prov:type* attribute.
	CName	The name of the component as it appears in the logical specification of the workflow.
	type	Runtime type of the *Component*. Multiple types can indicate the computational class and the provenance pattern or metadata schema associated with the Component, see Chapter 4.
ComponentInstance	*wasChangedWith*	This relationship indicates that a *ComponentInstance* changed its *ComponentParameters*, *Implementation* or *SystemProcess*.
	qualifiedChange	Qualifies the *wasChangedWith* allowing the introduction of additional information, such as its timestamp.
Implementation	*functionName*	Name of the function of a component's *Implementation*.
	source	Source code of an *Implementation* pointing to an external repository, such as a code revision service; *e.g.* a GitHub repository.
Invocation	*invocationIndex*	Index or sequence number of the current *Invocation*.
	message	Any text message or code generated at runtime during an *Invocation*. A message should be typically used to notify anomalies and errors.

S-PROV: Properties and their Domain Classes		
Domain Class *StateDerivation*, *FlowDerivation*	**Property** *stateLookupTerm*	**Description** A key associated with a *Data* entity referenced in the *StateCollection*, thus describing a *StateDerivation*.
	provone:Port	The input ports where the data was ingested in a *FlowDerivation*.
	hadInformer	The invocation that informed the activity involved in the current derivation.
CResource	*resourceName*	Name of a local or institutional computational infrastructure; *e.g. SCAI_Drachen* or *LRZ_SMUC*.
	modules	Collection of software modules used within the current execution environment; *e.g.* python modules with their version, *e.g.* `matplotlib==1.4.3`.
	resourceType	Type of the computational resource access point (if applicable); *e.g.* GT5, Unicore, OCCI.
	resourceUrl	URL of the remote resource where the workflow has been submitted to.
	queue	Refers to the queue of a cluster for the scheduling of the workflow job; *e.g.* `jobmanager-pbs`.
	mapping	Type of resource mapping for the concrete execution of the workflow; *e.g.* MPI, multiprocessing, Storm, *etc.*.
SystemProcess	*pid*	Operating System's process identifier.
	worker	Name or identifier of the worker node.
	atResource	Identifier of the hosting computational resource.

Table A.2: Overview and description of the provenance properties defined by the S-PROV model.

Appendix B

Queries Examples

In this appendix we collect query algorithms, expressed in Python-style pseudocode, that are referred from Chapter 5. In Table B.1 we provide an overview of the current methods offered by the RESTful API of the S-ProvFlow system.

Listing B.1: Function that produces a view on the activity of the workflow at different levels of granularity. From the provenance clusters, to the workflow's components, their distributed instances and invocations. Prefixes are omitted for readability.

```
#Extract and group information about the workflow's activity based
#on the requested "level" of detail.
#mongodb method: collection.aggregate([match,unwind,group,sort,skip,limit])

def showActivities(self, runid,level,start,limit):

 obj = self.lineage.aggregate([{"$match":{"WFExecution.@id":runid}},
               {"$unwind":"$StreamOut"},
               {"$group":{"_id":"$"+level,
                 "lastEventTime":{"$max":"$generatedAtTime"},
                 "message":{"$push":"$message"},
                 "worker":{"$first":"$worker"},
                 "generatedWithImmediateAccess":{"$push":"$StreamOut.immediateAccess"},
                 "generatedWithLocation":{"$push":"$StreamOut.location"},
                 "qualifiedChange": {"$push":"$qualifiedChange"},
                 "dataCount":{"$push":"$streams.id"}}},
                {"$sort":{"lastEventTime":-1}},
                {"$skip":start},
                {"$limit":limit}])

 return  formatActivity(obj))
```

Listing B.2: JSON-LD document returned by an invocation of the `worflklowexecution.showactivities` method of the `S-ProvFlow` API. The method has been queried to show the information grouped by components' instances. The example includes also the reporting of changes occurred during a specific invocations.


```
{
  "@context": {...},
  "@graph": [
     {
    "s-prov:lastEventTime": "2018-02-10 18:37:54.029421",
    "s-prov:worker": "orfeus-as",
    "s-prov:message": "",
    "s-prov:generatedWithLocation": false,
    "prov:actedOnBehalfOf": {
      "@id": "CorrCoef44",
      "@type": "s-prov:Component"
    },
    "s-prov:generatedWithImmediateAccess": false,
    "s-prov:dataCount": 570,
    "@id": "CorrCoef-Instance--orfeus-as-5399-80b5f751-0e91-11e8-9f7f-f45c89acf865"},
  {"s-prov:lastEventTime": "2018-02-10 18:37:52.774714",
    "s-prov:worker": "orfeus-as",
    "s-prov:message": "Parameter changed",
    "s-prov:generatedWithLocation": false,
    "s-prov:qualifiedChange": [{
      "s-prov:ComponentParameters" : {
          "sampling_rate" : 200,
          "batchsize" : 5},
      "prov:atTime" : "2018-02-10 18:37:52.473455",
      "s-prov:Invocaton" : {
      "@id" : "Source173_write_orfeus-as-92819-7ced2f33-962d-11e7-9ac0-f45c89acf865"}
          ,
      "@type" : "s-prov:Change"}],
        "prov:actedOnBehalfOf": {
        "@id": "Source51",
        "@type": "s-prov:Component"
      },
    "s-prov:generatedWithImmediateAccess": false,
    "s-prov:dataCount": 3,
    "@id": "Source-Instance--orfeus-as-5399-80b5e8de-0e91-11e8-bea2-f45c89acf865"},
    ..]}
```

Listing B.3: Interactive data derivations graph traversal algorithm performed on the lineage collection. From any node, users can set the depth of navigation for tuning the size of the graph to be visualised. The function returns a JSON-LD document representing in a nested structure the data derivation graph (Listing B.4). Prefixes are omitted for readability.

```
1
2  def getTrace(dataids,level) {
3
4   #Extract documents from the lineage collection according to the input dataids
5   #with the required projection ({"field":1}) on the retrieved lineage documents.
6   #mongodb method:  collection.find(query, projection)
7   lindocs = lineage.find({"StreamOut.@id":{"$in":dataids}},
8   { "StreamOut":1
9     "Invocation.@id":1,
10    "WFExecution.@id":1,
11    "ComponentInstance.actedOnBehalfOf":1,
12    "Derivation":1});
13
14  #Navigates recursively through the derivations
15  #(Breath-first) until the the desired depth (level)
16  for data in lindocs:
17        if level > 0:
18                data_dependencies = getTrace(getIds(data["Derivation"]),level-1)
19
20        #Format the payload and add to the results
21        result.add(formatTrace(data,data_dependencies))
22
23  return result}}
```

Listing B.4: JSON-LD document returned by the `data.wasDerivedFrom` method of the `S-ProvFlow` API. Nested information about the data dependencies at line 23 and 27 are omitted for brevity.

```
1  {
2    "s-prov:Data": {
3          "provone:Port": "output",
4          "s-prov:size": 145,
5          "@id": "58346-490c9sij-994b"
6          "prov:hadMember": [
7              {   "@type": "s-prov:DataGranule",
8                  "ro": -0.9970170978,
9                  "iteration": 3,
10                 "vars": "3_2"
11             }],
12         "prov:wasGeneratedBy": {
13             "s-prov:Invocation": { "@id": "CorrCoef-8a73"}
14             "s-prov:WFExecution": {"@id": "CORR_orfeus-as-4837"}
15             }
16         },
17         "prov:wasAttributedTo": {
18             "s-prov:Component":{@id": "CorrCoef2"}
19         },
20         "prov:Derivation": [
21             {   "@type":"s-prov:FlowDerivation",
22                 "s-prov:Data": { "@id": "4888-49086254-994b", ..},
23                 "provone:Port": "input_3"
24             },
25             {   "@type":"s-prov:StateDerivation",
26                 "s-prov:Data": { "@id": "4898-490ba5cf-994b", ..},
27                 "s-prov:lookupterm: "var_49_1"
28             }
29         ],
30     "@context": {.. }
31 }
```

Listing B.5: Query on domain metadata contained into lineage documents and represented as dictionaries of key,value terms. The example refers to an earthquake simulation workflow. In the specific query we want to return all the *WFExecution* (and associated bundle document) of the user `rafiq` that produced data where 5 < `magnitude` < 6 and using `stations` = AQU.

```
#Finds the ids of the WFExecutions
#mongodb method: collection.aggregate([match,group])
  runIds=lineage.aggregate([{
  "User.@id": {
    "$in": ["rafiq"]
  },
  "$or": [
    {"StreamOut": {
        "$elemMatch": {
          "indexedMeta": {
            "$elemMatch": {
              "val": {
                "$lte": 6,
                "$gte": 5
              },
              "key": "magnitude"
            }}}},
    {"StreamOut": {
        "$elemMatch": {
          "indexedMeta": {
            "$elemMatch": {
              "val": "AQU",
              "key": "station"
            }}}},
  ]
},
{"$group":{ "id":"$WFExecution.@id"}}])

#Extract all the documents whose ids are in runIds.
#The projection will include
#Components, Implementations, Location of execution etc.
#method use: collection.find(query, projection)

results = workflow.find({"id": {"$in":runIds} }, { "atLocation" : 1, .. }))
```

Listing B.6: Query sequence that finds workflows executions performed by three users that have processes data from stations CAFR and CERA. The result set is then used to create the image of Figure 5.5 interactively.

```
1
2  #Extract the ids of the workflow executions attributed to three users
3  #from the lineage documents, with data granules' properties
4  #matching the values-ranges (equality match in this case)
5  #mongodb method: collection.aggregate([match,unwind,group,match,group]
6  run_and_users = db.lineage.aggregate(
7  [
8    {"$match": {
9        "User.@id": {
10         "$in": ["aspinuso", "skhan","fmagnoni"]
11       },
12       "$or": [
13         {"StreamOut": {
14             "$elemMatch": {
15               "indexedMeta": {
16                 "$elemMatch": { "val": "CERA","key": "station"}
17                 }}}},
18         {"StreamOut": {
19             "$elemMatch": {
20               "indexedMeta": {
21                 "$elemMatch": { "val": "CAFR","key": "station"}
22                 }}}}}}},
23   {"$unwind": "$StreamOut.indexedMeta"},
24   {"$group": {
25       "_id": {"username": "$User.@id", "runId": "$WFExecution@id"},
26       "indexedMeta": {"$addToSet": "$StreamOut.indexedMeta"}
27       }},
28   {"$match": {
29       "$and": [{
30           "indexedMeta": {
31             "$elemMatch": { "val": "CERA","key": "station"}
32             }},
33         {"indexedMeta": {
34             "$elemMatch": {"val": "CAFR","key": "station"}
35             }}]}},
36  {"$group":{"WFExecution.@id":"$WFExecution.@id","User.@id":"$User.@id"}}])
37
38 #Extract the ids of the connected workflows based on the run_and_users resultset
39 for item in run_and_users:
40   connections = workflow.aggregate([{"$match":{"WFExecution.@id":item["WFExecution.@id"]}},
41                       {"$unwind":"$WFExecutionInput"},
42                       {"$match":{"WFExecutionInput.prov:type":"wfrun"}},
43                       {"$project":{"iWFExecutionInput.url":1,"User.@id":1}}])
```

Provenance API: A collection of web-based methods to store and access provenance information in S-PROV	
Provenance acquisition	
(1) workflowexecutions/insert	Bulk insert of bundle or lineage documents in JSON format.
(2) workflowexecutions/<*id*>/edit	Update of the description of a workflow execution. Users can improve this information in free-tex.
(3) workflowexecutions/<*id*>/delete	Delete a workflow execution trace, including its bundle and all its lineage documents.
Monitoring, validation and lineage queries	
(4) workflowexecutions*(/<id> | ?<query_string>)*	Extract documents from the bundle collection by the `id` of a *WFExecution* or according to a query string which may include `usernames`, `type` of the workflow, the components the run `wasAssociatedWith` and their `implementations`. Data results' metadata and parameters can also be queried by specifying the `terms` and their `min` and `max` values-ranges and data `formats`. Mode of the search can also be indicated (`mode` ::= (`OR` | `AND`). It will apply to the search upon metadata and parameters values of each run.

Provenance API: A collection of web-based methods to store and access provenance information in S-PROV	
(5) workflowexecutions/<***id***>**/showactivity***?*<***query_string***>	Extract detailed information related to the activity related to a *WFExecution* (`id`). The result-set can be grouped by invocations, instances or components (parameter `level`) and shows progress, anomalies (such as exceptions or systems' and users *messages*), occurrence of changes and the rapid availability of accessible data bearing intermediate results. This method can also be used for runtime monitoring.
(6) instances/<***id***> **(7) invocations/**<***id***> **(8) components/**<***id***>	Extract details about a single invocation instance or component by specifying their `id`. The returning document will indicate the changes that occurred, reporting the instances and the first invocation affected.
(9) data*(/*<***id***>$\mid$***?*<***query_string***>*)*	Extract *Data* and their *DataGranules*. The data is selected by specifying its `id` or a *query_string*. Query parameters allow to search by *attribution* to a component or to an implementation, *generation* by a workflow execution and by combining more metadata terms with their `min` and `max` values-ranges. Mode of the search can also be indicated (`mode` ::= (`OR` $\mid$ `AND`). It will apply to the search upon metadata and parameters values-ranges.

Provenance API: A collection of web-based methods to store and access provenance information in S-PROV	
(10) data/<*id*>/filterByAncestor*?*<*query_string*>	Filter a list of data `ids` based on the existence of at least one ancestor in their data dependency graph, according to a list of metadata terms and their `min` and `max` values-ranges. Maximum depth `level` and `mode` of the search can also be indicated (`mode` ::= (`OR` | `AND`).
(11) data/<*id*>/derivedData **(12) data/<*id*>/wasDerivedFrom**	Starting from a specific data entity of the data dependency is possible to navigate through the derived data (11) or backwards across the element's data dependencies (12). The number of traversal steps is provided as a parameter (`level`).
(13) terms*?*<*query_string*>	Return a list of discoverable metadata terms based on their appearance for a list of `runIds`, `usernames`, or for the whole provenance archive. Terms are returned indicating their type (when consistently used), min and max values and their number occurrences within the scope of the search.

Provenance API: A collection of web-based methods to store and access provenance information in S-PROV	
Comprehensive Summaries	
(14) summaries/workflowexecutions/<*id*>*?*<*query_string*>	Produce a detailed overview of the distribution of the computation, reporting the size of data movements between the workflow components, their instances or invocations across worker nodes, depending on the specified granularity `level`. Additional information, such as process `pid`, `worker`, instance or component of the workflow (depending on the level of granularity) can be selectively extracted by assigning these properties to a `groupBy` parameter. This will support the generation of grouped views, as shown in Figure 5.11.
(15) summaries/collaborative*?*<*query_string*>	Extract information about the reuse and exchange of data between workflow executions based on terms' values-ranges and a group of users. The API method allows for inclusive or exclusive (`mode` ::= (`OR` | `AND`) queries on the terms' values. As above, additional details, such as running infrastructure, type and name of the workflow can be selectively extracted by assigning these properties to a `groupBy` parameter. This will support the generation of grouped views, as shown in Figure 5.12.

Provenance API: A collection of web-based methods to store and access provenance information in S-PROV	
Export Methods to W3C PROV Formats	
(16) workflowexecutions/<*id*>/export **(17) data/<*id*>/export**	Export of provenance information PROV-XML or RDF format. The S-PROV information returned covers the whole workflow execution (16) or is restricted to a single data element (17). In the latter case, the graph is returned by following the derivations within and across runs. A *level* parameter allows to indicate the depth of the resulting trace.

Table B.1: S-ProvFlow API Methods

Bibliography

[1] 3D Xpoint Technology. https://www.micron.com/products/advanced-solutions/3d-xpoint-technology. Accessed: January 2018.

[2] 3rd National eScience Symposium. https://www.esciencecenter.nl/symposium201. Accessed: February 2018.

[3] A library for W3C Provenance Data. https://github.com/trungdong/prov. Accessed: February 2018.

[4] Apache Spark. http://spark.apache.org/. Accessed: February 2018.

[5] Apache Storm. https://storm.apache.org/. Accessed: December 2017.

[6] Arbor.js graph visualization library. http://arborjs.org/. Accessed: February 2018.

[7] Associating metadata in documents with graph provenance. http://patterns.promsns.org/pattern/12.

[8] CERFACS Institute. https://cerfacs.fr/en/. Accessed: January 2018.

[9] Challenges in irreproducible research. http://www.nature.com/news/reproducibility-1.17552. Accessed: January 2018.

[10] CLIPC project. http://www.clipc.eu. Accessed: February 2018.

[11] CMIP5 - Coupled Model Intercomparison. https://cmip.llnl.gov/cmip5. Accessed: February 2018.

[12] Committee on Data of the International Council for Science. http://www.codata.org. Accessed: February 2018.

[13] Copernicus Programme. https://climate.copernicus.eu. Accessed: February 2018.

[14] D3 Data-Driven Documents. https://d3js.org/. Accessed: February 2018.

[15] DARE Project. http://project-dare.eu/. Accessed: February 2018.

[16] DataONE Project. https://www.dataone.org. Accessed: January 2018.

[17] dispel4py stream-based workflow library. https://github.com/dispel4py/dispel4py. Accessed: January 2018.

[18] Docker. https://www.docker.com/. Accessed: February 2018.

[19] Earth System Grid Federation. https://esgf.llnl.gov/. Accessed: February 2018.

[20] EGI Applications Database. https://appdb.egi.eu/. Accessed: February 2018.

[21] EOSC Pilot Project. https://eoscpilot.eu. Accessed: January 2018.

[22] EPOS Training Pages. https://www.epos-ip.org/events/epos-seismology-workshop-lisbon-portugal-25-27-october-2017. Accessed: February 2018.

[23] EUDAT B3STAGE Tool. https://www.eudat.eu/b2stage. Accessed: February 2018.

[24] Exareme elastic dataflow processing. http://madgik.github.io/exareme/. Accessed: February 2018.

[25] FDSN, Federation of Digital Seismograph Networks. http://www.fdsn.org/. Accessed: February 2018.

[26] GridFTP: Users Guide. http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/user/index.html. Accessed: February 2018.

[27] Human Brain Project (HBP). https://www.humanbrainproject.eu. Accessed: December 2017.

[28] Human Brain Project Report. https://www.kcl.ac.uk/sspp/departments/sshm/research/Research-Groups/BIOS/BIOS-Projects/HBP/Report-Workshop-Fondation-Brocher-2015---Final.pdf. Accessed: January 2018.

[29] INGV Institute. http://www.ingv.it/it/. Accessed: January 2018.

[30] IRIS Data Center. http://ds.iris.edu/ds/nodes/dmc/. Accessed: February 2018.

[31] iRODS Data Management Software. https://irods.org/. Accessed: February 2018.

[32] IS-ENES Climate4Impact . https://climate4impact.eu. Accessed: February 2018.

[33] JSON for Linking Data. https://json-ld.org/. Accessed: February 2018.

[34] MAGIC Project. https://climate.copernicus.eu/development-c3s-software-data-analysis-climate-models. Accessed: February 2018.

[35] MongoDB Document-oriented Data Base. http://mongodb.org. Accessed: February 2018.

[36] Mongoosastic project. https://github.com/mongoosastic/mongoosastic. Accessed: February 2018.

[37] MPI, Message Passing Interface. https://computing.llnl.gov/tutorials/mpi/. Accessed: February 2018.

[38] MyExperiment. http://myexperiment.org. Accessed: February 2018.

[39] Neo4j and MongoDB integration. https://neo4j.com/developer/mongodb/.

[40] Neo4j Graph Database. http://neo4j.org. Accessed: February 2018.

[41] NetCDF Attribute Convention for Data Discovery. http://wiki.esipfed.org/index.php/Attribute_Convention_for_Data_Discovery_1-3. Accessed: February 2018.

[42] NetCDF, Network Common Data Form. http://www.unidata.ucar.edu/software/netcdf/. Accessed: April 2018.

[43] ObsPy a Python framework for seismology. https://www.obspy.org/. Accessed: February 2018.

[44] Open Geospacial Consortium. http://www.opengeospatial.org/. Accessed: February 2018.

[45] PASS: Provenance-Aware Storage Systems. http://www.eecs.harvard.edu/syrah/pass/. Accessed: February 2018.

[46] Processing 2 Billion Documents A Day. https://www.mongodb.com/blog/post/processing-2-billion-documents-a-day-and-30tb-a. Accessed: February 2018.

[47] Project Jupyter Notebook. http://jupyter.org/. Accessed: February 2018.

[48] PROV-AQ: Provenance Access and Query. https://www.w3.org/TR/prov-aq. Accessed: February 2018.

[49] PROV-N, The Provenance Notation. https://www.w3.org/TR/prov-n/. Accessed: February 2018.

[50] PROV-O, The PROV Ontology. https://www.w3.org/TR/prov-o/. Accessed: January 2018.

[51] PROV-Template: A Quick Start. https://lucmoreau.wordpress.com/2017/03/30/prov-template-a-quick-start/. Accessed: February 2018.

[52] ProvONE Data Model. https://purl.dataone.org/provone-v1-dev. Accessed: January 2018.

[53] ProvStore, provenance storage and distribution. https://provenance.ecs.soton.ac.uk/store/. Accessed: February 2018.

[54] Pyflex Python seismic analysis tool. http://krischer.github.io/pyflex. Accessed: February 2018.

[55] Python metaprogramming. https://docs.python.org/2/library/functions.html#type. Accessed: February 2018.

[56] Python Process-based threading interface. https://docs.python.org/2/library/multiprocessing.html. Accessed: February 2018.

[57] QA4Seas Project . https://climate.copernicus.eu/quality-assurance-multi-model-seasonal-forecast-products. Accessed: February 2018.

[58] RDF. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/Overview.html. Accessed: June 2018.

[59] RDF Schema. https://www.w3.org/TR/rdf-schema/. Accessed: February 2018.

[60] Research Data Alliance, Data Fabric Interest Group. https://rd-alliance.org/group/data-fabric-ig.html. Accessed: January 2018.

[61] Research Data Alliance, Data Provenance Interest Group. https://rd-alliance.org/groups/research-data-provenance.html. Accessed: January 2018.

[62] Research Data Alliance, Information Types Working Group. https://www.rd-alliance.org/groups/pid-information-types-wg.html. Accessed: February 2018.

[63] Research Data Alliance, Provenance Patterns Working Group. https://www.rd-alliance.org/groups/provenance-patterns-wg. Accessed: January 2018.

[64] Research Data Alliance (RDA). https://rd-alliance.org. Accessed: January 2018.

[65] SCAI-Fraunhofer, High Performance Computing. https://www.scai.fraunhofer.de/en/business-research-areas/high-performance-computing.html. Accessed: February 2018.

[66] SEIS-PROV: Provenance for Seismological Data. http://seismicdata.github.io/SEIS-PROV/. Accessed: February 2018.

[67] Semagrow federated SPARQL query processor. http://semagrow.github.io. Accessed: February 2018.

[68] Sencha Ext JS. https://www.sencha.com/products/extjs/. Accessed: February 2018.

[69] SPECFEM3D Cartesian. https://geodynamics.org/cig/software/specfem3d/. Accessed: February 2018.

[70] SPECFEM3D Globe. https://geodynamics.org/cig/software/specfem3d_globe/. Accessed: February 2018.

[71] SuperMUC Petascale System. https://www.lrz.de/services/compute/supermuc/. Accessed: February 2018.

[72] Taverna Provenance Management. http://www.taverna.org.uk/documentation/taverna-2-x/provenance/. Accessed: February 2018.

[73] Terse RDF Triple Language. https://www.w3.org/TR/turtle/. Accessed: February 2018.

[74] The DOI System. http://www.doi.org. Accessed: February 2018.

[75] The MongoDB 3.6 Manual. https://docs.mongodb.com/manual/. Accessed: February 2018.

[76] The MongoDB 3.6 Manual, compound indexes. https://docs.mongodb.com/manual/core/index-compound/. Accessed: February 2018.

[77] The National Science Foundation. http://www.nsf.gov/. Accessed: February 2018.

[78] The Virtual Research Environment for Seismology Portal. http://portal.verce.eu. Accessed: February 2018.

[79] UML Class Diagrams. https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/. Accessed: February 2018.

[80] VERCE Project. http://www.verce.eu. Accessed: January 2018.

[81] VERCE Trainings Pages. http://verce.eu/Training/Training2015.php. Accessed: February 2018.

[82] VisTrails open-source scientific workflow. http://www.vistrails.org/. Accessed: February 2018.

[83] W3C PROV Data Model. http://www.w3.org/TR/prov-dm/. Accessed: January 2018.

[84] Web Processing Service. http://www.opengeospatial.org/standards/wps. Accessed: February 2018.

[85] WINGS semantic workflow system. http://www.wings-workflows.org/. Accessed: February 2018.

[86] Workflow4Ever Project. http://www.wf4ever-project.org/. Accessed: January 2018.

[87] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, et al. The design of the borealis stream processing engine. In *Second Biennial Conference on Innovative Data Systems Research*, volume 5, pages 277–289, 2005.

[88] G. Agha. Concurrent object-oriented programming. *Communications of the ACM*, 33(9):125–141, 1990. ACM.

[89] G. A. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. *Journal of Functional Programming*, 7(1):1–72, 1997.

[90] G. Alonso and C. Mohan. *Workflow Management: The Next Generation of Distributed Processing Tools*, pages 35–59. Springer US, Boston, MA, 1997.

[91] P. Alper, K. Belhajjame, and C. A. Goble. Static analysis of taverna workflows to predict provenance patterns. *Future Generation Computer Systems*, 75:310 – 329, 2017.

[92] P. Alper and C. A. Goble. Automatic vs Manual Provenance Abstractions : Mind the Gap. *TaPP'16*, 2016.

[93] I. Altintas, O. Barney, and E. Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In L. Moreau and I. Foster, editors, *Provenance and Annotation of Data*, pages 118–132, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[94] M. Anand and S. Bowers. Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. *Scientific and Statistical Database Management*, pages 237–254, 2009.

[95] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear road: a stream data management benchmark. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 480–491. VLDB Endowment, 2004.

[96] M. Atkinson. Data-Intensive thinking with Dispel. In *[98]*, chapter 4, pages 61–122. Wiley, 2013.

[97] M. Atkinson, M. Carpen, E. Casarotti, S. Claus, R. Filgueira, A. Frank, M. Galea, T. Garth, A. Gemnd, H. Igel, I. Klampanos, A. Krause, L. Krischer, S. H. Leong, F. Magnoni, J. Matser, A. Michelini, A. Rietbrock, H. Schwichtenberg, A. Spinuso, and J. P. Vilotte. VERCE delivers a productive e-science environment for seismology research. In *2015 IEEE 11th International Conference on e-Science*, pages 224–236, Aug 2015.

[98] M. P. Atkinson, R. Baxter, P. Brezany, O. Corcho, M. Galea, M. Parsons, D. Snelling, and J. van Hemert. *The Data Bonanza: Improving Knowledge Discovery in Science, Engineering, and Business*. Wiley-IEEE Computer Society Pr, 1st edition, 2013.

[99] M. Baker et al. Is there a reproducibility crisis? *Nature*, 533(7604):452–454, 2016.

[100] S. Bechhofer, D. De Roure, M. Gamble, C. Goble, and I. Buchan. Research Objects: Towards Exchange and Reuse of Digital Knowledge. *Nature Precedings*, (ERIM Project Document erim1rep091103ab12), 2010.

[101] K. Belhajjame and *etal.* A Suite of Ontologies for Preserving Workflow-Centric Research Objects. *J. Web Semantics*, in press 2015.

[102] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J. M. Gómez-Pérez, S. Bechhofer, et al. Using a suite of ontologies for preserving workflow-centric research objects. *Web Semantics: Science, Services and Agents on the World Wide Web*, 32:16–42, 2015. Elsevier.

[103] G. D. Bensen, M. H. Ritzwoller, M. P. Barmin, A. L. Levshin, F. Lin, M. P. Moschetti, N. M. Shapiro, and Y. Yang. Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements. *Geophysical Journal International*, 169(3):1239–1260, 2007.

[104] M. Beyreuther and *etal.* ObsPy: A Python Toolbox for Seismology. *Seismological Research Letters*, 81(3):530–533, 2010.

[105] E. R. Boose, A. M. Ellison, L. J. Osterweil, L. A. Clarke, R. Podorozhny, J. L. Hadley, A. Wise, and D. R. Foster. Ensuring reliable datasets for environmental models and forecasts. *Ecological Informatics*, 2(3 SPEC. ISS.):237–247, 2007.

[106] M. A. Borkin, C. S. Yeh, M. Boyd, P. MacKo, K. Z. Gajos, M. Seltzer, and H. Pfister. Evaluation of filesystem provenance visualization tools. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2476–2485, 2013.

[107] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, and O. Pastor, editors, *Conceptual Modeling – ER 2005*, pages 369–384, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[108] S. Bowers, T. McPhillips, B. Ludäscher, S. Cohen, and S. B. Davidson. A Model for User-Oriented Data Provenance in Pipelined Scientific Workflows. *Lecture Notes in Computer Science*, 4145(4145):133–147, 2006.

[109] A. Brito, A. Martin, T. Knauth, S. Creutz, D. Becker, S. Weigert, and C. Fetzer. Scalable and low-latency data processing with stream mapreduce. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, pages 48–58, Nov 2011.

[110] P. Buneman, A. Chapman, and J. Cheney. Provenance management in curated databases. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 539–550, New York, NY, USA, 2006. ACM.

[111] P. Buneman, J. Cheney, and E. V. Kostylev. Hierarchical models of provenance. In *Proceedings of the 4th USENIX conference on Theory and Practice of Provenance*, TaPP'12, pages 10–10, Berkeley, CA, USA, 2012. USENIX Association.

[112] C. Buontempo. *European Climate Services*, pages 27–40. Springer International Publishing, Cham, 2018.

[113] L. C. Burgess, D. Crotty, D. de Roure, J. Gibbons, C. Goble, P. Missier, R. Mortier, T. E. Nichols, and R. O'Beirne. Alan Turing institute symposium on reproducibility for data-intensive research–final report. 2016.

[114] M. Carpené, I. A. Klampanos, S. H. Leong, E. Casarotti, P. Danecek, G. Ferini, A. Gemünd, A. Krause, L. Krischer, F. Magnoni, M. Simon, A. Spinuso, L. Trani, M. Atkinson, G. Erbacci, A. Frank, H. Igel, A. Rietbrock, H. Schwichtenberg, and J.-P. Vilotte. Towards addressing cpu-intensive seismological applications in europe. In J. M. Kunkel, T. Ludwig, and H. W. Meuer, editors, *Supercomputing*, pages 55–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[115] E. Casarotti, F. Magnoni, D. Komatitsch, D. Melini, A. Michelini, A. Piersanti, C. Tape, and J. Tromp. Towards full-waveform tomography of the Italian lithosphere. In *AGU Fall Meeting Abstracts*, 2015.

[116] R. Castro Fernandez, M. Migliavacca, E. Kalyvianaki, and P. Pietzuch. Integrating scale out and fault tolerance in stream processing using operator state management. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 725–736, New York, NY, USA, 2013. ACM.

[117] F. Costa, D. Oliveira, K. Ocaa, E. Ogasawara, and M. Mattoso. Enabling reexecutions of parallel scientific workflows using runtime provenance data. In P. Groth and J. Frew, editors, *Provenance and Annotation of Data and Processes*, volume 7525 of *Lecture Notes in Computer Science*, pages 229–232. Springer Berlin Heidelberg, 2012.

[118] F. Costa, V. Silva, D. de Oliveira, K. Ocaña, E. Ogasawara, J. Dias, and M. Mattoso. Capturing and Querying Workflow Runtime Provenance with PROV: A Practical Approach. *Proceedings of the Joint {EDBT/ICDT} 2013 Workshops*, pages 282–289, 2013.

[119] D. Crawl and I. Altintas. A provenance-based fault tolerance mechanism for scientific workflows. *Provenance and Annotation of Data and Processes*, pages 152–159, 2008.

[120] T. Crick, B. A. Hall, and S. Ishtiaq. Reproducibility as a technical specification. *arXiv preprint arXiv:1504.01310*, 2015.

[121] V. Cuevas-Vicenttín, P. Kianmajd, B. Ludäscher, P. Missier, F. Chirigati, Y. Wei, D. Koop, and S. Dey. The PBase Scientific Workflow Provenance Repository. *International Journal of Digital Curation*, 9:28–38, 2014.

[122] S. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1–6, 2008.

[123] T. De Nies, S. Coppens, E. Mannens, and R. Van de Walle. Modeling uncertain provenance and provenance of uncertainty in W3C PROV. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 167–168. ACM, 2013.

[124] D. De Oliveira, V. Silva, and M. Mattoso. How Much Domain Data Should Be in Provenance Databases? *Proceedings of the 7th USENIX Conference on Theory and Practice of Provenance*, page 9, 2015.

[125] W. De Pauw, M. LeŢia, B. Gedik, H. Andrade, A. Frenkiel, M. Pfeifer, and D. Sow. Visual debugging for stream processing applications. In *Runtime Verification*, pages 18–35, Berlin, Heidelberg, 2010. Springer.

[126] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. ACM.

[127] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[128] S. Dey, K. Belhajjame, D. Koop, M. Raul, and B. Ludäscher. Linking prospective and retrospective provenance in scripts. In *Proceedings of the 7th USENIX Conference on Theory and Practice of Provenance*, TaPP'15, page 11, Berkeley, CA, USA, 2015. USENIX Association.

[129] G. M. Draper, Y. Livnat, and R. F. Riesenfeld. A Survey of Radial Methods for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):759 – 776, 2009.

[130] M. Duchein. Theoretical principles and practical problems of respect des fonds in archival science. *Archivaria*, 16:64–82, 1983.

[131] Z. Duputel, V. Ferrazzini, F. Brenguier, N. Shapiro, M. Campillo, and A. Nercessian. Real time monitoring of relative velocity changes using ambient seismic noise at the Piton de la Fournaise volcano (la Runion) from January 2006 to June 2007. *Journal of Volcanology and Geothermal Research*, 184(12):164 – 173, 2009. Recent advances on the geodynamics of Piton de la Fournaise volcano.

[132] R. G. Dyson. Strategic development and swot analysis at the university of warwick. *European Journal of Operational Research*, 152(3):631 – 640, 2004. Applications of Soft O.R. Methods.

[133] D. Ferrucci, A. Levas, S. Bagchi, D. Gondek, and E. T. Mueller. Watson: beyond jeopardy! *Artificial Intelligence*, 199:93–105, 2013.

[134] R. Filgueira, A. Krause, M. Atkinson, I. Klampanos, A. Spinuso, and S. Sanchez-Exposito. dispel4py: An agile framework for data-intensive escience. In *11th IEEE International Conference on e-Science*, pages 454–464. IEEE, 2015.

[135] I. Foster. The virtual data grid: a new model and architecture for data-intensive collaboration. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, SSDBM '03, page 11, Washington, DC, USA, 2003. IEEE Computer Society.

[136] M. Fraser. Virtual research environments: overview and activity. *Ariadne*, (44), 2005.

[137] S. W. French and B. Romanowicz. Broad plumes rooted at the base of the earth's mantle beneath major hotspots. *Nature*, 525(7567):95–99, 09 2015. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.

[138] A. Friis-Christensen, A. Perego, C. Tsinaraki, and L. Vaccari. The jrc multidisciplinary research data infrastructure. In *Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services*, pages 338–342. ACM, 2017.

[139] L. M. R. Gadelha, M. Wilde, M. Mattoso, and I. Foster. MTCProv: a practical provenance query framework for many-task scientific computing. *Distributed and Parallel Databases*, 30(5):351–370, Oct 2012. Springer US.

[140] L. M. Gadelha Jr, M. Mattoso, M. Wilde, and I. T. Foster. Provenance query patterns for many-task scientific computing. In *Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP'11)*, 2011.

[141] D. Garijo, Y. Gil, and O. Corcho. Abstract, link, publish, exploit: An end to end framework for workflow sharing. *Future Generation Computer Systems*, 75:271 – 283, 2017. Elsevier.

[142] T. Garth, A. Rietbrock, S. Hicks, A. Fuenzalida Velasco, E. Casarotti, and A. Spinuso. Full waveform modelling using the VERCE platform-application to aftershock seismicity in the chile subduction zone. In *EGU General Assembly Conference Abstracts*, volume 17, 2015.

[143] A. Gehani and D. Tariq. SPADE: Support for provenance auditing in distributed environments. In *Proceedings of the 13th International Middleware Conference*, Middleware '12, pages 101–120, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

[144] R. Gençay, M. Dacorogna, U. A. Muller, O. Pictet, and R. Olsen. *An introduction to high-frequency finance*. Elsevier, 2001.

[145] D. Genius, A. M. Kordon, and K. Z. el Abidine. Space optimal solution for data reordering in streaming applications on NoC based MPSoC. *Journal of Systems Architecture*, 59(7):455 – 467, 2013.

[146] W. Gerlach, W. Tang, K. Keegan, T. Harrison, A. Wilke, J. Bischof, M. D'Souza, S. Devoid, D. Murphy-Olson, N. Desai, and F. Meyer. Skyport: Container-based execution environment management for multi-cloud scientific workflows. In *Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds*, DataCloud '14, pages 25–32, Piscataway, NJ, USA, 2014. IEEE Press.

[147] S. Gesing, M. Atkinson, R. Filgueira, I. Taylor, A. Jones, V. Stankovski, C. S. Liew, A. Spinuso, G. Terstyanszky, and P. Kacsuk. Workflows in a dashboard: a new generation of usability. In *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science*, pages 82–93. IEEE Press, 2014.

[148] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the Challenges of Scientific Workflows. *Computer*, 40(12):24–32, Dec. 2007.

[149] Y. Gil, P. Szekely, S. Villamizar, T. C. Harmon, V. Ratnakar, S. Gupta, M. Muslea, F. Silva, and C. A. Knoblock. Mind your metadata: Exploiting semantics for configuration, adaptation, and provenance in scientific workflows.

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7032 LNCS(PART 2):65–80, 2011.

[150] Google. KML Documentation. http://code.google.com/apis/kml/documentation/. Accessed 8th May 2011.

[151] I. Gorton, P. Greenfield, A. Szalay, and R. Williams. Data-intensive computing in the 21st century. *Computer*, 41(4):30–32, April 2008.

[152] R. Haberfellner and O. Weck. Agile systems engineering versus agile systems engineering. In *INCOSE International Symposium*, volume 15, pages 1449–1465. Wiley Online Library, 2005.

[153] A. J. G. Hey, S. Tansley, and K. M. Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[154] A. J. G. Hey and A. E. Trefethen. The data deluge: An e-science perspective. In F. Berman, G. C. Fox, and A. J. G. Hey, editors, *Grid Computing - Making the Global Infrastructure a Reality*, chapter 36, pages 809–824. Wiley and Sons, 2003.

[155] T. Hey, S. Tansley, and K. M. Tolle. Jim gray on escience: a transformed scientific method. In *The Fourth Paradigm* [153].

[156] R. Hoekstra and P. Groth. Prov-o-viz - understanding the role of activities in provenance. In B. Ludäscher and B. Plale, editors, *Provenance and Annotation of Data and Processes*, pages 215–220, Cham, 2015. Springer International Publishing.

[157] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. 12(5):741–748, 2006. IEEE.

[158] J. Hunter and K. Cheung. Provenance Explorer-a graphical interface for constructing scientific publication packages from provenance trails. *International Journal on Digital Libraries*, 7(1):99–107, 2007. Springer.

[159] T. D. Huynh, D. T. Michaelides, and L. Moreau. *PROV-JSONLD: A JSON and Linked Data Representation for Provenance*, pages 173–177. Springer, Cham, 2016.

[160] T. D. Huynh and L. Moreau. Provstore: a public provenance repository. In *International Provenance and Annotation Workshop*, pages 275–277. Springer, 2014.

[161] M. Interlandi, K. Shah, S. D. Tetali, M. A. Gulzar, S. Yoo, M. Kim, T. Millstein, and T. Condie. Titian: Data provenance support in spark. *Proc. VLDB Endow.*, 9(3):216–227, Nov. 2015.

[162] P. Kacsuk, editor. *Science Gateways for Distributed Computing Infrastructures: Development framework and exploitation by scientific user communities*. Springer, 2014.

[163] G. Kahn. The semantics of a simple language for parallel programming. *In Information Processing*, 74:471–475, 1974.

[164] T. Kiss, P. Kacsuk, R. Lovas, Á. Balaskó, A. Spinuso, M. Atkinson, D. DAgostino, E. Danovaro, and M. Schiffers. WS-PGRADE/gUSE in european projects. In Kacsuk [162], pages 235–254.

[165] M. Kozlovszky, K. Karóczkai, I. Márton, P. Kacsuk, and T. Gottdank. DCI bridge: Executing WS-PGRADE workflows in distributed computing infrastructures. In Kacsuk [162], pages 51–67.

[166] M. I. Krzywinski, J. E. Schein, I. Birol, J. Connors, R. Gascoyne, D. Horsman, S. J. Jones, and M. A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 2009.

[167] S. Lawrence and C. L. Giles. Inquirus, the NECI meta search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):95–105, Apr. 1998.

[168] E. A. Lee and E. Matsikoudis. The semantics of dataflow with firing. *G. Huet, G. Plotkin, J.-J. Lévy, and Y. Bertot, editors, From Semantics to Computer Science: Essays in Honour of Gilles Kahn*, pages 71–94, 2008. Citeseer.

[169] E. A. Lee and D. G. Messerschmitt. Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245, 1987.

[170] E. A. Lee and T. M. Parks. Dataflow process networks. *Proceedings of the IEEE*, 83(5):773–801, 1995. IEEE.

[171] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. OPQL: A First OPM-Level Query Language for Scientific Workflow Provenance. *2011 IEEE International Conference on Services Computing*, pages 136–143, July 2011. Ieee.

[172] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases. *Future Gener. Comput. Syst.*, 27(6):781–789, June 2011.

[173] L. Lin, X. Yu, and N. Koudas. Pollux: towards scalable distributed real-time search on microblogs. *Proceedings of the 16th International Conference on Extending*, 2013.

[174] D. Logothetis, C. Olston, B. Reed, K. C. Webb, and K. Yocum. Stateful bulk processing for incremental analytics. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 51–62, New York, NY, USA, 2010. ACM.

[175] A. Lopez Alos, C. Bergeron, and B. Raoult. Copernicus climate data store toolbox. `https://www.ecmwf.int/sites/default/files/elibrary/2017/17123-copernicus-climate-data-store-toolbox.pdf`, 2017. Accessed: January 2018.

[176] B. Ludaescher, N. Podhorszki, I. Altintas, S. Bowers, and T. McPhillips. From computation models to models of provenance: the RWS approach. *Concurrency and Computation: Practice and Experience*, 20(5):507–518, 2008. Wiley Online Library.

[177] B. Ludaescher, N. Podhorszki, I. Altintas, S. Bowers, and T. McPhillips. From computation models to models of provenance: the RWS approach. *Concurrency and Computation: Practice and Experience*, 20(5):507–518, 2008. Wiley.

[178] B. Ludäscher and *etal.* Scientific workflow management and the Kepler system. *Concurrency and Computation: P&E*, 18(10):1039–1065, 2006. Wiley.

[179] S. Madougou, S. Shahand, M. Santcroos, B. van Schaik, A. Benabdelkader, A. van Kampen, and S. Olabarriaga. Characterizing workflow-based activity on a production e-infrastructure using provenance data. *Future Generation Computer Systems*, 29(8):1931 – 1942, 2013. The fourth IEEE International Conference on e-Science 2011.

[180] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

[181] J. H. Marburger, E. F. Kvamme, G. Scalise, and D. A. Reed. Leadership under challenge: Information technology r&d in a competitive world. an assessment of the federal networking and information technology r&d program. Technical report, Executive Office of The President Washington DC President's Council of Advisors on Science and Technology, 2007.

[182] C. A. Mattmann. Cultivating a research agenda for data science. *Journal Of Big Data*, 1(1):6, Aug 2014. Springer.

[183] M. Mattoso, J. Dias, K. A. Ocaña, E. Ogasawara, F. Costa, F. Horta, V. Silva, and D. De Oliveira. Dynamic steering of HPC scientific workflows: A survey. *Future Generation Computer Systems*, 46:100–113, 2015. Elsevier B.V.

[184] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, Y. Cao, F. Chirigati, S. Dey, J. Freire, D. Huntzinger, C. Jones, D. Koop, P. Missier, M. Schildhauer, C. Schwalm, Y. Wei, J. Cheney, M. Bieda, B. Ludaescher, and B. Lud. YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts. *International Journal of Digital Curation*, 10(1):1–15, 2015.

[185] H. Meng, R. Kommineni, Q. Pham, R. Gardner, T. Malik, and D. Thain. An invariant framework for conducting reproducible computational science. *Jour-*

nal of Computational Science, 9:137 – 142, 2015. Computational Science at the Gates of Nature.

[186] A. Mihajlovski, A. Spinuso, M. Plieger, and W. Som de Cerff. Enabling data-driven provenance in NetCDF, via OGC WPS operations. climate analysis services use case. In *AGU Fall Meeting Abstracts*, 2016.

[187] A. Misra, M. Blount, A. Kementsietsidis, D. M. Sow, and M. Wang. Advances and challenges for scalable provenance in stream processing systems. In J. Freire, D. Koop, and L. Moreau, editors, *IPAW*, volume 5272 of *Lecture Notes in Computer Science*, pages 253–265. Springer, 2008.

[188] P. Missier and B. Ludascher. Linking multiple workflow provenance traces for interoperable collaborative science. *5th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, 2010.

[189] P. Missier, N. W. Paton, and K. Belhajjame. Fine-grained and efficient lineage querying of collection-based workflow provenance. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 299–310, New York, NY, USA, 2010. ACM.

[190] L. Moreau, B. Batlajery, T. D. Huynh, D. Michaelides, and H. Packer. A templating system to generate provenance. *IEEE Transactions on Software Engineering*, PP(99):1–1, 2017.

[191] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson. The open provenance model: An overview. In J. Freire, D. Koop, and L. Moreau, editors, *Provenance and Annotation of Data and Processes*, pages 323–326, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[192] M. R. Munafò, B. A. Nosek, D. V. Bishop, K. S. Button, C. D. Chambers, N. P. du Sert, U. Simonsohn, E.-J. Wagenmakers, J. J. Ware, and J. P. Ioannidis. A manifesto for reproducible science. *Nature Human Behaviour*, 1:0021, 2017. Nature Publishing Group.

[193] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer. Provenance-aware storage systems. In *USENIX Annual Technical Conference,*

General Track, pages 43–56, 2006.

[194] L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire. noWorkflow: Capturing and Analyzing Provenance of Scripts. In B. Ludscher and B. Plale, editors, *Provenance and Annotation of Data and Processes*, volume 8628 of *Lecture Notes in Computer Science*, pages 71–83. Springer International Publishing, 2015.

[195] J. Myers, M. Hedstrom, D. Akmon, S. Payette, B. A. Plale, I. Kouper, S. McCaulay, R. McDonald, I. Suriarachchi, A. Varadharaju, P. Kumar, M. Elag, J. Lee, R. Kooper, and L. Marini. Towards sustainable curation and preservation: The sead project's data services approach. In *2015 IEEE 11th International Conference on e-Science*, pages 485–494, Aug 2015.

[196] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 170–177. IEEE, 2010.

[197] E. Ogasawara, J. Dias, and D. Oliveira. An Algebraic Approach for Data-Centric Scientific Workflows. *Proceedings of the VLDB Endowment (PVLDB)*, 4(12), 2011.

[198] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, et al. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[199] OPeNDAP Inc. OPeNDAP (Open-source Project for a Network Data Access Protocol). http://www.opendap.org. Accessed 1st May 2011.

[200] D. Peter, D. Komatitsch, Y. Luo, R. Martin, N. Le Goff, E. Casarotti, P. Le Loher, F. Magnoni, Q. Liu, C. Blitz, et al. Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes. *Geophysical Journal International*, 186(2):721–739, 2011.

[201] J. F. N. Pimentel, V. Braganholo, L. Murta, and J. Freire. 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, 2015. USENIX Association.

[202] O. V. Precup and G. Iori. Cross-correlation measures in the high-frequency domain. *European Journal of Finance*, 13(4):319–331, 2007.

[203] R. Ranjan. Streaming big data processing in datacenter clouds. *IEEE Cloud Computing*, 1(1):78–83, 2014.

[204] B. Rodriguez-Castro, H. Glaser, and L. Carr. How to reuse a faceted classification and put it on the semantic web. In *International Semantic Web Conference*, pages 663–678. Springer, 2010.

[205] C. Rostoker, A. Wagner, and H. Hoos. A parallel workflow for real-time correlation and clustering of high-frequency stock market Data. *Proceedings - 21st International Parallel and Distributed Processing Symposium, IPDPS 2007; Abstracts and CD-ROM*, 2007.

[206] G. Salton, J. Allan, C. Buckley, and A. Singhal. Automatic analysis, theme generation, and summarization of machine-readable texts. In M. Agosti and A. Smeaton, editors, *Information Retrieval and Hypertext*, Information Retrieval and Hypertext, pages 51–73. Springer US, 1996.

[207] W. Sansrimahachai, M. J. Weal, and L. Moreau. Stream ancestor function: A mechanism for fine-grained provenance in stream processing systems. In *2012 Sixth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12. IEEE, May 2012.

[208] M. Seltzer and P. Macko. Provenance Map Orbiter: Interactive Exploration of Large Provenance Graphs. In *Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP'11)*, pages 20–21, 2011.

[209] B. Shneiderman. *The new ABCs of research: Achieving breakthrough collaborations*. Oxford University Press, 2016.

[210] C. Sigovan, C. Muelder, K.-L. Ma, J. Cope, K. Iskra, and R. Ross. A visual network analysis method for large-scale parallel I/O systems. In *IEEE 27th*

International Symposium on Parallel Distributed Processing (IPDPS), pages 308–319, May 2013.

[211] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *ACM SIGMOD Record*, 34(3):31, Sept. 2005.

[212] A. Spinuso, J. Cheney, and M. Atkinson. Provenance for seismological processing pipelines in a distributed streaming workflow. *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, 2013.

[213] A. Spinuso, R. Filgueira, M. Atkinson, and A. Gemuend. Visualisation methods for large provenance collections in data-intensive collaborative platforms. In *EGU General Assembly Conference Abstracts*, volume 18, 2016.

[214] S. Sweeney. The ambiguous origins of the archival principle of provenance. *Libraries & the Cultural Record*, 43(2):193–213, 2008. University of Texas Press.

[215] G. Terstyanszky, T. Kukla, T. Kiss, S. Winter, and P. Kacsuk. Sharing workflows through coarse-grained workflow interoperability. In *Grid Computing*, volume 3, 2005.

[216] D. Tunkelang. aceted search (synthesis lectures on information concepts, retrieval, and services). *Claypool: Morgan*, 2009.

[217] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 42:1–42:6, New York, NY, USA, 2010.

[218] N. N. Vijayakumar and B. Plale. Towards low overhead provenance tracking in near real-time stream filtering. In *Proceedings of the 2006 international conference on Provenance and Annotation of Data*, IPAW'06, pages 46–54, Berlin, Heidelberg, 2006. Springer-Verlag.

[219] T. Weigel and T. DiLauro. Separation of concerns: Pid information types and domain metadata. In *International Conference on Dublin Core and Metadata*

Applications (DC-2013). Dublin Core Metadata Initiative. Lisbon, Portugal, 2013.

[220] T. Wilde, A. Auweter, H. Shoukourian, and A. Bode. Taking advantage of node power variation in homogenous HPC systems to save energy. In *High Performance Computing - 30th International Conference, ISC Frankfurt, Germany, July 12-16, 2015, Proceedings*, pages 376–393.

[221] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.

[222] E. Wong, T. Wong, and J. Z. Kolter. dreaml: A library for dynamic reactive machine learning. In *Workshop on Machine Learning Systems at Neural Information Processing Systems (NIPS)*, December 12, 2015.

[223] X. Xiang and G. Madey. Improving the reuse of scientific workflows and their by-products. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 792–799. IEEE, 2007.

[224] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark : Cluster Computing with Working Sets. *HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, page 10, 2010.