H2020-EINFRA-2017

EINFRA-21-2017 - Platform-driven e-infrastructure innovation

DARE [777413] "Delivering Agile Research Excellence on European e-Infrastructures"

D2.2 DARE Architecture and Technical Positioning II

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D2.2 DARE Architecture and Technical Positioning II
Work package	WP2: Architecture Specification and Innovation
Tasks involved	T2.1: Architecture Specification, Tools and Components
Type	R: Document, report
Dissemination Level	P = Public
Due Date	31/12/2020
Submission Date	30/12/2020
Status	Draft
Editor(s)	Malcolm Atkinson (UEDIN) Iraklis Klampanos (NCSRD)
Contributors	Valentina Andries (UEDIN) Malcolm Atkinson (UEDIN) Aurora Constantin (UEDIN) Rosa Filgueira (UEDIN)

	André Gemünd (FRAUNHOFER) Ellen Gottschämmer (KIT) Vangelis Karkaletsis (NCSRD) Iraklis Klampanos (NCSRD) Antonis Koukourikos (NCSRD) Amélie Levray (UEDIN) Mike Lindner (KIT) Federica Magnoni (INGV) Christian Pagé (CERFACS) Andreas Rietbrock (KIT) Alessandro Spinuso (KNMI) Chrysoula Themeli (NCSRD) Xenofon Tsilimparis (GRNET) Fabian Wolf (FRAUNHOFER)
Reviewers	Xenofon Tsilimparis (GRNET) (§1, §2 & §3) André Gemünd (FRAUNHOFER) (§4) Alessandro Spinuso (KNMI) (§5) Vangelis Karkaletsis (NCSRD) (§5 & §6)
Document description	Development of the DARE architecture since D2.1 and innovation planning to address the requirements of DARE user communities and to enhance sustainability. Assessment of progress.

Document Revision History

Version	Date	Change made	Contributor
1	19/10/2020	Initial draft started	Malcolm Atkinson (UEDIN)
2	29/10/2020	Updated section 4.3	Alessandro Spinuso (KNMI)
3	5/11/2020	Revised for meeting with authors	Malcolm Atkinson (UEDIN)
4	5-18/11/2020	Added sections & revised to end section 2	Malcolm Atkinson (UEDIN)
5	24-25/11/2020	Section 3.1.1 revised	Federica Magnoni (INGV)
6	24/11/2020	Section 3.2 revised	Christian Pagé (CERFACS)
7	25/11/2020	Section 5.3 drafted	Valentina Andries (UEDIN) Aurora Constantin (UEDIN)
8	26/11/2020	Added Section 3.1.2	Mike Lindner (KIT)
9	30/11/2020	Checked and added to Section 3.1.2 and checked 5.3.1	Ellen Gottschämmer (KIT)
10	30/11/2020	Revised Section 5.1	Xenofon Tsilimparis (GRNET)
11	1/12/2020	Reviewed sections 1 to 3	Xenofon Tsilimparis (GRNET)
12	3/12/2020	Revised section 4.2	Malcolm Atkinson (UEDIN)
13	4/12/2020	Revised section 4.4	Malcolm Atkinson (UEDIN)

		Revised section 3.1	Federica Magnoni (INGV)
14	7/12/2020	Rewrote section 5.3.2	Valentina Andries (UEDIN)
15	8/12/2020	Wrote section 5.4	Malcolm Atkinson (UEDIN)
16	10/12/2020	Added DKB implementation and demonstration in section 4.2	Amélie Levray (UEDIN) Malcolm Atkinson(UEDIN)
17	11/12/2020	Revised 4.2.5	Malcolm Atkinson (UEDIN)
18	12/12/2020	Section 6 revision	Malcolm Atkinson (UEDIN)
19	14/12/2020	Layout tidied & final check	Malcolm Atkinson (UEDIN)

Executive Summary

DARE is an ambitious project that aims to provide novel approaches for creating and using data-powered methods at the frontiers of today's research and innovation. DARE's central goal is to support research developers – domain-expert software developers – to transparently make use of European e-infrastructures, research infrastructures and other platforms and software in order to create data- and computationally-intensive applications for their domains. DARE aims to achieve these goals by providing much needed technology and methodology aligned with EOSC developments.

This deliverable presents the progress since D2.1 with an interim report ID2.2 [Atkinson *et al*. 2020] and the new understanding of what is required from the DARE architecture, driven by the interplay of user requirements and technological opportunities. These are constrained and enhanced to yield future utility, extensibility and sustainability. New requirements for users to directly create and control complex computational and data challenges are pushing DARE to extend its integration, automation and optimisation. The initial progress with these is presented and evaluated. They lead to a strategy for sustainability.

Table of Contents

1. Introduction

The DARE architecture has to pursue two goals:

1. Shape the DARE platform and its future versions to meet emerging and anticipated requirements and thereby improve user communities' research work, and
2. Identify frameworks and strategies that will be extensively used to improve return on investment and sustainability.

This requires a practical balance between delivering the required capabilities to the two DARE user communities and a longer-term vision. The architecture described in D2.1 [Atkinson *et al.* 2018] shaped the first platform release [Klampanos *et al.* 2019]. The overall structure interlinking three major subsystems, as shown in Figure 1.1 has proved successful and has been retained.

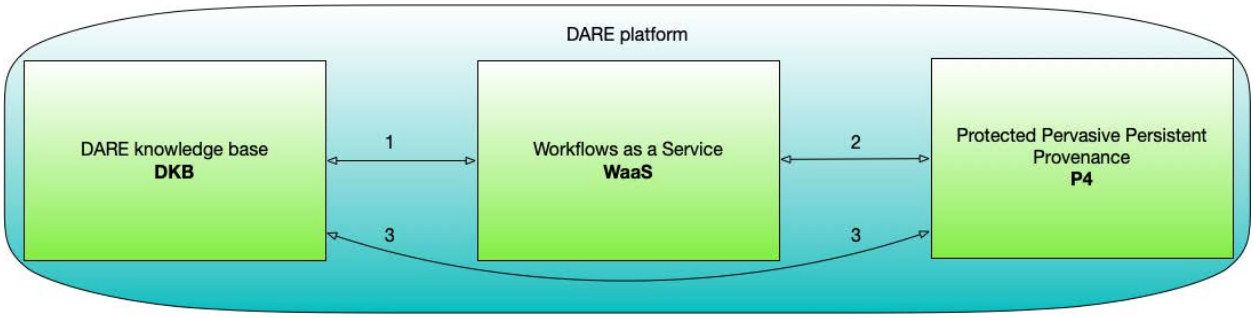


Figure 1.1: *The three principal subsystems forming the DARE platform.*

However, each technological pillar has been further developed, as outlined here and as reported in more detail in Section (**§**) 4.

1. A lightweight integration of the DARE Knowledge Base (**DKB**) [Atkinson & Levray 2019] makes it more easily used by the other subsystems and as a standalone service. It enables incremental introduction into established research environments and provides a foundation for increased abstraction, automation and stability co-existing with innovation. It will provide an API and a Python library to enable developers and methods to use it directly. It has two novel features: (a) research Contexts to manage scope within its information space, and (b) a Conceptual library to accelerate productive use and help with organising and interpreting the information. See §4.2 for more details.
2. The Workflows-as-a-Service (**WaaS**) has developed containerisation, orchestration and dynamic deployment of dispel4py data-streaming workflows to meet demanding user needs [LIang *et al*. 2020]. It includes CWL[1] formalised workflows as part of that work, and to extend the range of methods facilitated. See §4.1 for details.
3. The provision and exploitation of provenance via the Protected Pervasive Persistent Provenance (**P4**) subsystem has extended its scope, configurability and visualisation, as reported in [Spinuso *et al.* 2019]. It aims to deliver Reproducibility-as-a-Service (**RaaS**). The adoption of provenance by scientists is being incentivised by more powerful provenance-driven tools. See §4.3 for details.

[1] W3C Common Workflow Language https://www.commonwl.org/

The combined and released platform is supporting challenging data-intensive and computationally demanding scientific methods and making them easy to deploy and use. For example, the seismic rapid assessment calculation and comparison of ground motion - see §3.1. This supports the next stages of seismological research and of climate-impact modelling. These can be generalised to accommodate additional communities.

The current implementation is presented in §4.4. The WaaS handles dispel4py and CWL. It employs optimisation when mapping dispel4py workflows onto production platforms, to improve scalability. The DKB functionality provides an extensible and flexible information sharing facility that should prove easy to use and thereby aid self sufficiency. The provenance handling provided by P4 has extensive collection capacity connected by parameterised queries to provence-powered tools. It delivers reproducibility-as-a-service, incentivising the use of provenance by application communities. A foundation for reproducible science and minable records of scientific procedures and progress. Offering convenient WaaS and good provenance-driven tools is an essential step in achieving trustworthy evidence underpinning major decisions.

Sustainability is essential. Research communities will not learn how to use advanced technologies if there is a risk that they will disappear. However, this requires the commitment of resources for maintenance and support for the long term. All of the stakeholders need to share this responsibility - see §5.1. Sustainability depends on establishing value and having that value recognised and on recruiting sufficient support. Costs should be minimised by careful engineering and by progressively empowering user communities to be self-sufficient by reducing the hurdles encountered and by simplifying, vautomating and eliminating tasks.

The work of a research community uses resources they have access to, e.g., EOSC or institutional or regional resources. Section 5.2 identifies the available services and how they are deployed and protected.

DARE focuses on supporting research engineers/developers, who work with research communities identifying and exploiting new opportunities from innovations in ICT and its applications. An analysis of questionnaires and interviews assessed how far DARE was successful - see §5.3.

Section 6 draws together all these issues and proposes a way forward that continues to extend the DARE platform's capabilities, while improving self-sufficiency and sustainability. The sustainability strategy presented is DARE's commitment to extending the range of applications adopting DARE's approach and architectures.

This is a final deliverable. Several publications are being prepared. Others joining in or adopting DARE's approach and aspects of its architecture will always be welcome. Consequently, we invite allies, users, criticisms, observations or advice pertinent to DARE and its applications. Please email: Malcolm.Atkinson@ed.ac.uk or iaklampanos@iit.demokritos.gr and we will respond, take account of suggestions and acknowledge contributions.

2. Architecture overview

Role and context

The DARE architecture should shape a framework that facilitates ambitious research undertaken by distributed, loosely federated multi-disciplinary communities typified by the solid-Earth and climate communities DARE works with. This imposes several requirements and constraints.

1. The work of research developers and specialists should retain its value as digital technology evolves. This requires their work should be expressed precisely and abstractly so that it can be mapped (as far as possible automatically) to new digital infrastructures. This should accelerate advances as new power becomes available while minimising disruption and loss of methods and established practices. As reported in §3 and §4, DARE has already made significant progress towards this goal. Meeting it also facilitates deploying the DARE platform on a diversity of individual, institutional and regional computing services.
2. Multiple expert viewpoints co-exist, as illustrated in Figure 2.1. Their collaboration should be facilitated, e.g., between:
    a. Application *domain experts* who set goals, pioneer new research methods and organise teams, resources and campaigns.
    b. *Research developers* who draw on RSE products (see below), compose, package, steer and revise those elements to deliver tested contributions to their application community's goals.
    c. *Research Software/Systems Engineers (RSEs)* who have specialist knowledge in some aspects of computer science, distributed systems engineering, simulation systems, data analytics, etc. They draw on theoretical and practical advances and develop subsystems, libraries, simulation codes, etc. for use by multiple application communities.
    d. *Resource providers* who establish and sustain computation, storage, information and other resources as services on which research communities depend.
3. The architecture has to be implementable, sustainable and affordable while meeting today's goals as rapidly as possible. At the same time, it has to deliver a good foundation on which to build support for future research goals exploiting emerging and specialised technologies. Keeping these immediate and longer-term considerations in balance is an architectural duty with a concomitant obligation to communicate with and gain buy-in from all of the stakeholders - the range of professionals listed above and a broad spectrum of citizen scientists[2].

[2] During a career individuals transition between these roles and sometimes span more than one. We use the term 'Research Engineer' to encompass 2b and 2c above in this document.

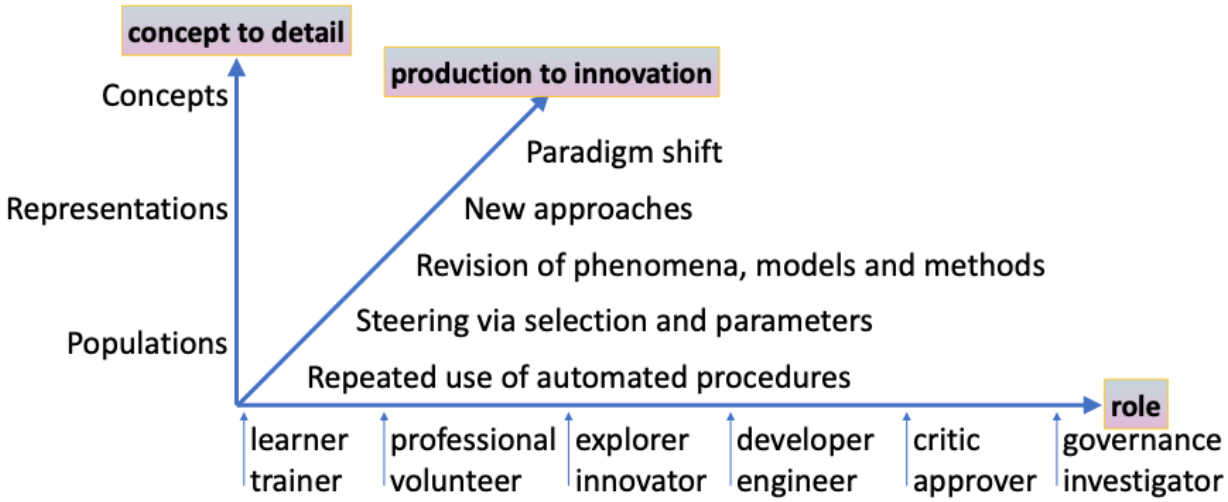


Figure 2.1: *Diverse roles are shown on the x axis. They may have inherently different viewpoints which change as their activities move in the other two dimensions. Research success depends on effective collaboration between these viewpoints while avoiding being slowed by attempting to bring everything into a rigid consistent framework.*

Architectural goals

The architecture therefore has to achieve the following goals to meet stakeholders' priorities.

1. *Delivering power and control to each user community*. Ideally, they should be able to immediately implement and use new methods exploiting all of the available computational power and the full richness of available data to address their most demanding and complex research challenges without undue disruption of their existing practices and knowledge infrastructure [Edwards 2013].
2. *Reducing the application communities' dependence on IT specialists*. A synergy between computational experts and domain experts will always be necessary to push the frontiers of research or to polish the optimisation of intensively used methods. However, most of the needed innovation should be achievable by the application communities themselves.
    a. Depending on others requires investment in explaining what is wanted and introduces delays, leads to divergence from goals and loss of sustainability.
    b. It means that an application community takes longer to spot new opportunities.
3. *Achieving affordable long-term sustainability*. Ultimately, all software on which the application communities depend has to be maintained and supported[3]. As far as possible, the DARE platform should be built using standard components that are widely used and therefore their maintenance is amortised over an extensive community[4]. The remaining software which tailors and integrates existing software and services has to be engineered with maintenance in mind[5]. This is equivalent to considering the operational and maintenance costs for a building. In the end, each application community has to meet its

[3] Maintenance is approximately 90% of lifetime software costs. Open source doesn't remove this cost, it redistributes it.
[4] E.g., global alliances for environmental science, geo-spatial, or data-intensive and computational R&D.
[5] The Software Sustainability Institute, https://www.software.ac.uk/, develops and promotes the required standards.

share of these costs, either by finding the resources, expert staff time or funds, or by persuading its funders to top-slice budgets to meet these common requirements.

Significant progress towards the first goal has been made in DARE, see §3 and [Klampanos *et al.* 2020, Atkinson *et al.* 2019, Klampanos *et al.* 2019, Pagé *et al.* 2019a, Spinuso *et al.* 2019 & Magnoni *et al.* 2019a].

Architectural vision

Figure 2.2 shows the ideal state when these goals are reached. Agile application teams incrementally build and test complex methods. When these are judged ready by application experts, they are moved to production. There those methods are repeatedly used, on specified targets, with steering of diagnostics, provenance collection, data handling and parameter revision under the control of a broad community of practitioners. They comply with a community's agreed standards. When improvements are identified the application team can implement them and deploy the improved version. Work is still required to generically support this goal and to fully automate and optimise production. This includes tools to make this easier for research engineers and to reduce the need to master technical detail. Both of which will help with system mobility and address the users' requirement to be able to instal and run DARE on their local, community or national computational services.

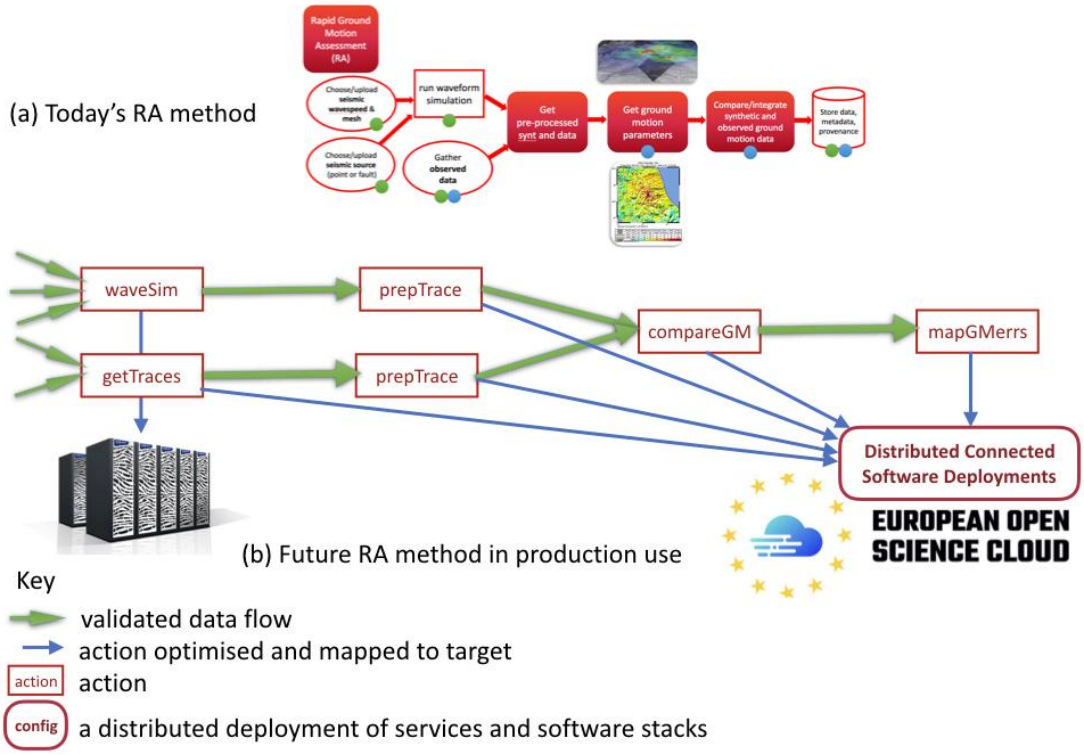


Figure 2.2: *Fluent path from an application group's development of a method to production (see 3.3 for details). The method (in this case for computational seismologists performing rapid assessment of ground motion) is developed and repeatedly refined and tested by the application experts. When they judge it ready it is moved to production automatically on the specified or automatically chosen target computing, data and network services.*

Application communities in the driving seat

The current intense interaction with specialist computer scientists, distributed systems engineers, data scientists and data architects is highly beneficial and leading to rapid progress. However, it depends on research-project funding[6]. That level of research funding designed to stimulate innovation cannot be indefinitely sustained or spread to much wider communities undertaking application-domain focused long-running campaigns and application-led R&D. For these reasons, and for the reasons given above, it is essential to arrange that the *majority* of this work can proceed effectively and efficiently without sustained specialist input. The need for specialist input at places where significant innovation is needed will always reappear.

Achieving application-domain control and self-sufficiency depends on three interrelated objectives:

[6] A succession of eInfrastructure and big-data projects that have built the capabilities and skills we draw on; they were invaluable and much appreciated.

1. *Improving automation* so that there are fewer administrative tasks and less need for users to provide information that could be supplied. This reduces the rate of failures during development, learning and production.
2. *Raising the level of abstraction* so that their work is mainly achieved using stable and implementation-independent concepts, terms and data - less to learn and fewer occasions when it is necessary to re-learn.
3. *Intellectual ramps* that facilitate the acquisition of new skills incrementally, without having to climb over substantial thresholds before users benefit from a new skill - delivering a quick return on intellectual investment and exploration.

Recognising key roles

These improvements need to be delivered for each role in a community's team; but, in DARE, we focus on those for *application specialists* and *research engineers*. However, if we meet the requirement to deploy instances of the DARE platform on institutional facilities, then the *systems administrators* who install and support those instances will also need consideration. These aspects of the DARE architecture increase in importance as the basic functionality is delivered. Progress may be found in §3 and §4 respectively.

Achieving sustainability

Sustainability is critical for two reasons.

1. Without it we are *behaving unethically*, by leading application communities to depend on a research environment that may disappear, leaving them a difficult recovery path finding replacements and reformulating their methods and working practices.
2. Without it the *return on investment* is lost; the funds put in by our funders, ultimately European taxpayers, and the effort put in by many researchers, developers and engineers will yield very little.

Sustainability is hard to assess. It can only be measured in retrospect. It depends on the balance between the cost of sustaining facilities (maintenance and support) and the available resources. The latter depends on two factors:

1. The importance the application communities and their funders attach to it, which depends on the quality and power of the system, and
2. The breadth and scale of the user communities.

DARE seeks to minimise costs by building on widely used software components. For example, the Python and notebook technologies used are very widely adopted and supported[7]. And by adopting professional software and systems engineering practices; e.g., those recommended by SSI (§4). Sustainability and the development of take up is covered in §5.

Implementing the Architecture

The DARE platform reflects these architectural goals. It is accessed via the DARE API (a collection of RESTful services) and, during development (§3.3) via the DARE playground. This is

[7] Notebooks are particularly useful in combining documentation with functionality. They still need to be used carefully, i.e., avoiding distracting detail that reduces learning success, mobility and longevity.

made easier using a Python library called the DareManager. This realisation is shown in Figure 2.3. See §4 for details, after the uses of the DARE platform are presented in §3.

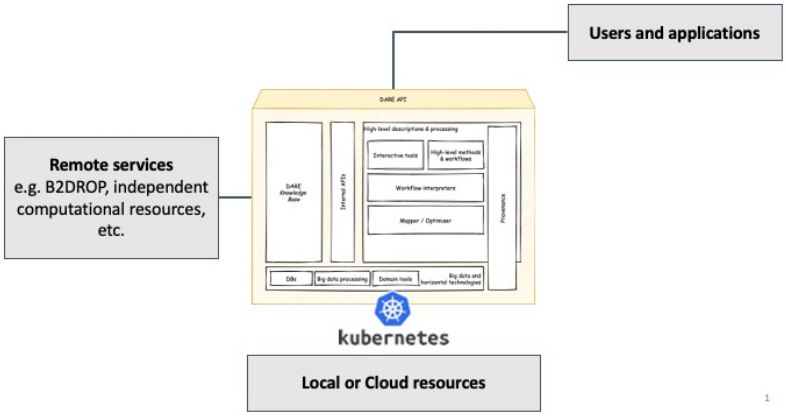


Figure 2.3: *The operational context and use of the DARE platform showing its major components (see §4 and §5.2) and its interaction with its environment. It is deployed and orchestrated by kubernetes. It is used via a RESTful API using a library of Python wrappers delivered as a DareManager. It calls on an open-ended set of external services and data sources.*

3. User requirements and experience

Architectural components such as WaaS and aspects of P4 have been integrated within the platform, further extended and used by the communities' workflows. Aspects of the DARE API exposing the WaaS to research developers have had their usability, stability, portability, maintainability and performance improved. These included the management of input and output files, authentication and the use of Jupyter Notebooks.

The communities represented by WP6 and WP7 have tuned their provenance traces by specifying the levels of detail and the metadata to be recorded during the execution of their experiments. The framework allows for configuration and detailed extraction of customised information. This is delivering benefits in terms of results management, discovery of relevant past runs and reproducibility.

DARE supports the execution of different workflow technologies (dispel4py and CWL). dispel4py can be explicitly implemented and configured by the application experts and their research engineers. CWL is used to organise and execute tasks. Provenance traces are gathered for both of these by P4 and are then used by the tools it supports. CWL is used for the SPECFEM parts of seismic-simulation workflows (§3.1.1), the Fall3Dpy simulation for vulcanology (§3.1.2) and to organise the cyclone tracking system (§3.2). The two systems have different approaches to provenance generation but these are harmonised via s-ProvFlow (§4.3).

Other aspects which are relevant to an effective use of the DARE conceptual design and architecture is the management of user's identities and how these are handled across all of the DARE components and microservices while enabling each community to use its established mechanisms (§5.1).

Both communities of seismologists and climate scientists, and recently the volcanology community, benefit from the adoption of remote development environments based on executable notebooks (Jupyter, Jupyter Lab)[8] [Rule, Adam, *et al.* 2018] . Notebooks became familiar among computational scientists because they facilitate the generation and sharing of documentation of methods, source-code and results in a single *de-facto* standard format. The successful support of such tools requires the DARE API to include a set of utilities that allow users to develop, execute and evaluate the results of a workflow within the same notebook.

The adoption of advanced and interactive development environments, such as notebooks, in contexts where reproducibility is a priority, opens challenges concerning the correct use of such tools and the realisation of mechanisms that facilitate consistent provenance acquisition and interpretation. This has to capture changes to computational environments, such as software stacks, configurations and resources. The latter include new algorithms as well as data. Concepts concerning system and application domain need to be combined to represent setup and exploitation of the computational spaces in a way that guarantees the consistent interpretation of

[8] https://jupyter.org/

the various entities involved in the longer term. Although enactment technologies might change over time, the provenance records should guarantee that researchers are able to locate and understand failures during attempts to reproduce a certain result or re-apply a method. DARE in cooperation with the ENVRIFair project[9] is addressing these challenges (§4.3).

3.1 Use by EPOS communities

Collaborative work of domain specific scientists, data architects and developers produced significant advances in the design and implementation of the EPOS seismological use case with its test cases (Deliverable D6.1 [Rietbrock *et al.* 2018]). Additionally, based on the mid-term review, we slightly broadened our scope to a new community in the EPOS framework, volcanology, to exemplify the versatility of the approach of the DARE platform and API. This is achieved by introducing a new test case which involves workflows and tools useful for the volcanology community (Deliverables D6.2 [Magnoni *et al.* 2020a] and D6.4 [Magnoni *et al.* 2020b]).

A summarised framing of the EPOS test cases is outlined below. Then, §3.1.1 and §3.1.2 contain a more detailed description of each case.

- Rapid Assessment (**RA**) of ground motion test case

Community involved:
EPOS Earth Science community
Scientific problem:
Rapid calculation of seismic ground motion parameters after an earthquake
User requirements:
Handling of HPC numerical codes; implementation of complex, user customised scientific procedures; shareability and reusability of procedures and their substeps; investigation and reproducibility of results and errors; access to external RI services; monitor of job execution
DARE approach/solution:
Docker and CWL workflow to manage the execution of a numerical simulation code; dispel4py workflows to manage the other substeps of the test case; customised provenance and metadata capturing; consistent shared file system; Jupyter Notebook executed in Jupyter Lab
Final experience:
Successful implementation and execution of the workflow in the DARE platform via Jupyter Notebook; promising expectations from the first seismological training event

- Moment Tensor in 3D (**MT3D**) test case

Community involved:
EPOS Earth Science community
Scientific problem:
Study of earthquake point-like source parameters and uncertainties using 3D Earth structures
User requirements:

[9] ENVRIfair EU project https://envri.eu/

Handling of HPC numerical codes; implementation of complex, user customised scientific procedures; shareability and reusability of procedures and their substeps; investigation and reproducibility of results and errors; access to external RI services; monitoring of job executions
DARE approach/solution:
Docker and CWL workflow to manage the execution of a numerical simulation code also for multiple jobs with different inputs; dispel4py workflows to manage the other substeps of the test case; customised provenance and metadata capturing; consistent shared file system; Jupyter Notebook executed in Jupyter Lab
Final experience:
Successful implementation and execution of the workflow in the DARE platform via Jupyter Notebook; positive outcomes from the second seismological training event

- Volcanology (**VC**) test case

Community involved:
EPOS Earth Science community
Scientific problem:
Analyse ash dispersal models after volcanic eruptions
User requirements:
Handling of HPC numerical codes; implementation of complex, user customised scientific procedures; shareability and reusability of procedures and their substeps; investigation and reproducibility of results and errors; access to external RI services; monitoring of job executions
DARE approach/solution:
Docker and CWL workflow to manage the execution of a numerical simulation code; dispel4py workflows to manage the other substeps of the test case; customised provenance and metadata capturing; consistent shared file system; Jupyter Notebook executed in Jupyter Lab
Final experience:
Successful implementation and execution of the workflow in the DARE platform via Jupyter Notebook; positive outcomes from summer school training event

3.1.1 Use by Seismologists

Following the requirements detailed in Deliverable D2.1 §7.1 [Atkinson *et al.* 2018] and exploiting the main components described there, we started focusing on the **RA test case**. The aim was to structure a workflow that could help researchers ease and speed-up the calculation of seismic ground motion parameters (such as the peak ground acceleration (pga), peak ground velocity (pgv) or peak ground displacement (pgd)), especially after large earthquakes, generating specified outputs useful both scientifically and for communication with public and emergency authorities. We used the RA test case as a typical example of our working methods to steer the DARE platform development and build an easy-to-use, customisable framework made of reusable, abstract and flexible components that can serve multiple purposes and extend beyond the immediate EPOS seismological community.
The RA workflow has been designed with modular high-level steps that are represented in Figure 3.1 and described in Deliverables D6.1 [Rietbrock *et al.* 2018], D6.3 [Magnoni *et al.* 2019b] and D6.2 [Magnoni *et al.* 2020a] where examples of the output results are also shown.

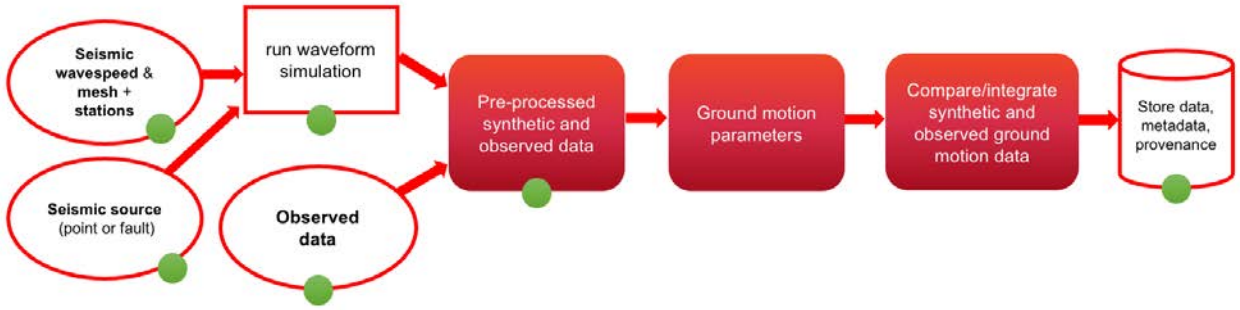


Figure 3.1: *The Rapid Assessment (RA) workflow*. *Green dots are the steps in common and reusable in the MT3D test case (Fig. 3.2).*

The implementation and execution of RA has been made possible by the development of the DARE API components delivered by the first and second releases of the DARE platform (Milestones MS6 & MS21) and their updates for the third release (MS10). In particular, fundamental components meeting this aim are (see §4):

- The *Execution API* to enable distributed and scalable execution of simulation codes and dispel4py workflows;
- The *dispel4py Registry* (or Processing Elements Library) to provide a workspace structure for registering workflow entities (as processing elements) supporting reusability and sharing;
- The *CWL Workflow registry* that acts as the registry for workflows based on CWL;
- The *Provenance components* sProv and sProv-viewer to record metadata and provenance and offer visualisation functionalities.

Exploitation of DARE API components to perform workflows is realised through Jupyter Notebooks executed in a Jupyter Lab[10], a development environment introduced in DARE during the second phase of the project, as detailed in §3.3. Specifically, the notebook allows users to access the API functionalities to:

- register dockers, and dispel4py and CWL workflows;
- launch numerical simulations, specifically with the code SPECFEM3D_Cartesian (Fig. 3.1), with a simple API call that executes on the DARE cluster a dockerized version of the code containing all the required dependencies:
  dm.exec_cwl(nodes=nodes,input_data={input_data})
  where input_data is a dictionary specifying the input files (see §3.3 and D6.4 [Magnoni *et al.* 2020b] for details); in particular, the steps needed for a SPECFEM3D simulation are implemented as a CWL workflow launched by executing the SPECFEM3D docker;
- execute dispel4py workflows, as those describing the other steps of RA (Fig. 3.1), through other specific API calls that allow users to specify needed inputs and requirements (see §3.3 for more details):
  dm.exec_d4p(nodes=nodes,no_processes=no_processes,iterations=iterations,target=d4p_model ,prev_run_id=prev_run_id,reqs="requirements",inputdata=input_data) ;

[10] https://jupyter.dare.scai.fraunhofer.de

- upload required input files (as user customised input models, see Fig. 3.1), monitor the launched jobs, check output directories, check and download useful output files.

A refinement phase followed the initial implementation of the RA test case in order to remove obsolete intermediate, fine-grain steps that produced input files for main steps or that post-processed outputs from previous main steps. This results in an even more modular workflow constituted by proper, self-contained dispel4py sub-workflows that perform specific tasks and can be executed, parallelised at scale, by themselves (if the required input files are already available) or in a pipeline. Thus, they can be easily reused for other workflows or can be customised or updated in the future without the need of modifying the whole procedure.

The provenance record has been implemented and recently refined for the RA dispel4py and SPECFEM3D CWL workflows and it is easily customisable by users.

The latest versions of the developed codes are available in a git repository:

- Jupyter Notebook for RA test case
  https://gitlab.com/project-dare/dare-examples/-/blob/master/wp6/WP6_RA.ipynb
- dispel4py sub-workflows composing the test case
  https://gitlab.com/project-dare/WP6_EPOS/-/tree/RA_total_script/processing_elements/Download_Specfem3d_Misfit_RA
- SPECFEM3D docker and CWL workflow
  https://gitlab.com/project-dare/dare-platform/-/tree/master/containers/exec-context-specfem3d
  https://gitlab.com/project-dare/dare-platform/-/tree/master/containers/specfem3d
  https://gitlab.com/project-dare/WP6_EPOS/-/tree/RA_total_script/specfem3d/specfem3d_test_input_cwl

The next EPOS test case taken into account during the second DARE phase focused on the analysis of the parameters that characterise the earthquake source and uncertainties of these parameters, a key study for many seismological applications. For the chosen pilot, the seismic source was approximated as a point source and the studied parameters were the earthquake location, magnitude and rupture mechanism represented by the moment tensor, hence up to 9 free parameters (see D2.1 [Atkinson *et al.* 2018] and [Aki & Richards 1980]). The final goal was to improve an initial model of the earthquake source by calculating the perturbations to its parameters that minimised the misfit between simulated and recorded waveforms, a typical inverse problem, and estimating the uncertainties attributed to the new solution. Since this application specifically considered a 3D model to represent the Earth structure and invert for moment tensor solutions, we named it **Moment Tensor in 3D (MT3D)**. The workflow structure is represented in Fig. 3.2 and described in Deliverables D6.2 [Magnoni *et al.* 2020a] and D6.4 [Magnoni *et al.* 2020b] where examples of the output results are also shown.

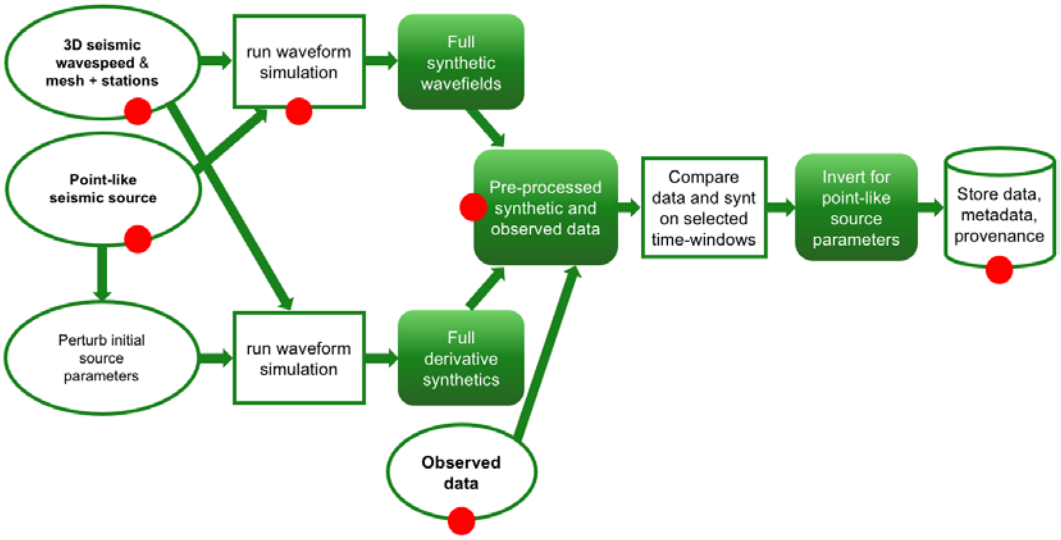


Figure 3.2: *The Moment Tensor in 3D (MT3D) workflow*. *It calculates improved seismic source parameters by minimising the misfit between recorded and simulated waveforms. Red dots are steps in common with RA test case (Fig. 3.1)*

To implement and execute the MT3D workflow in the DARE platform we benefit from the common steps with RA (see the red dots in Fig. 3.2) already implemented as described above and easily reusable. A new important step is then the simulation of the synthetics for perturbed source parameters called 'derivative synthetics'. The basic simulation with SPECFEM3D is the same as RA but now multiple simulations with different input files need to be managed. This has been handled through the same Kubernetes pod that runs multiple simulations. Moreover, related metadata and provenance information should be carefully handled in order to combine these simulated derivative synthetics in the following steps of the procedure. This will also enable reusability for future seismic events with a similar starting solution thereby progressively and significantly reducing computation time and costs as the system will have stored more and more well-described fundamental solutions over time. Finally, the last two steps, specific for MT3D, have been performed by constructing a new dispel4py workflow and API call to execute well-established Python codes (already cited in D2.1 [Atkinson *et al.* 2018], D6.2 [Magnoni *et al.* 2020a] and D6.4 [Magnoni *et al.* 2020b]): pyflex[11] for the selection of the time windows suitable for waveform comparison and inversion, and pycmt3d[12] for seismic source inversion in a three-dimensional Earth structure. Future update/substitution of these codes could be necessary as the science advances. It is essential that such upgrades should be straightforward. **The simplicity of future upgrades must be taken into account when making implementation decisions[13].**

[11] pyflex, L. Krisher; http://krischer.github.io/pyflex
[12] pycmt3d, https://github.com/wjlei1990/pycmt3d

[13] This simplicity requirement, "*it remains easy to install and use new versions of application-domain software and services now and after the project ends*", applies in virtually every aspect of applications development and platform use. **It is a critical aspect of sustainability.** WP6 is exposing it first.

Metadata and provenance are captured and stored for the MT3D dispel4py workflows (as well as for SPECFEM3D) following the same approach as used in the RA test case.

The development and implementation of the MT3D test case has been facilitated by the recent creation of a playground framework, as described in §3.3. It allows users to directly test and debug their workflows by launching 'debug API calls' in the Notebook getting inline output and logs. Further exploitation of the Jupyter Notebook framework could become fundamental if used as an environment where processing elements can be directly developed, registered and executed.

Based on the experience with the RA test case, a similar Notebook has been created for MT3D in order to execute its specific workflow steps through the Jupyter Lab. The flexibility aimed for the DARE platform is well demonstrated by noting that the structure of the API calls to execute the sub-workflows (both CWL and dispel4py) is the same as described above for RA. This eases the implementation of the new steps for which the only specificities are suitable input and requirements.

The latest versions of the developed codes for MT3D test case are in the DARE git repository (SPECFEM3D docker and CWL workflow are the same as RA test case above):

- Jupyter Notebook for MT3D test case
  https://gitlab.com/project-dare/dare-examples/-/blob/master/wp6/WP6_MT3D.ipynb
- dispel4py sub-workflows composing the test case
  https://gitlab.com/project-dare/WP6_EPOS/-/tree/RA_total_script/processing_elements/MT-3D/workflow

In general, from the point of view of both research developers and domain experts, we can highlight some significant advantages of exploiting the DARE platform for the EPOS use case but also for more general scientific applications:

- Exploiting the Cloud for execution to provide elasticity in acquiring and using resources and make on-demand computing and storage resources available.
- Transparent set up and execution of runs without the need to deal with environment specificity and details of code/scripts execution. Here a single call is used to do all the required steps to prepare the environment and run a SPECFEM3D simulation.
- Exploiting Research Infrastructure (RI) services by including them in the whole workflow procedure, so taking care of the required input, query parameters and gathered output. Here a simple call allows users to query FDSN web services of European archives to download recorded waveforms.
- Rapid and transparent data analyses and transfer between co-working environments.
- Automatic description and storage of complete lineage and multiple metadata that allow us to track runs and data through the whole workflow, to easily search and reuse them and to also combine numerous outputs from multiple workflows that are in widespread use in many scientific applications.
- High-level description of workflow steps that are as abstract as possible to increase the flexibility in reusing them to assemble different workflows.

- Existence of managed knowledge-bases (e.g. the PE registry) that allows users to easily exchange information about what they deployed and executed.
- Workflow structure and provenance information that can be customised.

The incremental advances in the platform components and structure will favour the development of more complex test cases. In the seismology framework an example could be the Ensemble Simulation analyses (cited in D6.1 [Rietbrock *et al.* 2018]) that statistically characterises ground motion parameters and their uncertainties and that would combine multiple executions of both RA and MT3D test cases. This exemplifies once again how the powerful capabilities of the DARE platform of reusing and combining workflows as much abstract as possible and keeping track of the runs and errors are key for science.

The DARE approach could then be interesting for other communities, beyond the EPOS seismologists, who are looking for a powerful and easy-to-use framework to develop their applications. For this reason, WP6 spent part of the second phase of the project engaging with the volcanology community by developing a new test case focused on the implementation of a volcanological application, as described in §3.1.2 and Deliverables D6.2 [Magnoni *et al.* 2020a] and D6.4 [Magnoni *et al.* 2020b].

At the end of the first reporting period a training event was organised in order to present the execution of the RA workflow running on the first release of the DARE platform (D6.3 [Magnoni *et al.* 2019b] and D8.4 [Casarotti *et al.* 2019]). A second training event was carried out as webinar (due to COVID-19 emergency) to present the execution of the new seismological test case MT3D through the developed Jupyter Lab and the latest release of the platform (D6.4 [Magnoni *et al.* 2020b] and D8.5 [Magnoni *et al.* 2020c]). The main advantages of using the platform have been successfully caught by the trainees, while useful suggestions have been gathered on platform development and aspects of the pilot implementation that could be improved. For the last training detailed documentation has been prepared to guide the trainees and future users on DARE platform methods and use[14].

3.1.2 Use by Volcanologists

In addition to the original test cases, we have developed and implemented the **Volcanology (VC) test case** as described also in Deliverables D6.2 [Magnoni *et al.* 2020a] and D6.4 [Magnoni *et al.* 2020b], based on the suggestions from the mid-term review. The scientific purpose is to analyse ash dispersal after volcanic eruptions including distributions of deposit thickness, ground load and airborne mass.

[14] Tutorial Jupyter Notebook, https://gitlab.com/project-dare/dare-examples/-/blob/master/tutorial/WP6_MT3D_tutorial.ipynb

The high-level structure of the workflow that we implemented on the DARE platform is shown in Fig. 3.3.

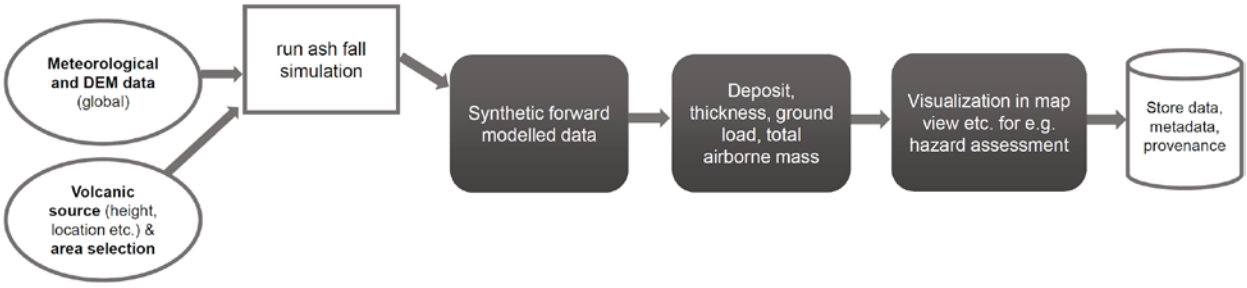


Figure 3.3: *Steps of the workflow for the volcanology test case*.

Using freely available meteorological data (daily or monthly means), digital elevation models (DEM) and parameters describing the volcanic eruption, we run simulations with FALL3DPy, a python port of the original Fortran code FALL3D by [Folch *et al.* 2009] to obtain ash dispersal distributions. The corresponding meteorological data (NOAA or ECMWF/ERA5) as well as the digital elevation models (ETOPO1) are downloaded from external archives. Based on user input (including the location of the volcano, the surrounding area, eruption duration, erupted mass, etc.) ash dispersal is forward modeled and saved in a single NetCDF file which includes all simulated time steps. Finally, in the post-processing step the content of the NetCDF file together with the user input is used to generate visualisations of the individual ash distributions. In Fig. 3.4 we show exemplary time steps for a simulation at Stromboli volcano in the Mediterranean Sea.

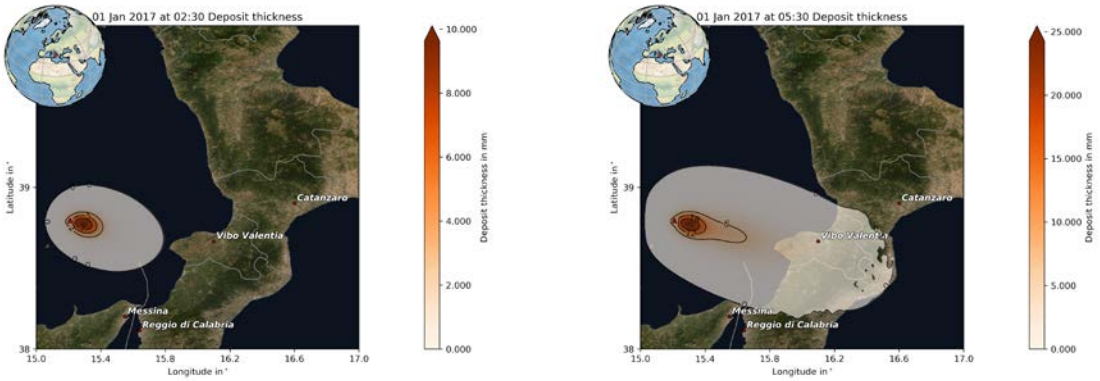


Figure 3.4: *Simulation results showing the deposit thickness in mm 2:30 h (left) and 5:30 h (right) after the eruption started for a scenario at Stromboli volcano on 01. January 2017.*

This workflow has been implemented using the DARE platform and is executable via a Jupyter Notebook interface, as for the seismological test cases. In an initial step a user registers in the DARE platform and receives a token for authorization. For the simulation step, the dockerized version of FALL3DPy is executed through a specific API call that creates in Kubernetes an MPI cluster to run the code. Finally, the other steps of the VC test case (downloading meteorological data, post-processing including plotting of the results, see Fig. 3.4) are described using dispel4py workflows, made by several dispel4py PEs, and each separate executable workflow is launched

through another specific API call. These calls always have the same structure (see description in §3.1.1).

In addition to validation tests, the VC test case has been used by students and other users during the exercises during a training course (24th July 2020 at KIT) about volcanic hazards in the Mediterranean area. All participants were inexperienced with the test case (they only had some background in general volcanology and numerical modeling) and worked with a Jupyter Notebook that was provided via JupyterHub (Fig. 3.5). This allowed the users to perform further simulations from any place after the onsite training event had concluded. This was a necessity given for the elaboration phase of the students to produce case reports of the examined volcanological studies. The deadline of the report was set to early November 2020. The overall feedback of the users during the event was positive, especially with regard to performing such modeling scenarios with an easy-to-use platform infrastructure. Minor suggestions for improvement and general issues of the workflow and execution have been recorded by the tutors of the course and forwarded to the developers of the use case. Furthermore, the training was evaluated by means of a questionnaire given out to students participating during a course on volcanic hazard assessment. The detailed description and evaluation results of the training is summarized in [Constantin 2020], deliverables D6.4 [Magnoni *et al.* 2020b] and D8.5 [Magnoni *et al.* 2020c], and §5.3 evaluation stage 2.

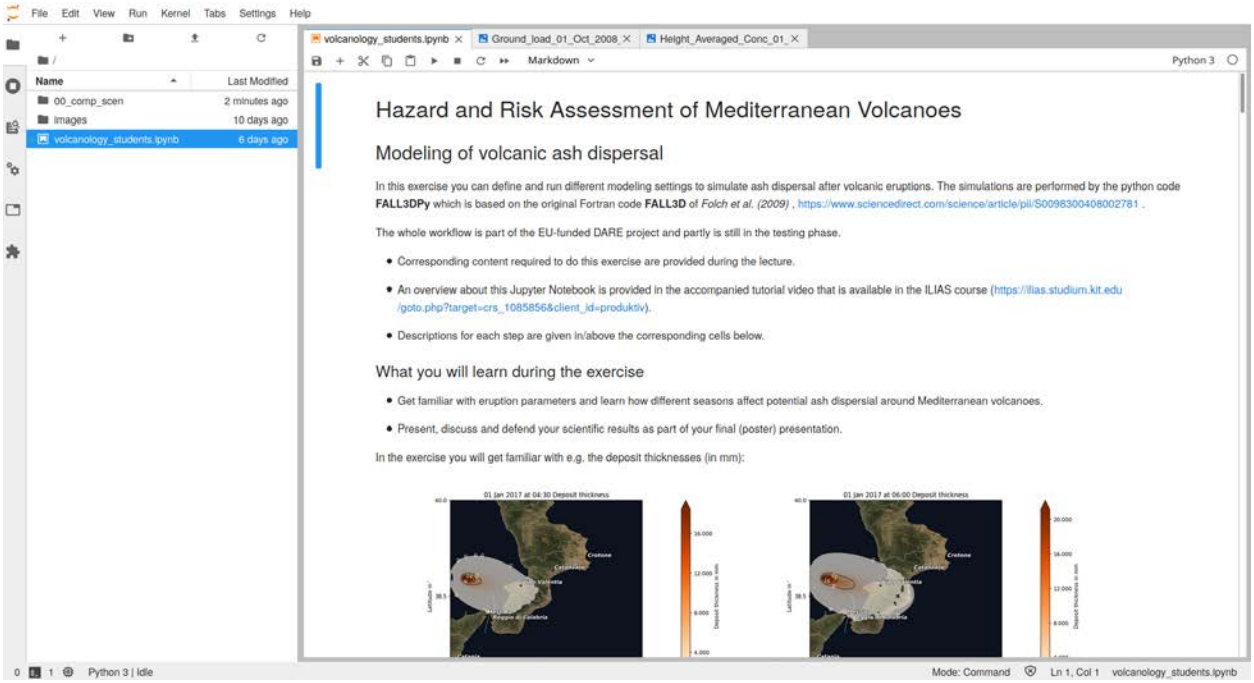


Figure 3.5: *Interface of the JupyterHub in which the Jupyter Notebook of the training event is displayed.*

The VC-DARE approach gives the opportunity to perform fast hazard and risk assessment for not only the volcanological community but also for insurance modelling applications. Ashfall simulation in combination with mapped exposure and vulnerability information may allow for estimations of a monetary impact as secondary effect on surrounding infrastructures. The existing

notebook can hence be further extended using the developer-friendly DARE API to include such routines in the VC workflow.

3.2 Use by climate-impact modellers

Exciting developments have taken place towards the implementation of a generic workflow tool to access and process climate data, aimed at the climate change impact modelling research communities' users. The tool supports climate platform developers to provide on-demand data processing for users using heterogeneous computational platforms, through the deployment of the DARE Platform. One of the major objectives is to provide transparent access to on-demand data processing using easy to use front-end for end users, through interaction with tailored front-ends, such as that provided by climate4impact.eu (Figure 3.6)

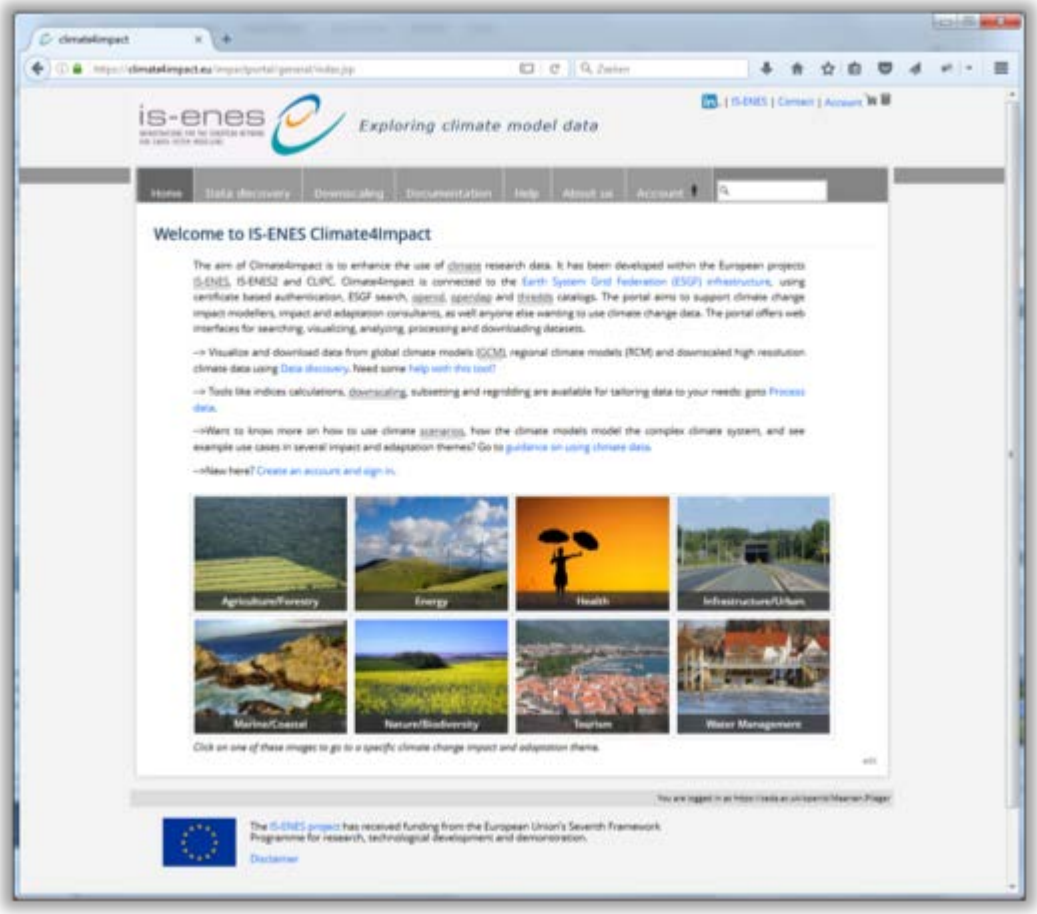


Figure 3.6: *climate4impact.eu Portal, providing guidance and on-demand data processing.*

The design of the first and second prototypes took into account user requirements that have been gathered. Those are described in detail in Deliverable D7.1. To complement this approach, User Stories (described in Deliverable D3.1) have been used to provide information on how to properly plan component development and design application architecture. The second prototype of this

generic climate data analysis workflow tool, which has been implemented in October 2019, is shown in Figure 3.7.

A summary of the second Climate Use cases is outlined below. The reader is referred to Deliverable 7.3 [Pagé 2019] for details on its evaluation during a training Webinar.

Climate Cyclone Use Case

Community involved:

ENES Climate Research Developers

Scientific problem:

Leverage complex analysis tools, such as this specific one: the Tropical and Extratropical Cyclone Tracking.

User requirements:

Handling of complex numerical codes and their configuration; implementation of user customised scientific procedures; shareability and reusability of procedures and their substeps; investigation and reproducibility of results and errors; access to external RI services; monitoring of job execution

DARE approach/solution:

Docker and CWL workflow to manage the execution of a numerical analysis code; parallel execution of parts of the workflow; customised provenance and metadata capturing; consistent shared file system; Jupyter Notebook executed in Jupyter Lab.

Final experience:

Successful implementation and execution of the workflow on the DARE platform using Jupyter Notebooks.

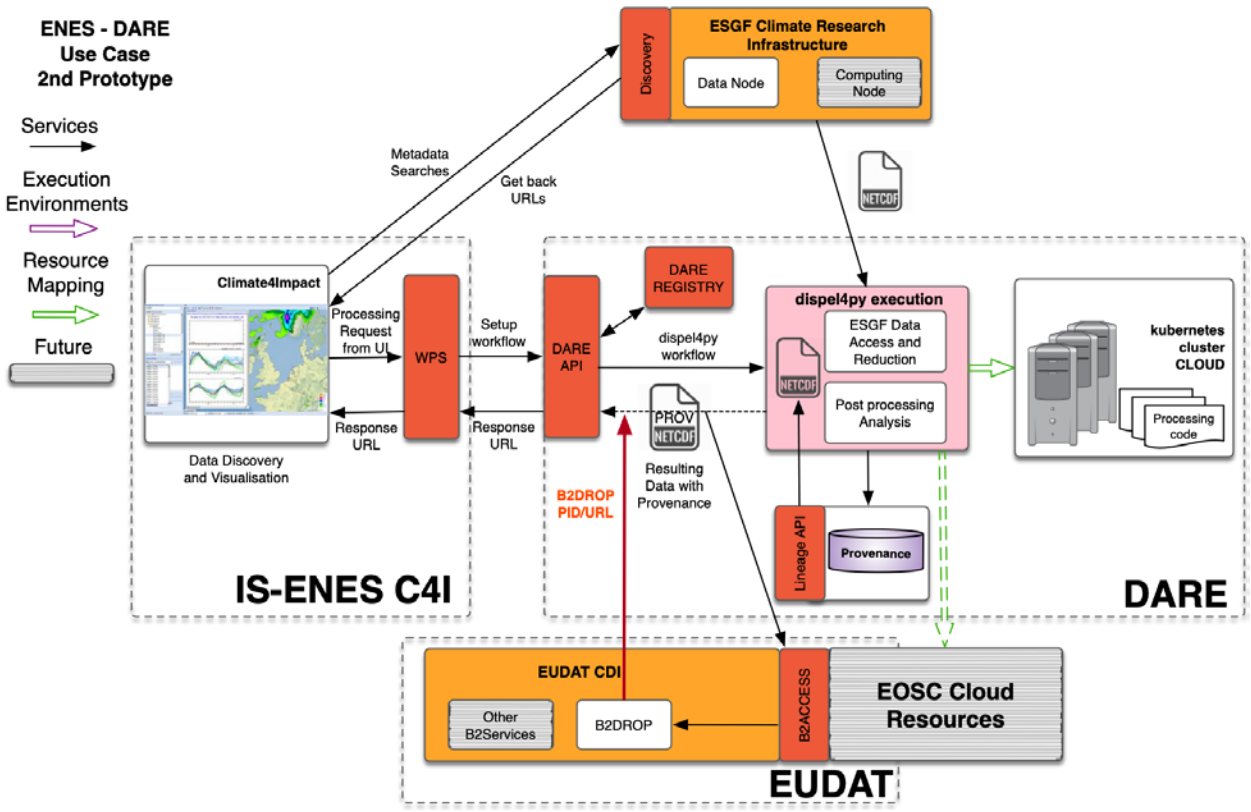


Figure 3.7: *Generic climate data analysis workflow second prototype. Implemented in Oct. 2019.*

This second prototype has been evaluated during the first training event that took place on June 21st, 2019 in Utrecht, Netherlands, under the aegis of IS-ENES. The developers who participated in this evaluation had a very significant interest in this generic workflow approach, notably by requesting access to the DARE Platform API as soon as possible to test it and to begin developing user services using it. The results of this evaluation is detailed in Deliverable D7.3.

The approach here for this generic workflow is to provide software developers using a platform such as C4I a faster way to develop and provide on-demand data processing for users. The goal with those platforms is not to provide an operational climate service like C3S provided by Copernicus. Instead, it is aimed at researchers in other scientific domains, as well as at those doing climate-change impact modelling. Those users, with such a generic workflow approach, will be able to provide their own processing functions (as Python functions added to the icclim[15] open source software), as well as using their own OpenDAP-accessible datasets or use non-standard MIPs from CMIP5 and CMIP6 experiments.

This approach requires an extensive metadata data description, as well as extensive lineage information. This is because when data processing is automated and delegated, metadata standards are necessary to be able to correctly process datasets. Furthermore, it is important for

[15] https://github.com/cerfacs-globc/icclim

users to know how to reproduce the calculations as well as to know exactly which methods and software were used.

During the mid-term review of the DARE project, the idea of designing a new climate-related use case more in line with the seismological use cases emerged. This should attract users as it would offer significantly different tools from those offered by other platforms such as the Copernicus C3S Service[16]. We considered the possibility of running climate-impact models (such as hydrological models), but these proved to be too specific and too complex to achieve given the effort available within WP7. We therefore decided to develop a new use case centered on a method/tool that can track extra-tropical as well as tropical cyclones[17] in climate simulations. This use case is called the Cyclone-Tracking climate use case. Currently, no front-end or online platform offers researchers the opportunity to run on-demand extra-tropical cyclone tracking on climate simulations selected by users. The idea is to provide researchers and users of climate data the possibility to have access to this advanced tool and to provide them with the possibility to generate end-products on-the-fly, such as tracks' density plots (Figure 3.9) or Probability Density Function plots (Figure 3.10). The code to generate those specific plots was developed within the DARE project.

The crucial benefit of DARE is that this can then be easily transferred to a sufficient range of target eInfrastructures, that the resources for the users using this new facility can be sustained.

Given the framework of the DARE Platform the workflow has been developed and implemented by composing the following steps (see overview in Figure 3.8):

1. Selection of input file(s) by the user
2. Start of the execution initiated by the user using the C4I front-end in a Jupyter Notebook
3. Preparation of input parameters using interface user input
4. Upload of the tracking algorithm configuration file using the DARE API
5. Initiation the execution of the algorithm using the DARE API
6. Pre-processing of the input file(s) (taking place on the DARE Platform)
7. Download of the input file(s) by the DARE Platform, accessing the ESGF Climate Research Infrastructure
8. Execution of the tracking algorithm (binary executable) on the input file(s)
9. Post-processing of the output files
10. Generation of end-products (e.g. tracking density map)
11. Upload of end-products and raw output files to the B2DROP user's account
12. Notification to the user and transfer of the output files into the C4I front-end user space[18]

Part of this workflow is embarrassingly parallel, because climate simulations are always independent from one to another. Since ensembles data analysis over multiple climate simulations are almost always needed, steps 6 to 9 can be run independently for several individual

[16] Copernicus Climate Change Service https://climate.copernicus.eu/
[17] https://github.com/cerfacs-globc/cyclone_tracking
[18] This step has not been implemented yet, because C4I 2.0, developed within the H2020-IS-ENES3 project, is still in alpha phase at this time.

climate simulations in parallel. This parallel execution in the case of the user selection of multiple scenarios has been implemented in the latest version of the workflow.

The cyclone tracking workflow has been implemented using the Common Workflow Language[19] (CWL). This has been possible because support for CWL has been added to the DARE platform, including the automated provenance generation. CWL was more suitable than dispel4py for the cyclone tracking workflow because it is a sequential workflow. The automated provenance is generating a lot of information, and the main workflow sequence can be summarised using this information as shown in Figure 3.11.

An important point is that all developed source codes are available in several git repositories, including the tracking software itself along with its end users' products generation:

- CWL Workflow Cyclone Tracking source code: https://gitlab.com/project-dare/wp7_cyclone-tracking/-/tree/cwl
- DARE Cyclone Tracking Container (within the DARE Platform repository): https://gitlab.com/project-dare/dare-platform/-/tree/master/containers/exec-context-cyclone/cwl
- DARE Cyclone Tracking Training Tutorial Jupyter Notebook (within the DARE Platform Examples repository): https://gitlab.com/project-dare/dare-examples/-/tree/master/wp7/tutorial
- Cyclone Tracking Software (Fortran code): https://github.com/cerfacs-globc/cyclone_tracking
- Cyclone Tracking Software End User Products https://github.com/cerfacs-globc/cyclone_tracking_products

[19] Common Workflow Language (CWL): https://www.commonwl.org

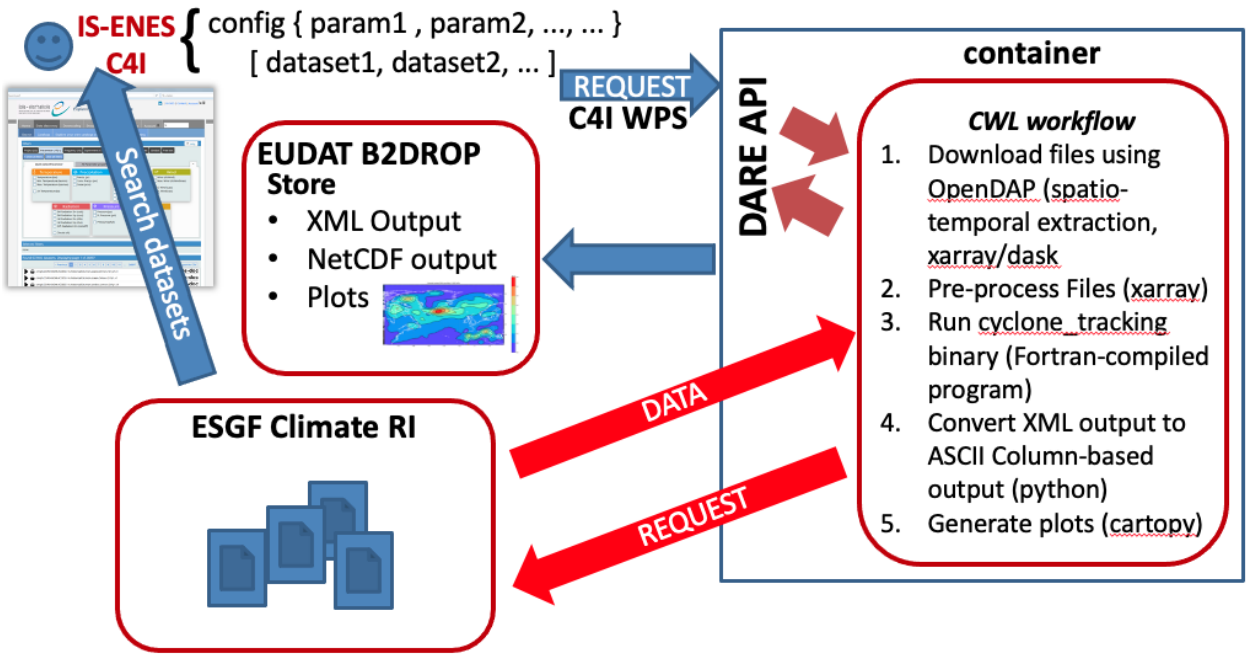


Figure 3.8: *Cyclone-Tracking Climate Use Case Workflow and Infrastructures integration.*

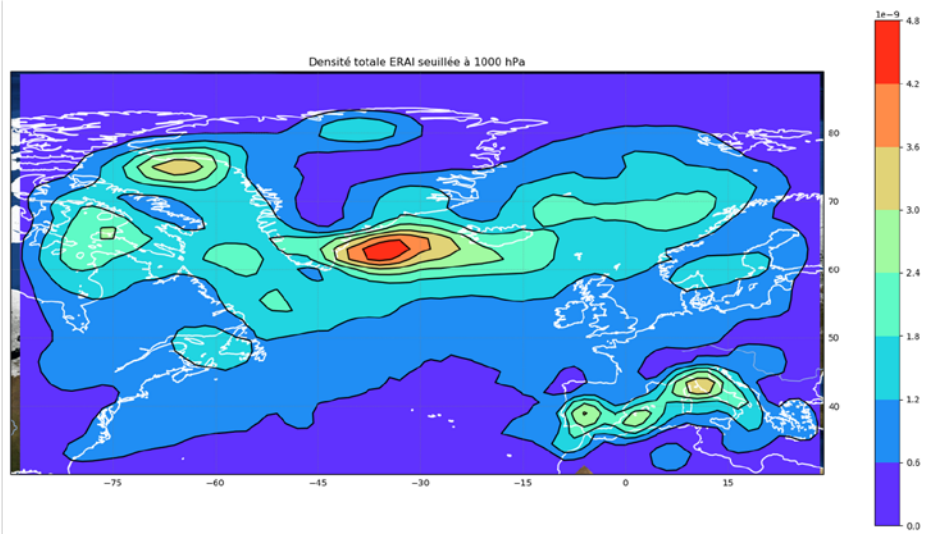


Figure 3.9: *Extra-Tropical Cyclones track density example plot.*

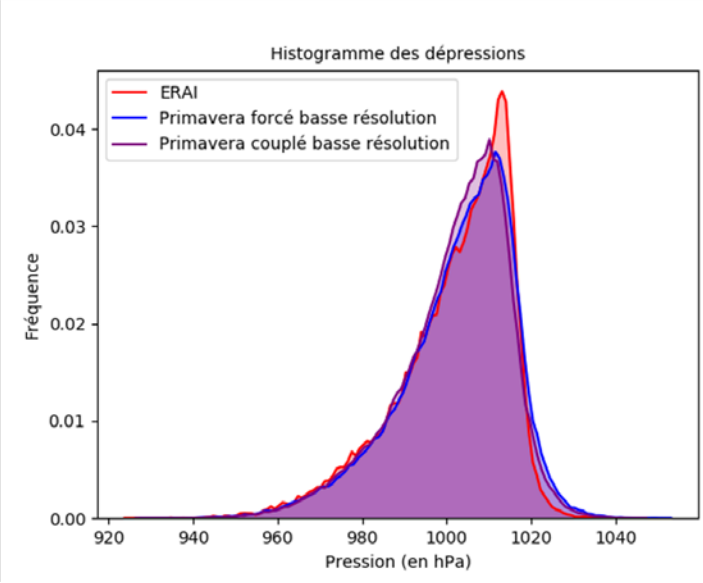


Figure 3.10: *Extra-Tropical Cyclones probability density functions example plot.*

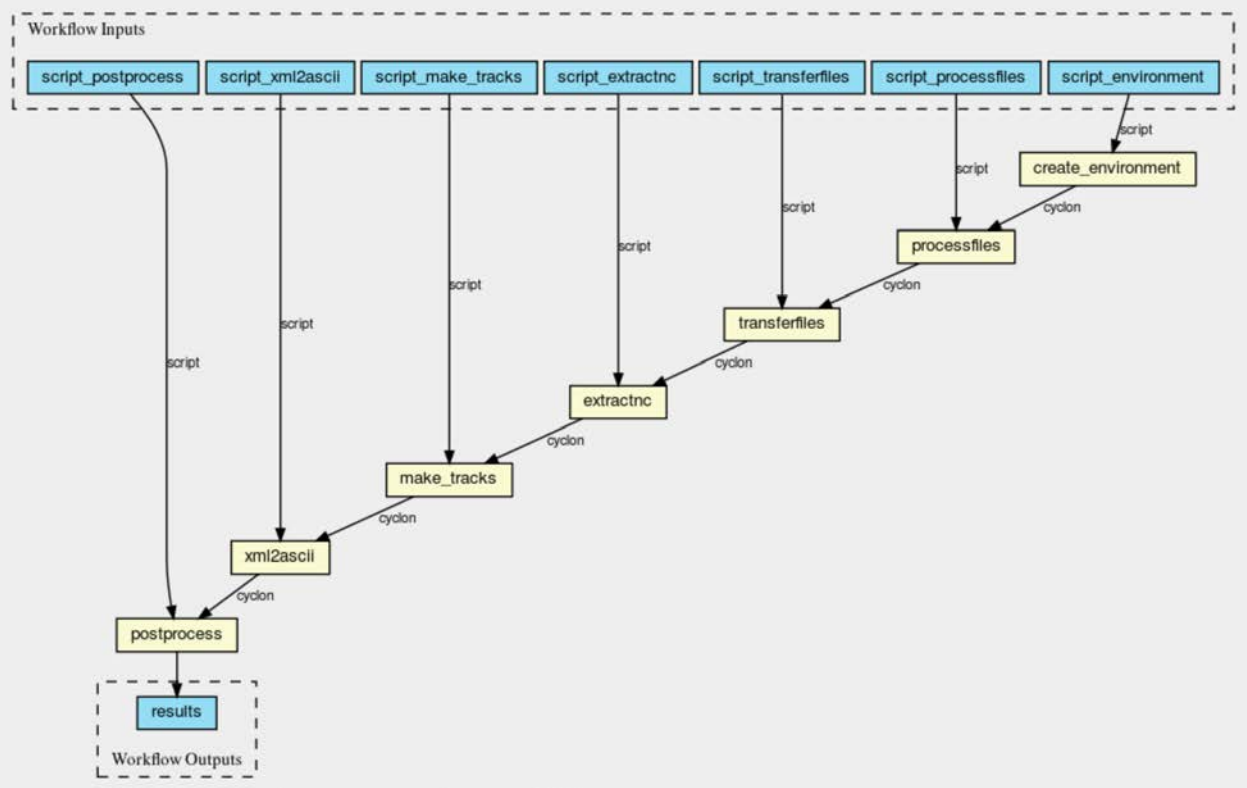


Figure 3.11: *Extra-tropical Cyclones CWL workflow automated provenance generation.*

3.3 Use for development

There is a continuity between users and developers, in the sense that some application specialists use DARE workflows developed by others, but still choose them, parameterise them and want to

make adjustments to them. Others focus on developing new research methods, materialising them as software libraries, simulation and analysis tools and integrated workflows. There is no boundary between them, rather a continuum. Indeed, individuals move in this continuum as their research requires. Consequently, that continuum needs consistent support.

The new interfaces that we are building on DARE provide a fluent path from prototyping to production. Applications are not locked to platforms but can be moved to suitable new platforms without human intervention and with the encoded method's semantics unchanged.

The complex development and debugging requirements encountered in the latest steps of co-design and co-development with the seismologists have provoked refinement of this development playground and will lead to requirements for self-sufficiency in conducting further refinements and production use. The quality of this support has been tested by a member of the KIT team using it for an unanticipated use case. He developed, exploiting DARE's framework, a method for using a computational model of mass transport via ash and other ejecta from a volcanic eruption (§3.1.2). This production has been used by staff and MSc students in the summer of 2020 during a field course on Stromboli with its recent volcanic activity.

The DARE platform acts as an intermediary between users' applications and the underlying computing resources, making use of technologies including:

- Container Orchestration -- Kubernetes
- Distributed Engineering -- MPI cluster
- Workflows' technologies -- dispel4py, CWL, Registry, S-Prov

The DARE API (see Figure 3.12), allows users and developers to register dispel4py workflows (applications) to the registry, such as the RA seismology test case from WP6. Once a workflow has been registered, it can be submitted for an execution, and the DARE API will automatically deploy all the necessary environments on demand. It also facilitates monitoring the execution status of a workflow in the platform.

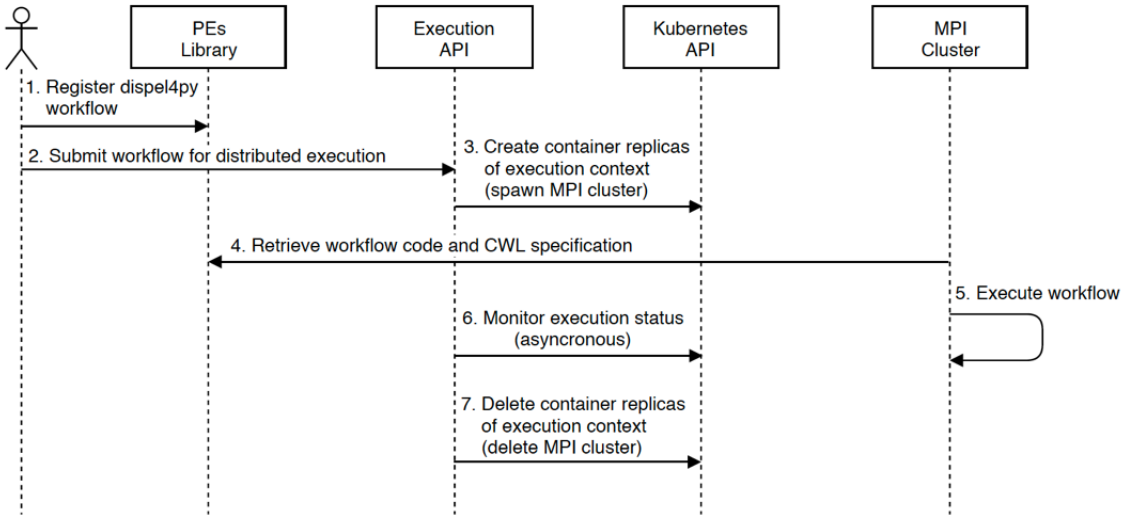


Figure 3.12: *The execution of a dispel4py workflow using the DARE API. This figure shows all the underlying steps as well as the entities that are involved*.

In order to facilitate the development and testing of dispel4py workflows, a docker container has been made available to users, to allow them to develop workflows locally (on their laptops or local hosts), which mimics the configuration of the DARE platform. The idea is that users and developers can have a "DARE environment" locally, which has the same libraries, python versions, and so on that the DARE platform has. In the case that a library is missing, or another version for a particular library is required for implementing a use case, these could be installed locally in the docker container, and later they need to be specified to the DARE API at the time of submitting the workflow by using a requirement file, as shows the following API call:

```
dm.exec_d4p(impl_id=impl_id,pckg="mysplitmerge_pckg",workspace_id=workspace_id,pe_name="mySplitMerge", reqs='https://gitlab.com/project-dare/dare-api/raw/master/examples/jupyter/requirements.txt', token=F.auth(), creds=creds, n_nodes=6, no_processes=6, iterations=1)
```

Furthermore, a playground endpoint has been recently provided to users with more facilities for debugging their applications and workflows inside the DARE platform. The playground simulates a terminal allowing users to provide a command and see immediately the output results, giving more direct control to users. Below, we show the previous example submitted in a "debug" mode:

```
F.debug_d4p(impl_id=impl_id, pckg="mysplitmerge_pckg", workspace_id=workspace_id,
pe_name="mySplitMerge", token=F.auth(), creds=creds, no_processes=6, iterations=1,
reqs='https://gitlab.com/project-dare/dare-api/raw/master/examples/jupyter/requirements.txt')
```

The current methodology for users to develop new workflows is to use the local docker container, and then later test them on the DARE platform using the "playground mode". When they are satisfied with their validation they submit them to the platform in the "normal" mode.

It is worth noting that the DARE API allows users not only to register and submit dispel4py workflows (with or without additional requirements), but also to monitor and download the results, files and log files associated with a workflow execution.

An example of how the DARE API can be used by users and developers can be found at the following notebook: https://gitlab.com/project-dare/dare-api/blob/master/examples/mySplitMerge/mySplitMerge%20workflow.ipynb,
in which a "mySplitMerge" dispel4py workflow is registered, submitted, executed (with and without additional requirements) debugged, and monitored using the DARE API.

In addition to dispel4py, the DARE platform has been extended with the support for CWL workflows management and execution. Similar to the dispel4py, DARE can register CWL workflows, and the execution environments (dockers) they need to run. These can be retrieved by name and version. In Table 3.1 we describe the high-level functions enabling users and administrators to control the registration and execution of this type of workflows/environments. Details on the implementation of this new capability is illustrated in D3.6 (DARE API II).

Below we provide a list of high-level management and execution functions that can be used programmatically by the users of the DARE platform. These enable access to the platform, registration and retrieval of workflows and environments, and offer basic monitoring and data management tasks. Readers are referred to https://project-dare.gitlab.io/dare-platform/api/ for complete and up-to-date information about the DARE API and helper functions. Those wishing to install or deploy DARE should consult https://project-dare.gitlab.io/dare-platform/installation/. All DARE technical documentation can be reached at https://project-dare.gitlab.io/dare-platform.

Table 3.1: Functions currently provided via the DARE API.

DARE platform API function	Description
`login(username, password, hostname)`	Get dispel4py registry credentials by logging in
`create_folders(hostname, token)`	Create the working environment
`get_auth_header(token)`	Return the authentication header
`get_workspace(name, creds)`	Get a workspace URL by name
`create_workspace(clone, name, desc, creds)`	Create a workspace using dispel4py registry API
`create_pe(desc, name, conn, pckg, workspace, clone, peimpls, creds)`	Create ProcessingElement / dispel4py workflow using d4p registry API

create_peimpl(desc, code, parent_sig, pckg, name, workspace, clone, creds)	Create ProcessingElement/ Workflow Implementation using d4p registry API
auth(length=10)	Generate user "access token" / Simulate user login
submit_d4p(impl_id, pckg, workspace_id, pe_name, n_nodes, token, creds, reqs=None, **kw)	Spawn MPI cluster and run dispel4py workflow
debug_d4p(impl_id, pckg,workspace_id, pe_name, token, creds, reqs=None, output_filename="output.txt", **kw)	Debug a dispel4py workflow in “playground mode”
exec_command(hostname, token, command, output_filename="output.txt")	Allows for running a command in “playground mode”.
register_docker(docker_name, docker_tag, docker_url, script_names, docker_folder)	Allow a user to define a container (docker) environment and registers it to the DARE platform
download_docker(docker_name, docker_tag, local_path) provide_docker_image_url(docker_name, docker_tag, docker_url)	Allow an administrator to download the container files. He/She would validate them, build the image and update the docker entry so as to provide a url to a public image repository
update_docker(..)	Update the tag or name of a container
add_script_to_existing_docker(..) edit_script_in_existing_docker(..) delete_script_in_docker(..)	Allow the management of the executable scripts hosted in a target container
register_cwl(cwl_params, docker_params, register_docker)	Register CWL workflow and associate it with the docker environment

exec_cwl(workflow_name, workflow_version)	Execute a CWL workflow previously registered within the associated container.
upload(token, path, local_path, creds)	Upload data into a working environment
myfiles(token, creds) and files_pretty_print(_json)	List the uploaded files.....
download(path, creds, local_path)	Downloads a file using exec-api filesystem reference.
delete_workspace(name, creds)	Deleted a workspace
submit_specfem(n_nodes, data_url, token, creds)	Spawn an MPI cluster and run a specfem workflow
my_pods(token, creds)	Returns user created pod properties (name and status)
send2drop(token, path, creds)	Uploads a file from the exec-api shared filesystem to the project-dare b2drop account in order to get a shareable link for a single file
pod_pretty_print(_json)	Monitoring container status
monitor(creds): Monitor	Monitor a dispel4py workflow run

3.4 Summary and conclusions

Throughout the DARE project interaction between developers of the platform and the two user communities has been very intensive. It has led to increased understanding of requirements, recognition of challenges that are relevant to many communities and to many e-Infrastructure builders and providers. Progress with both user communities has been significant, providing evidence of the potential value of the DARE platform and approach. The seismic methods are developing in complexity and computational demands with a prospect of wider use throughout EPOS and beyond. The climate-impact modelling has taken a similar path as it has extended its scope to include cyclone-tracking. The support for R&D undertaken by research engineers, provided by the DARE playground and facilitated through the Python library is critical to self-sufficiency, innovation stimulus and sustainability.

Inevitably, there are issues that will continue as DARE and its user communities transition to a sustainable mix of innovation and production use. Illustrative examples here are revisited in §5 and §6.

1. The application experts need to control development paths and deal rapidly and effectively with emerging opportunities or limitations with their own teams, standard widely supported

software and modest amounts of support from computing experts external to their community.
2. The computational and data management resources change, differ between institutions and users' entitlement to use them also changes, e.g., as a result of a competitive award, technology advances or institutional investment. Research communities, groups and individuals need to exploit these variations without undue disruption of their methods, working practices and knowledge infrastructure.
3. The PlayGround developments to support advanced developers and innovators are proving their value already. They need further development, improving flexibility and usability for those not already embedded in the DARE platform-development team.
4. Induction into the world of sophisticated data use and computation must continue and be accelerated.

All aspects of these emerging requirements influence the architecture implementation reported in the next section, §4. A broader issue is their link with sustainability considered in §5. Research engineers and platform builders must use minimal new software and build on widely used software and standards - see §5.2. However, significant new software technology is developed by DARE. It will need to be supported beyond DARE as part of DARE's sustainability plan. That support will include software maintenance and expert advice. Addressing the extension of user requirements as summarised above will increase the potential user community thereby expanding the number of organisations contributing to that support. For example, if more of the workflow systems used by EPOS are accommodated, if their model of catalogues interworks with the DKB and if provenance is collected and delivered to P4 tools (§4.3) from all relevant software, the investment in support will come from the whole EPOS community. A similar argument can be made for IS-ENES and for similar projects, e.g., those within the ENVRI cluster of environmental research infrastructures.

4. Architecture Implementation

In this section we review the current DARE platform's architecture, note the progress since D2.1 [Atkinson *et al.* 2018] and report on developments addressing the architectural goals in §2 and to meet the users' requirements in §3. The current implementation is reported in [Klampanos *et al.* 2020, Klampanos *et al.* 2019 and Spinuso *et al.* 2019]. The DARE platform, its APIs, examples and tutorials are documented in a dedicated "microsite" on GitLab[20]. The key points are summarised here; readers are referred to those papers for more detail and to the platform's dedicated website on GitLab for up-to-date documentation.

As introduced with Figure 1.1 the DARE platform has three technological pillars:

1. *Workflows-as-a-Service* (**WaaS**) help communities develop and use formalisations of their methods. For the supported scripting notations and workflow languages it enables authoring, debugging, validation and optimised productive use of methods. It selects appropriate targets for enactment, prepares them, e.g., by installing the required container, updating its configuration and initiating processing on a network of interconnected distributed processes. This is significantly more integrated and therefore easier to use for all stages of method development than the system reported in D2.1 and the accommodated formalisations now include CWL and Jupyter notebooks. Its technical details are presented in §4.1.
2. *DARE Knowledge Base* (**DKB**), has three roles:
    a. *Human communication*, a place where practitioners in any role can leave any information they wish for themselves or others.
    b. *Software communication*, a place where software can leave information for its own or other systems' future use.
    c. *Human-system communication*, a place where humans leave information for software to use and where software leaves information for humans; this should improve human-system relationships, improve understanding and enable responsible control of quality of methods and results.

    This is based on two independent catalogues: the Processing-Element (PE) registry [Klampanos *et al.* 2015] for the components workflows are built from and the Data Catalogue, as D2.1 reported. These have been developed further and used more extensively in the current release of the platform. The current development brings these into an integrating, flexible, extensible and incrementally adopted common framework - see §4.2.
3. *Protected Pervasive Persistent Provenance* (**P4**) and the tools and interactions it supports. This is intimately connected with the DKB, as it is a major repository of and source of information about user and system behaviour. However, it warrants separate identification because of the crucial role it has underpinning the quality of science and evidence. By delivering *reproducibility* it has a stand-alone role that may be utilised by many research communities. By supporting provenance-driven tools it significantly improves understanding, addressing the second architectural goal in §2. These tools also reduce

[20] https://project-dare.gitlab.io/dare-platform/

labour-intensive data administration by automating selection and providing batch operations, thereby improving productivity. It is possible to mine information from the growing persistence repository, for example:

a. The costs and resources used for running processes and complete methods.
b. The frequency with which each category of data is used.
c. The bottle-necks and common pitfalls encountered when performing established or required procedures.
d. The parts of data collections or model-parameter spaces that have been explored; perhaps alerting researchers to critical omissions.
e. The errors users exploiting or authoring methods are making repeatedly; implying that changes should be made to help those users.

These uses of the growing wealth of provenance data have great potential to improve the science and the methods used pursuing that science. This potential is just becoming available, but there are also more mundane steps needed. See §4.3 for details.

These subsystems are presented below. They need to maintain consistency with each other as detailed in D2.1 §8.4 [Atkinson *et al.* 2018]. Their current implementation is shown in Figure 4.1.

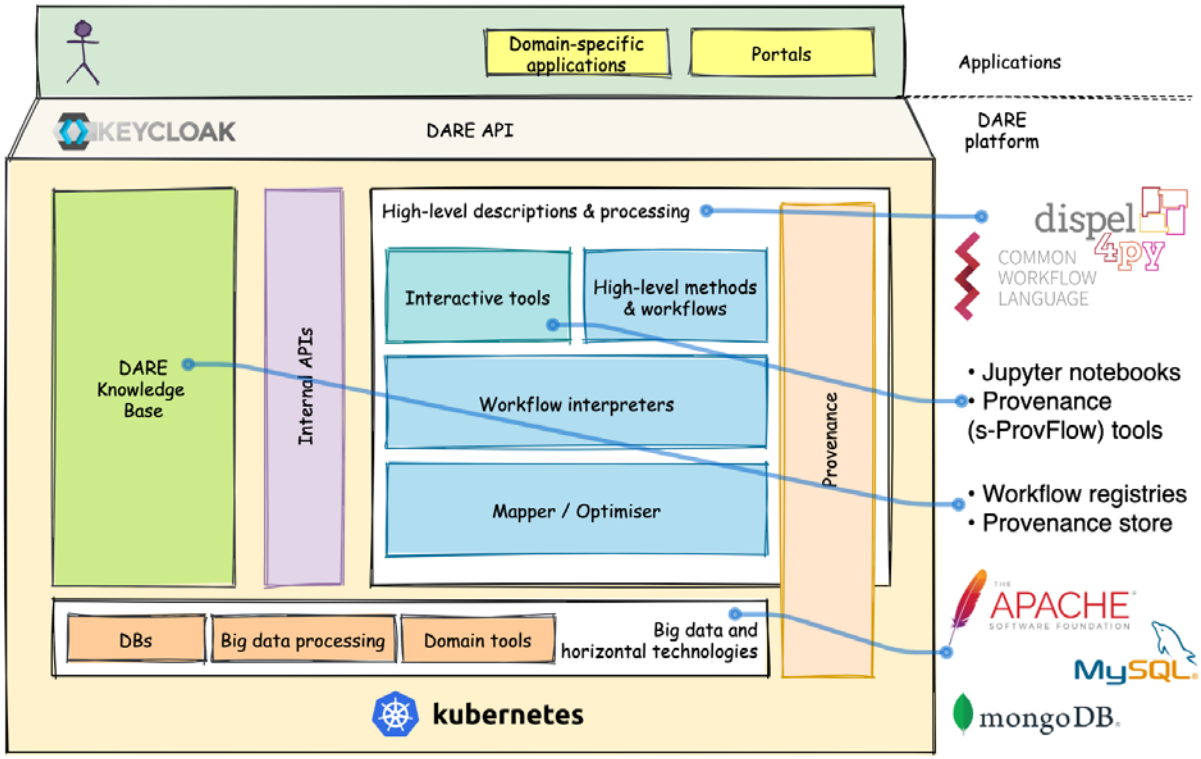


Figure 4.1: *The structure and main components of the current DARE platform*

4.1 Workflows as a Service (WaaS)

Workflows-as-a-Service extend the functionality of typical Workflow Management Systems (WMS) to automate set up and management of the computing facilities the WMS needs for the requested tasks. A WMS will support all of the phases of workflow development and the repeated enactment of the resulting workflows. A WaaS identifies the needs of a requested enactment and locates e-Infrastructure resources to meet those needs. It then prepares them using software deployment, configuration, interconnection and orchestration mechanisms, monitors the enactment, organises data movement and storage, and delivers results and enactment records to the users [Filgueira *et al.* 2016, Rodriuez & Buyya 2018, Liang *et al.* 2020]. The DARE WaaS fully supports the dispel4py WMS and CWL specifications targeted at multi-core shared-memory environments.

4.1.1 Concepts

The DARE platform provides all the necessary tools to research developers for them to execute various workflow development, deployment and production-use tasks. Research developers write code in Python, utilising a workflow language library (dispel4py[21]), which allows them to define fine-grained streaming workflows of arbitrary complexity. Conceptually, a workflow is a graph that connects well-defined units of processing functionality - processing elements (**PE**s). More information on dispel4py can be found in [Filgueira *et al.* 2017]. Each PE defines a Python method that describes the process to be executed. An experiment is logically divided into multiple PEs, connected by directional arcs in the aforementioned graph. Data units flow along these arcs, from outputs on a source PE to inputs on one or more destination PEs; order is preserved. This enables dispel4py to represent abstractly a number of parallelisation patterns.

The main concepts of dispel4py are managed via a dispel4py Information Registry (**Registry**)[22], which is part of the DARE platform's knowledge base (§4.2). The Registry is used in order to efficiently store and retrieve workflows and enable workflow reusability. Users can create their own workspaces and register the Processing Elements (PEs) that they intend to execute or share. The Registry provides an API that enables creating, updating and deleting workspaces and PEs. Before a workflow can be executed, it needs to be registered in the Registry.

In addition to dispel4py, the DARE platform supports workflows expressed in the CWL workflow langage[23]. CWL defines an open standard for describing workflows across different architectures and software or hardware environments. CWL is a process-based and more coarse-grained workflow system than dispel4py. CWL workflows typically define pipelines of execution where at the end of each step files are being created, some of which to be used as input in the following step, etc. In this respect, CWL is largely complementary to dispel4py, therefore potentially increasing the impact and usefulness of the platform. Similar to the dispel4py information registry,

[21] https://gitlab.com/project-dare/dispel4py
[22] https://zenodo.org/record/3361395#.Xg22gy2Q0Wo
[23] https://www.commonwl.org

the platform comes with a CWL workflows registry[24], which is also a component of the overall DARE knowledgebase. Due to CWL being dependent on external executables being present, the CWL registry is able to also register execution environments as docker images and associate them with CWL workflows. Once a CWL workflow has been named and registered, it becomes invokable via the DARE execution API.

4.1.2 User Instructions

In order to execute a workflow, users need first to create or reuse a workspace and inside it register the necessary PEs in the corresponding Registry (dispel4py or CWL). PEs that are stored in a Registry can be reused in future experiments/executions by name.

The DARE platform provides a test environment, as mentioned in §3.3, in order to execute workflows with immediate diagnostic information and direct control with the DARE platform's computational environment accurately emulated. This accelerates development and substantially improves research developers' powers to investigate issues. The relevant component ("*playground*") provides two functionalities. Firstly, a user can simulate a workflow execution and immediately check the logs and outputs of the execution. The second functionality provides the environment to execute any command, simulating a terminal. More details are available in the corresponding GitLab[25] repository.

When the workflow is ready for execution, the user can execute it via the official API endpoint of the DARE platform. While a workflow is being executed, the user can monitor the containers that execute the workflow, through the API endpoint provided for that purpose. Users have their own directory where the files are organised per execution (test and production executions use different directories). The DARE platform through its API provides functions to list the folders and files in those directories as well as to download any produced files (see Table 3.1).

4.1.3 Workflow Execution

The DARE platform provides workflow execution as a service via a RESTful API. In order to use the provided services, the first step is to register the workflows in the Registry. Subsequently, users can execute the workflows using the registered name. Users can configure the execution parameters, for example the number of nodes required for the execution. Based on the requested number of nodes, the DARE platform generates an appropriate number of MPI containers to execute the requested workflow.

The DARE platform contains a Shared File System that the MPI containers can access to store or read files. Each run is stored in a different directory. When a workflow requires additional Python libraries, a virtual environment is generated inside the respective directory. After the execution, all the output files are also collected in the run directory.

[24] https://gitlab.com/project-dare/workflow-registry
[25] https://gitlab.com/project-dare/playground

Through the platform, a user can obtain provenance information in order to track what experiments have been executed, as well as to obtain the input and output data. The DARE platform provides a user interface on top of the provenance API and storage, where the user can view the executions and the data produced in the platform (§4.3).

4.1.4 Future Work - Optimisations

In the preceding sections, we have summarised the current state of the DARE platform. In the next phase of the DARE project we will improve the use of the shared file system by separating the executions of a user based on the respective experiments (see §4.2). Users should not need to specify platform- or implementation-specific details, such as the number of processes to be utilised. These matters will be investigated as part of the workflow-optimisation effort. They are important for method portability and durability. They will also reduce the distractions of underlying detail, initially for application-domain users and eventually for research developers when they trust the automation.

Another aspect of our work on optimisations is to enable dynamic deployment of dispel4py workflows. Currently, during enactment and prior to execution each PE is translated into one or more PE instances (an executable copy of a PE with the input and output ports running in a process as a node in the data-streaming graph), depending on the number of nodes to be utilised, and once assigned a PE instance to a process, it can not be changed during the execution. The main inconvenience of this static deployment, is that if a PE during its execution needs to be mapped to more processes (e.g. the data-rate consumed/produced by a PE has increased more than expected) or to fewer (e.g. a PE is just executed in few occasions) processes, we can not do anything about it apart from manually intervening to stop the current execution and re-assign the process to PEs either manually or by applying an assignment algorithm based on previous executions. This is clearly costly in human effort and computational resources compared with preemptive adaptation.

By enabling dynamic deployment and enactment of dispel4py workflows, PE instances will not be locked to specific processes, scheduling PE instances on-the-fly, meaning that if a PE needs more or less “resources”, it will dynamically up-scale or down-scale, rebalancing automatically the graph, without stopping the workflow execution. To do so, we are planning to implement the *work-stealing scheduling strategy*[26] [Frigo *et al.* 1998, Mattheis *et al.* 2012]. A mechanism to provide load balancing in case of dynamic workloads, which offers several benefits (e.g. data locality, scalability) in terms of efficiency and usability. It has been employed in a number of frameworks for parallel programming, e.g. as Intel Threading Building Blocks (Intel TBB[27]) or GrPPI [Dolz *et al.* 2018], and has found a variety of applications, from simple divide-and-conquer algorithms to more complex stream processing applications [Anselemi & Gaujal 2009, Navarro *et al.* 2009]. This work will create a new dispel4py enactment mapping, based on ZeroMQ message queue[28],

[26] https://en.wikipedia.org/wiki/Work_stealing

[27] Intel TBB https://software.intel.com/en-us/blogs/2018/08/16/the-work-isolation-functionality-in-intel-threading-building-blocks-intel-tbb

[28] ZeroMQ https://zeromq.org/

which will implement a runtime work-stealing scheduler to execute the different PEs respecting dependencies and balancing the parallel workload. The concept of *affinity* will be exploited in this new mapping to ensure locality-aware scheduling.

4.2 The DARE Knowledge Base (DKB)

The DARE Knowledge Base (**DKB**) is an integration and packaging of the information repositories that developers and application domain experts use. As introduced above, the DKB is there to help communities cope with immensely complex, diverse and evolving information that is the context for their research. For research to be authoritative and relevant it has to be embedded in established global knowledge based on years of collaborative work [Edwards 2013]. This becomes more challenging as the knowledge infrastructures of different disciplines collide in the cauldron of multi-faceted, multi-disciplinary global and societal challenges. Urgent problems and scientific explorations require agility, the ability to make quick changes to the parts of the information space where innovation is taking place. The DKB must combine those stable contexts, often formalised by global agreements and established information services, and the dynamic intellectual workbenches where new ideas are tested and new methods created.

The DKB's design is described in [Atkinson *et al.* 2020a, Atkinson *et al.* 2019]. There will be one instance of the DKB per DARE platform deployment. The current developments can be found in the GitLab repository for the DKB open-source software[29] and in those for the other information sharing facilities, described in §4.2.4. A user manual (**URM**) [Levray 2020] provides up-to-date definitions of the DKB Python functions intended for use by all users, services and tools with introductory examples. More abstract structures with their functions may be built using this library. The user manual also has installation instructions for those who wish to use it as a stand alone service or with a selection of services.

4.2.1 DKB requirements

The DKB should deliver the following benefits compared with developing and using individual information-sharing subsystems directly:

1. *Easier for application-focused users to extend and use* - they should be able to directly create, change and use their shared information to help them organise their production, collaboration and innovation work.
2. *Provide incremental support for adoption* so that current developed practices, catalogues and information-sharing frameworks can sustainably co-exist with DKB use[30]. Users and developers will decide when they use existing information services directly and when they or their software works via the DKB. In the latter case, they can make local changes in one place to accommodate changes in the used service or to add functionality.

[29] https://gitlab.com/project-dare/dare_kb.

[30] In DARE these include the Registry, the Data Catalogue and the Kubernetes catalogues. The relationship with P4 is complex and under development. Each user domain also has its established archival practices and data-exchange standards.

3. *Progressively raising abstraction* improves understanding, portability and durability. It protects those who use it from unwanted external changes and distracting technical detail. This also enables DKB implementers to re-engineer their mapping to services to accommodate and exploit external changes and to address issues identified by users, with less disruption to ongoing production and development work.
4. *Boundary crossing*, experts from different disciplines need to collaborate to develop innovations and advances in complex and challenging campaigns. The DKB should deliver the support for CSCW *boundary objects* where their work overlaps, while delivering as much freedom as possible to experts where it doesn't.
5. *Commonalities* between and within communities can be discovered and exploited.

The DKB has to comply with the following constraints:

1. It must be *open ended* because the paths researchers, developers and communities will take are unpredictable - this mirrors Linked Open Data (LOD) represented by RDF[31]. However, within the DARE KB we underpin this freedom with a consistent foundation and accelerate productive use with a conceptual library, intending to get the best of both worlds.
2. It must be *directly controllable and manageable* by application communities as this makes them more self-sufficient, with agility (speed of response to needs and opportunities) and less dependent on technical experts. This means that the formal underpinnings and implementations must be well hidden.
3. It must, nevertheless, facilitate *sustained productive collaboration* between application and technical experts, as identified by Trani for the EPOS RI [Trani *et al.* 2018]. This means that the formal underpinnings and implementations must be understood by experts in their use and in the aspects of technology they are used to describe.
4. It *cannot require a 'green field' site*; it must operate in conjunction with existing information stores, operational software and established professional practices.
5. Constraints 2 and 4 imply that it cannot take *full* responsibility for correctness and consistency. It should do this for information entirely assembled via its functions. It may also support methods for verifying consistency that developers and others may employ.
6. It must *persistently retain information* entrusted to it, while respecting the structure and dynamics of the organisations that contribute to and use the DARE platforms. This means reflecting the recursive pattern of commonalities and releases while retaining all local work accomplished in the context of earlier releases. This requires methods for preserving the work so far, installing the new releases and adapting to any changes as it restores local work. Discrepancies encountered during these procedures must be referred to relevant users.
7. It must be *sufficiently fast* to meet production and concurrency requirements.
8. It must be *sufficiently protected* to prevent tampering, leakage of private or confidential information, and loss by accidents and user errors.

[31] Resource Description Framework (RDF) https://www.w3.org/RDF/

Within these goals and constraints, the DKB was co-designed and co-developed with DARE platform and application developers to establish its immediate relevance and path to adoption.

4.2.2 DKB roles

The term 'DKB' in this section refers to the composition of the *logically* centralised integrator and the co-existing other information services. Four *low-level roles* are supported:

1. Any user[32] may *enter* information into the DKB for their own or other users' future use.
2. Any user may *enter* information into the DKB for software's future use.
3. Software may *enter* information into the DKB so that it informs users - with appropriate adaptations for the target recipients if at all possible.
4. Software may *enter* information into the DKB for its own or other software's future use.

We therefore introduce the generic term "***Instance***" for any item of information considered as a unit by the agent that enters it. As for other data, we then need to manage the lifetime of each Instance. Some of the transitions envisaged during such a lifetime are shown in Figure 4.2.

[32] Acting directly, e.g., using a Python function in an interactive Jupyter session, or interactively via intermediate software.

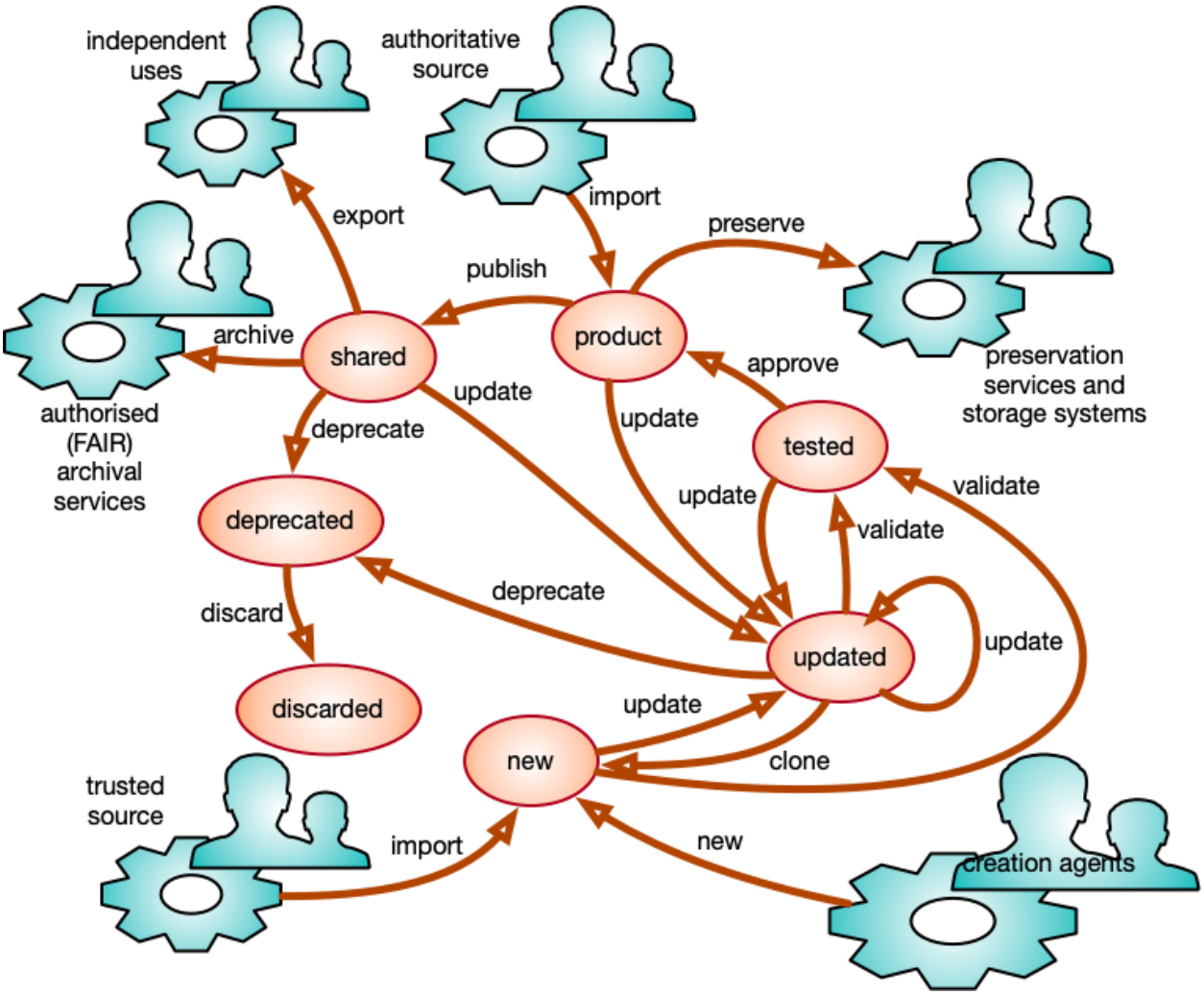


Figure 4.2: State transitions during the life of a DKB Instance. Not all are shown. The external initiation of an action causing a transition is only shown for the initial creation, but transitions are normally initiated and controlled by external agents. Research developers perform much of their work in Git-managed spaces, so many of the imports will be from such spaces.

The implementation of these functions is an appropriate combination of the following mechanisms.

1. *Delegation*, the direct presentation of a function of some information service or a wrapped request for that service, e.g., to handle simple format changes.
2. *Local action*, the representation of the relevant information entirely within the DKB service with changes to that information required to implement the action. Normally, actions implemented within the DKB will be *atomic*, i.e., the state change is complete or there is no change.
3. *Harvesting*, the acquisition of required information from an external source that is then held locally. This will be a snapshot. The external service may later change the source

data[33] making the snapshot out of date. This may be semantically significant, only a user can decide which value is now the required one.

4. *Querying*, sending queries to one or multiple sources and then combining and transforming the results returned.
5. *Caching*, conducting queries, as in harvesting and querying with a policy for discards from the cache, to limit local resource use or to reduce the risk using out-of-date results.
6. *Proxying*, the DKB represents aspects of an information service as if they are local based on agreements with that service, ideally supported by digital or human protocols. It then presents the selected service locally, potentially introducing adaptations when details of that service change, but otherwise representing it accurately.

In some cases the implementation may combine these using a workflow mechanism. Consequently, the relationship with the provenance system, P4, needs careful design to avoid unwanted behaviour. This is revisited in contemporaries (§4.2.4) and in DKB R&D (§4.2.5).

When fully exploited the methods created by research developers will use the DKB frequently. The interpretation of user actions, developer actions and the coded methods will all interrogate the DKB to translate into finer-grained actions, to map to evolving infrastructure and to optimise based on accumulated information about prior work, about users, about services, about software and about data. This is illustrated in Figure 4.3.

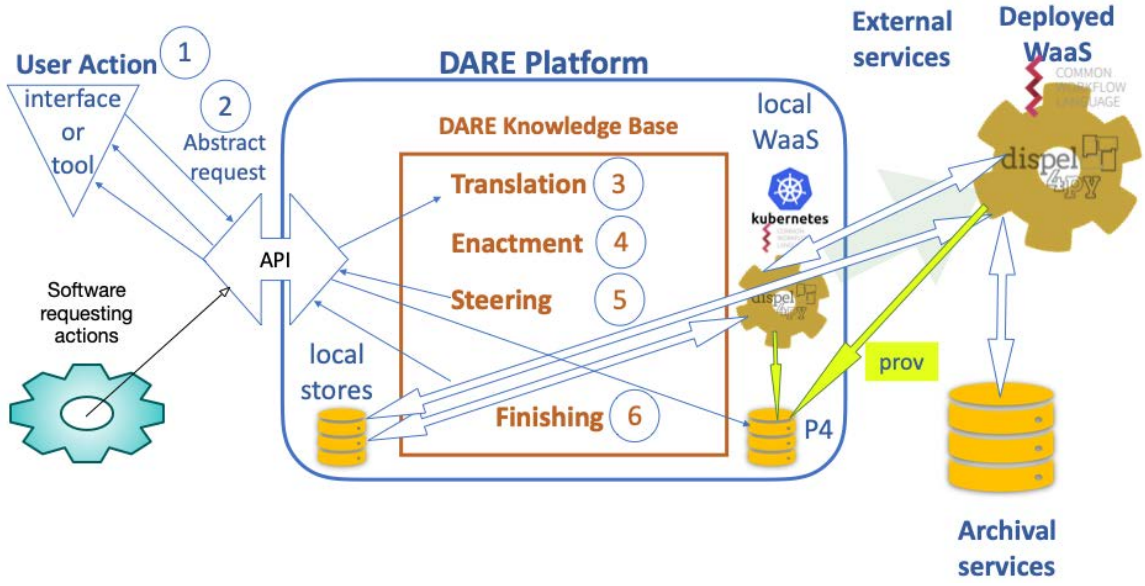


Figure 4.3: The DKB acting as an intermediary throughout the initiation and enactment of a sophisticated method steered by and reported to a DARE platform user. The numbered stages of this process are described below.

Figure 4.3 depicts a future DARE platform where the DKB is used intensively.

[33] Protocols may be introduced to detect and warn of this divergence, but they are not standardised or commonly available - they should exist for DKB contemporaries (§4.2.4).

1. A user, using a tool such as a Jupyter notebook, requests an action in a form suitable for them. Similarly, an action may be requested by some external or internal software.
2. The request with parameters, etc., arrives in an agreed form at the DARE platform API. Entries in the DKB describing methods and built-in functions automatically populate this API.
3. The DKB retrieves information about the identified method, parameters, and referenced data. It passes this to the WaaS, which may use this information, may request more and may write records for subsequent parts of this enactment or future similar enactments.
4. The WaaS receives the request and may request further information from the DKB in order to optimise, map to a target technology and to deploy and configure the required virtual infrastructure[34]. It may consult the DKB for descriptions of potential targets the requestor is authorised to use. It will ask the DKB about past costs to estimate expected costs. Harvesting-processes may scan provenance data to summarise past costs.
5. During the conduct of the workflow, information gathered by P4 will be transformed using DKB data about the requestor's preferences. Similarly, incoming steering actions will be translated using DKB information. Engaging the DKB in the information flows from the DARE platform and running software is a crucial innovation. It is during such flows towards users, particularly when failures are reported, that systems expose technical detail that was abstracted away in input flows. Mapping these to forms that are easily understood by the interacting user is essential[35]. Otherwise, users have to learn to understand them. They may then exploit them in their future work, locking their methods into a particular technical context. At the very least it is a distraction. However, developers may want them. Hence the tailoring to the current user.
6. The final records written in the DKB will link up the run with the provenance records and with results and if they were specified in the run's profile, intermediate data sets normally discarded. If a user chooses to consult the results and diagnostic data at any time in the future, the DKB will 'know' where they can be found, how they can be retrieved and transformed for that user.

Each deployed instance of the DARE platform will have its own DKB instance as shown in Figure 4.4.

[34] This may require integration with the orchestration technology, currently Kubernetes (see §4.1 & §5.2).
[35] By working via P4, the mapping is from one standardised, PROV-O representation that avoids higher-level tools and platform elements having to work with the vast diversity of system monitoring data.

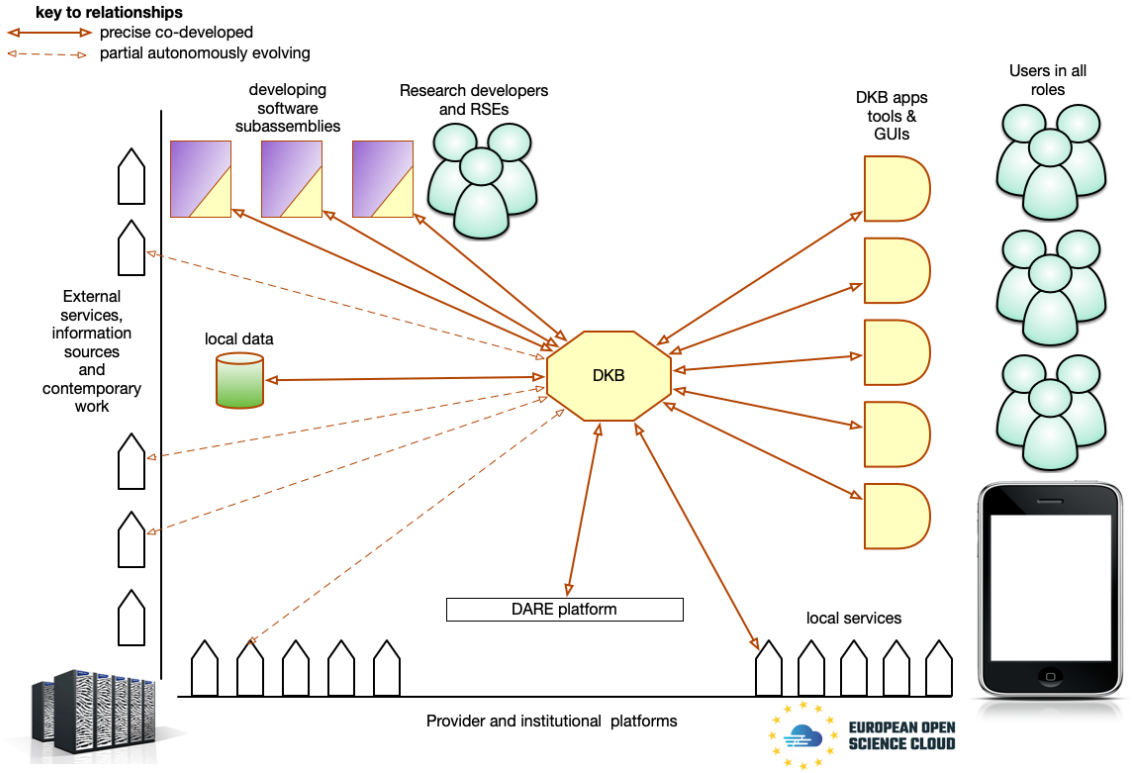


Figure 4.4: A typical deployment of a DKB maintains relationships in two forms. New applications and tools developed using the DKB maintain consistent information in it - solid arrows. Legacy systems and external systems controlled by others will not have complete information in the DKB. It will hold only the relevant aspects when last used - dashed arrows.

This incomplete information is an inevitable consequence of supporting research and innovation that may lead anywhere[36]. The quality and reliability of DKB information therefore depends on the care and precision of its developers and users. To mitigate the risk without constraining users with an excessively rigid regime we partition the DKB information space into *research contexts*, *Contexts* for short, that are the analogue of file-system directories and workspaces.

4.2.3 DKB contents, structure and functions

Users[37] are free to record any information in the DKB that they choose. However, that freedom has three problems:

1. It takes far too much effort before the benefits are available to most users.
2. It is hard for anyone else to understand what has been done.
3. Optimisation is limited to the regularity which software can uncover.

These are addressed by the following mechanisms:

1. Built-in regularity in the properties and management of every Instance.

[36] It is also necessary because the DKB has to be introduced into operational contexts and it will not 'know' about everything that is going on.
[37] This includes any software that writes to the DKB.

2. Contexts over which users have total control; these range from very stable contexts denoting selected aspects of the knowledge infrastructure to 'lab-bench experiments' exploring a new insight.
3. A specifiable inheritance of information from other Contexts.
4. Local naming within Contexts.
5. Global persistent identification using automatically generated PIDs.
6. A conceptually organised library provided with releases of the DARE platform.

Instance specifications

The Instance as the unit of recording was introduced at the start of §4.2.2. It will have a set of built-in attributes several of which are set automatically. These are illustrated in Listing 4.1.

```
"Common attributes of every Instance in a DKB"
name: String            # name unique in a Context
prefix: String          # its prefix is unique in this DKB installation
pid: String             # persistent exportable identifier
instance_of: Concept    # every Instance is an instance of a specified Concept
timestamp: Instant      # when the Instance was created or changed
state: String           # where it is in its life
predecessor: Instance   # if an Instance updated, PID of previous version
successor: Instance     # PID of the next version of this instance
```

Listing 4.1: The built-in properties of every Instance see [Levray 2020] for details.

Users may add any other data in an Instance supplied as a Python dictionary. This is supported and controlled by a conceptually organised library provided to meet common requirements. It may be extended by communities and user groups to help with their additional recurrent needs or to specify how their information should be organised[38]. To encourage this we establish Conceptual modelling through the conceptual library[39].

The chaining of versions when an Instance is updated, is intended to allow stand alone uses of the DKB to work without a separate provenance capture system. The DKB will provide simple functions for which this record keeping is efficient. They will normally be atomic and affect only local state.

Persistent Identifiers (PIDs)

The PIDs are manufactured as shown in Figure 4.5.

Figure 4.5: The structure of the PIDs manufactured for each Instance

[38] For example, seismology groups might establish a Context holding the FDSN features they use and another holding the OGC features they use. Groups and individuals would add these to their Context's search paths.

[39] Of course, the conceptual library provider may build this by using the Entry functions.

Precise and persistent identification of anything in a user's or a software system's world is essential to ensure unvarying interpretation of those entities when required, e.g., to ensure that an established practice is conducted consistently or to achieve reproducibility. The DKB takes on this responsibility by forging, preserving and interpreting PIDs. As users and software may copy references to Instances in the DKB, this interpretation cannot depend on local addresses or current storage arrangements. The first part of the PID identifies the particular installation of the DARE platform where this DKB service is employed. If this identity, when expanded, meets PID guidelines, e.g., those espoused by the PID forum[40], then the Instance identity will also meet those, i.e., both will be URIs. But each user community will choose how formal to make their PIDs and how much to invest in ensuring the longevity of their interpretation. The context prefix is a string guaranteed to be unique in this DKB instance[41]. The uniqueness counter ensures successive updates to an Instance have a different PID[42], i.e., a PID identifies a specific version of a specific instance. The user's identifier is the name given by a user, e.g., as an identifier in their Jupyter notebook. The DKB can act as a proxy for an external information source or a contemporary service and forge a local Instance with a PID containing a reference to that external service. It may hold timestamps and signatures of the referenced entity to detect autonomous mutation. We now have the machinery in place to build and use research Contexts.

Context specifications

The primary role of research Contexts is to gather a set of Instances to provide a work context well adapted to a particular user working on a particular task. They also represent the common requirements of multiple users, e.g., members of a group working on a common problem, or of multiple similar activities, e.g., repeated performances of a standard procedure. This is achieved by nested Contexts with specified inheritance from outer Contexts, as shown in Figure 4.6, where we see an additional Context for each user community and an upper-level, dare, common to all DARE applications, which in turn builds on a universal conceptual library, kb, and pre-imported bundles from standard sources.

[40] https://www.pidforum.org/
[41] It may also have a defined expansion for URIs in ontological representations - behind the scenes.
[42] An index from PID to Entry would accelerate DKB operations.

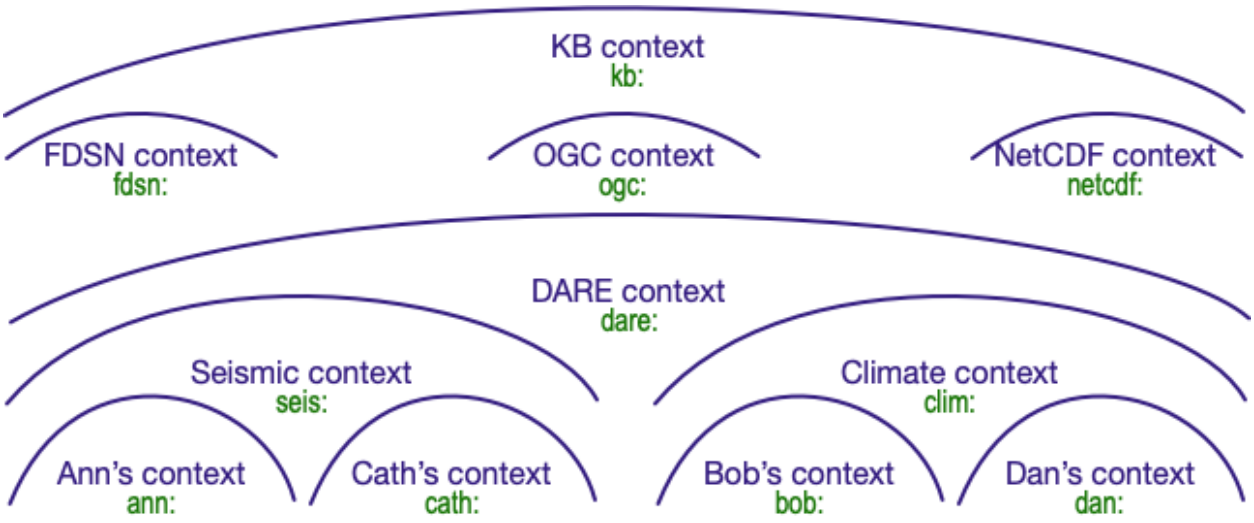


Figure 4.6: The nesting of research contexts, progressively forming work environments that are highly tuned to an activity, an individual or both.

A Context has a prefix that is unique in the DKB installation. It may have an initial population of Instances. These may be updated, added to or discarded as the Context is used. Each Context has a specified search path of other Contexts, e.g., ['seis', 'dare'] for Context 'ann', to arrange inheritance, and to specify overlapping interests. If users wish to inherit from a Context c, with prefix 'c', adopting all of c's inherited Instances, they simply specify ['c'] as the search path. This is the default, if they are in Context c when they make the new Context. The effect is transitive.

The uses of Contexts include:

1. *Importing a bundle of terms* and related entities into a KB, as illustrated by three Contexts in the second row of Figure 4.6. These may then be used in any search path and be separately maintained, e.g., to reflect changes issued by the authoritative source.
2. *Denoting a set of shared terms* and resources, as illustrated in the next two rows of Figure 4.6. Communities may govern how these are maintained. Explicit references using a full PID delay the impact of changes in such Contexts.
3. *Providing a work Context* for an individual, a group or a procedure, that is progressively tailored as it is used to better support that work. Illustrated as the bottom two rows of Figure 4.6.
4. *Providing a method enactment Context* (not shown in Figure 4.6). Methods are repeatedly run with the same or different parameters. The method needs a new Context for each run, so that it can use the same set of names each time, differentiated by the automatically varied prefix.
5. *Acting as a boundary for access controls* and authorisation. The Context can have a consistent aspect of sharing, privacy, confidentiality, etc., and an owner or governance body can set that.

6. *Supporting a user-controlled transaction*. The updates to all Instances within a Context from a defined time (denoted by the uniqueness counter) to a chosen instant can be considered together, e.g., to be 'pushed' to persistence or to be retracted[43].

An Instance in an explicit Context, not necessarily in the search path, may be specified as <prefix>:<name>, e.g., 'cath:eqEvent20190723' from Ann's context to refer to the latest update of the earthquake event Cath is working on. A platform may provide functions, e.g., publish, to reveal such Instances and to control visibility and mutability. Within such constraints, a user may specify an explicit version using its PID, e.g., to freeze a method in an authorised form, while others are developing and releasing a future version.

When a user interacts with a DARE platform they use a specific Context[44]. To facilitate induction, they may be allocated a clone of a group's Context when they first start work. Their work will modify their current Context so that all new things they create and revisions they make appear *in their Context*. As searches are by name in the local Context first, and then in each Context along the search path, when DKBs are mature they will start in a rich and productive Context. They can proceed uninhibited to use names and create Instances as these are local to their Context. *Innovation is uninhibited*, since they can redefine things named along the search path and thereby hide them and experiment with new forms.

However, this does not support collaboration. That has to be done, by publishing new things to 'friends' and then they explicitly name them using your prefix, e.g., to confirm that what you have done is valid and useful in their Context, or to conduct the next authoring steps. When authorised, validated and valuable Instances may be promoted to a higher shared Context others in a group can then use. Authorised users will explicitly visit a Context, e.g., to work directly on a shared context, to move between production and innovation, to maintain a shared Context or to help someone solve a problem[45].

The methods for conducting routine repeated processes start by cloning a Context containing the Instances that differ for each repetition, so that they are grouped under the new Context's prefix identifying the repetition with the same local names for every repetition. A user responsible for a number of these running concurrently will move between them. Details of the operations on Contexts may be found in the URM [Levray 2020].

Contexts are *not* static snapshots. They continue to change. Users and methods may exploit this. For example, they group a bundle of changes during an experiment or while a method runs. If these are all to be discarded, the Context that holds them can be discarded. Similarly, if they are to be retained, the Context may be archived. A Context associated with a method under development can be readied for reuse with a reset function.

[43] Uses 5 and 6 will not be attempted in the initial prototype.
[44] Normally the one they were using last time, which involves their identity being obtained via AAI and this being mapped to their Person PID in the DKB. This needs inclusion in the platform's login API.
[45] This is a common requirement for support staff. It may require permission from the Context's owner.

Conceptual library specifications

Whilst users and software have the option of using the DKB in completely novel ways, there are compelling reasons for providing an initial library of Instances all in the Context ‘kb’, as they span potential KB platform uses. This will reflect invariant properties of the platform and universal aspects of the knowledge infrastructure. The commonalities expected in the variety of DARE applications will then build a Context ‘dare’ that exploits ‘kb’, see Figure 4.6. These introduce several groups of Concepts. We focus on *Concepts*, e.g., Energy, Temperature or Country, as they underpin thinking and communication in humans.

We expect a body of well-established and widely adopted Concepts to underpin every research community; developed and sharply defined in their minds by their education and training and reflecting their knowledge infrastructure. This enables them to communicate and think effectively, as they use words with those precise meanings. In the DKB we intend that they should use those same terms as easily and precisely; building on Trani’s work with EPOS [Trani 2019, Trani *et al.* 2018].

Contexts enable different groups to use different terms or the same terms differently. In due course, to enable boundary crossing and novel interworking between disciplines without excessive consistency requirements that slow innovation in multidisciplinary collaborations. The DKB will need an underpinning metadata translation and cross-referencing framework, to enable collaboration and innovation to co-exist productively. The requirements and a viable approach are well illustrated for Europe’s museums collections of natural science specimens [Lannon *et al.* 2020]. The EU VRE4EIC project demonstrated the automation of metadata translation to deliver an integrated view of catalogues [Martin *et al.* 2019]. We need to build the foundations of the DKB and establish its initial use before these issues can be explored. Current thinking looks at immediate needs. The need to have variety and consistency, stability and innovation co-existing and flourishing together will emerge as soon as sustained use for a research campaign is attempted [Ramakrishnan 2018] (see page 263).

As research progresses the revision and refinement of the Concepts is inevitable. The DKB supports and stimulates those Conceptual refinements by enabling application communities and individuals to directly shape their active repertoire of Concepts.

We provide a Concept library prepopulated with Concepts and a structure relevant to data-intensive and computationally intensive research campaigns. Building on these, a research campaign or community will develop a sophisticated and highly tuned set necessary for their research. We expect this to become a significant intellectual and practical asset. Groups, organisations and specialists will also build on the initial foundation, on existing and contemporary developments and on imported bundles of relevant knowledge. The common starter includes:

1. An initial set of Concepts, some of which have instances, to provide users and application domains with common information and organisational structure that they will need.
2. Concepts that are examples to help those developing the use of the DARE platform, for themselves or for sub-communities.

3. Support for consistent structures, particularly for handling Collections, to provide optimisation opportunities and to steer users to well supported or efficient methods.

The conceptual library will also present an API with Python methods for *basic* Actions to change the state of the DKB. These include:

1. Actions on Concepts to define them and their relationships, evolving them and managing their lifetimes.
2. Actions on instances of Concepts, creating them, finding them, interrogating them and updating them.

As the definitions of refined Concepts and sophisticated Contexts develop new functions specific to particular Concepts and methods composing basic functions will be introduced. It is intended that user communities will become self-sufficient, initially in using the basic functions and eventually in refining them and in maintaining sophisticated Contexts and home grown Python encoded Methods. To enable this, once *published*, new Concepts, new functions and new Methods should automatically become available in the API. Eventually their use may be controlled by authorisation mechanisms, but initially we will depend on communities collaborating with careful consideration for their colleagues. This is only feasible while the user community is small.

It will be possible to build more complex and longer running Methods out of these basic functions and other Methods, including those in any workflow system that DARE supports. Some of these may be used to manage aspects of the DKB, e.g.,

1. Promote a set of revised Concepts and instances from an innovation Context to a shared Context.
2. Import a bundle of information from an authoritative source, such as a curated ontology, creating a Context to represent it.
3. Build a Collection representing the current Python loaded libraries.
4. Compare two such Collections and report on their differences.
5. Visualise the tree of specialisations of a particular Concept.

Such Methods will be implemented when needed. When they are not simple functions[46], they will need to record progress in a provenance service, such as P4.

It is intended that DKB users will be able to do everything that they need to do in this way, i.e., by working in Contexts, by building on provided Concepts, and by composing basic functions and Methods built ultimately from basic functions. At the same time, the DKB should help experts serve application communities by supporting Contexts containing the Methods and information they develop, e.g., workflows, so that their products may be used straightforwardly. Data architects, data scientists, systems specialists and software engineers may also work on critical Instances once they have stabilised and proved useful, to improve their performance, reduce their costs and improve their provenance tracking. A future development made feasible in the DARE project.

[46] For example, they may fail after changing some of the persistent state, and provenance records will be needed to support attempts to complete the work or to undo what cannot be completed.

Concepts are introduced as they reflect established ways of thinking and communicating in application and technical fields. They often have agreed names, developed in the application's culture and global consortia and corresponding to an authorised terminology[47]. For each viewpoint/use of a Concept they have properties used by practitioners. The DKB supports communities and individuals agreeing and using their Concepts as shown in Figure 4.7. Note that an application community will have familiar Concepts, with well understood names, that are understood by them, but not by the systems team that enables their computational and data-driven work. Similarly, those system experts have their own vocabulary, Concepts and properties that they understand. The points where these worlds overlap, called "*boundary objects"*, are vital for effective collaboration, and have to be well supported by the DKB.

In some senses, Concepts are similar to classes in object-oriented programming. Indeed users may be helped by being able to use Python classes in the programs corresponding to the Concepts they are using, with instances of each class corresponding to instances of Concepts. *The extent to which the DKB system will automatically support this has yet to be decided*.

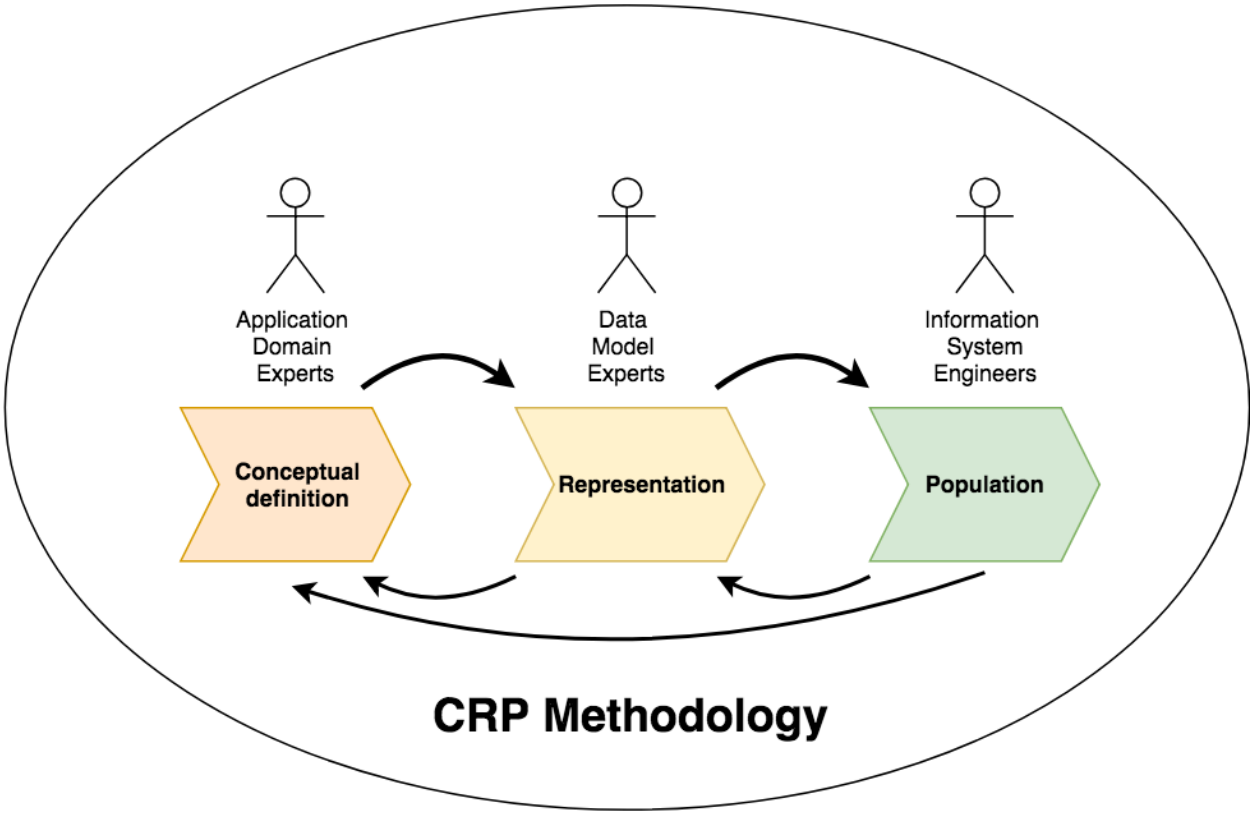


Figure 4.7: Showing the CRP methodology to develop and manage concepts shared by a collaborating research community (taken from Fig. 5.2 [Trani 2019]).

[47] These precise terminologies may draw on and implement LOD ontologies, but it must not be a requirement to understand ontologies and OWL to use the DKB. However, experts in that approach should be able to help users using their knowledge. For example, they may develop and use methods to import authorised ontologies directly into the DKB and ready for production use.

Having decided what Concepts they need, users choose the properties that they consider important, and how they would like them to be represented consistently. We envisage three categories of attributes:

1. *Mandatory*, ones that must appear in every Instance in addition to the universal attributes (see Listing 4.1). This has the effect of specifying the name and the form of each attribute. It will trigger an error if an instance is made which does not have the attributes in the correct form.
2. *Recommended*, are properties that should be present in every instance, this has the effect of specifying the name and form of the attribute and may have the effect of prompting users to supply values for these attributes in each instance.
3. *Optional*, is a category of attributes that may be included. This specifies the name and form to be used whenever an attribute appears in this Context with the given name.

Concepts, may specialise other Concepts, e.g., SpecialWidget may inherit from the definition of another Concept, e.g., Widget. In that case, instances of SpecialWidget may appear whenever or wherever instances of Widget are required.

There are substantial conceptual, organisational and practical advantages from delivering in the library a harmonised bundle of Concept, Method, Data and Collection Instances. We anticipate each platform release will include advances in this bundle. Please consult [Levray 2020] for more information.

4.2.4 DKB contemporaries

These include the data catalogue and the Registry. They also include some aspects of the P4 provenance handling framework described in §4.3. They provide two valuable aspects to the design and development of the DKB:

1. As well-established and populated subsystems they provide a significant part of the required functionality.
2. They also ensure that the DKB does not require a ‘green field’ operational context, as cooperating with complex contemporaries is essential for long-term adoption.

Data Catalogue

The DARE data catalogue has been operational throughout the platform’s development and provides basic services for recording information about the files in use. This has been enhanced and extended as the DARE Semantic Data Discovery Service.

The Semantic Data Discovery Service builds on the Data catalogue component of the DARE platform. The Data catalogue stores metadata about datasets in DARE described with the Resource Description Framework (RDF) model, which conforms to the Data Catalogue Vocabulary (DCAT). Currently there are only low-level interfaces to these datasets, provided by the RDF database Openlink Virtuoso (SPARQL, ODBC, JDBC, ADO.NET, OLE DB, etc.). The

complexity and low-level nature of these interfaces inhibits the full use of the Data Catalogue's information.

The Semantic Data Discovery service should enable a user to access the data stored in the Data Catalogue conveniently. To achieve this goal the application scans recursively through the existing datasets, indexes all known information patterns found and provides an interface to search this data. The Python code provides functions, such as trigger indexing, deleting the index and starting a search. These are accessed via a REST API using the Python web development framework Flask[48] and are exposed by OpenAPI Swagger[49]. To create, manage and use an index, the search engine Apache Solr[50] is used. This offers a wide range of functions including: simple text-based search, a search by date or by geo-location. It is easy to add vocabularies[51] to meet the need to be open-ended, as a wide range of specialised linked data vocabularies may be used. The implementation architecture is shown in Figure 4.8.

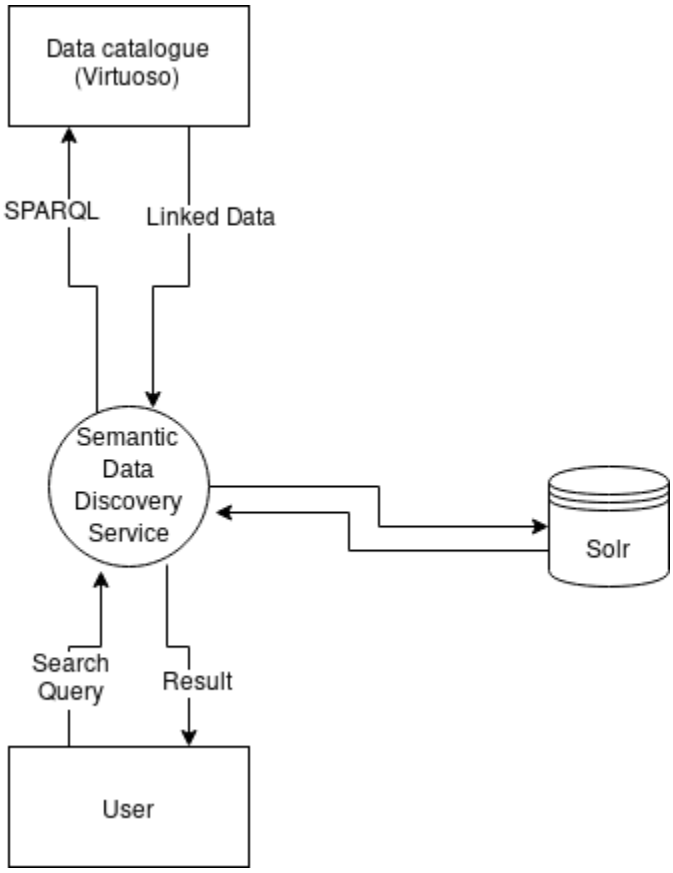


Figure 4.8: The architecture of the Semantic Data Service extension to the Data Catalogue.

[48] Flask https://www.fullstackpython.com/flask.html
[49] Swagger https://swagger.io/docs/specification/about/
[50] Apache Soir https://lucene.apache.org/solr/
[51] This is done using a configuration file.

In the future, a web GUI should be placed on top of the search, for example comparable to the European Data Portal[52]. An integration with the provenance database is also planned. A further extension could be the integration of external Data catalogues to meet known requirements for the climate or seismology communities. This can be done as a change of configuration, as long as the data exist with the vocabulary DCAT and are provided through a SPARQL endpoint.

Registry

The registry is a part of the DARE platform that coexists alongside the DKB. As do the provenance system and the Data Catalogue. The previous item discussed the relationship between the Data Catalogue and the DKB. Here we focus on the Registry's relationship with the DKB. We then consider the relationship with provenance.

The registry was developed in the VERCE project[53], where authors of [Klampanos *et al.* 2015] prototyped the dispel4py information registry to facilitate consistency and collaboration in workflow development[54]. Consequently, the current state of the registry is strongly linked to dispel4py. The current version is implemented in Django[55], a Python-based Web framework and is linked to a relational MySQL database server. The current usage allows for the development of workspaces, and the storage and production of information regarding workflows.

The registry is currently used in the DARE platform via an API, to register workflows, (i.e., register PEs of the workflow). It also can execute and monitor the runs of the workflow, i.e., use the provenance system to stream provenance traces at run time. The main functions of the API are given in Section 3.3.

As mentioned, the registry deals with the notion of workspaces. In its design, workspaces refer to a snapshot of whole sets of components linked to registry (including the registry) to allow for refined, specific, user-defined context work. In that sense, the idea of workspaces in the registry is closely related to the definition of contexts in the DKB.

In order to make use of the DKB and registry together, it is critical to link those two definitions. It is also important to add more specificities either to the registry or to the DKB so that its uses cover more than dispel4py. Indeed, the idea is for users and developers to be able to register methods in any format they desire, including: bash script, python script, CWL, dispel4py, etc. So far, the CWL option has been included as a logically parallel service.

Relationship with P4

The provenance system runs a database into which it collects information from many runs and from different technologies into a standard form §4.3 and [Spinuso 2018]. It also collects metadata associated with runs specified by users, supports tools for examining and visualising these

[52] https://www.europeandataportal.eu/
[53] http://www.verce.eu
[54] More information can be found in https://zenodo.org/record/3361395#.XfjPVOvgrUZ.
[55] https://www.djangoproject.com

records. It can be accessed by a set of web-service functions including specific requests to export selections of its data in standard formats. Consequently, it is performing the role like that of the DKB: there are therefore two directions of development to consider:

1. *Independence*, each develops independently and users/developers tackle the integration of their functionalities. Even with this approach they still need to coordinate, cross-reference and align their treatment of entities that they both handle.
2. *Integration*, where they converge through co-design, so that eventually users and developers see them as one system that they use unaware of the two subsystems.

Integration is clearly the desirable long-term goal, but it is so challenging that a period[56] of *independence* but *convergence* is needed before it is attempted. The necessary *interdependence* will need to progressively deliver:

1. *Coordination*, e.g.,
    a. notification to P4 from DKB when it starts a com[plex function that needs provenance,
    b. notification to the DKB and P4 when the WaaS detects completion or failure.
2. *Cross-reference*, the following examples indicate what will be needed:
    a. References that the DKB can use to refer to traces in P4, preferably PIDs,
    b. References (PIDs) P4 can use, to refer to entities identified via the DKB, preferably with known fixity.

    The timing of PIDs being allocated and being used in each subsystem, as well as the decisions as to which entities require PIDs, will need to be worked out as the alignment is developed.
3. Alignment of representations, such as:
    a. Identification of individuals and representations of attributes both use.
    b. Identification of sessions and representations of attributes both use.
    c. Ditto for computational environments, containers, deployments, software, data and services, etc.

This convergence should be achieved incrementally in co-design and co-development closely aligned with pressing requirements from use cases, e.g., as identified in §3.4.

4.2.5 DKB Status and Potential

The current status of the DKB is that there are three mature, operational subsystems: data catalogue, registry and the provenance system independent from one another and a prototype general-purpose DKB that has yet to contribute to progress beyond beta testing in one pilot context. It is clear that users, particularly research developers, would benefit from increased automation and integration (see §3.4). As explained above, the *general-purpose and directly manipulated* DKB should help those extending and developing data-intensive and computationally demanding applications in the context of existing systems and established practices. The co-development and experimentation with WP6 suggest this potential is a short

[56] Almost certainly longer than that remaining in the DARE project.

step from being fully available. This is precisely what DARE promises to provide and we should therefore use the opportunity to develop the general-purpose element of the DKB:

1. To help the domain esports and research developers in our user communities and in successors to DARE, and
2. To use DARE to progress to the point where the general-purpose element of the DKB has proved to be useful and feasible, so that it is both sustainable and a good foundation for future work.

General-purpose DKB implementation

The general-purpose DARE knowledge base (DKB) is implemented as a RESTful Flask web service (microservice) and a Python library to help users build clients. The open-source code can be found here https://gitlab.com/project-dare/dare_kb. The functions provided to create, access and manipulate contexts, concepts and instances are described in the User Reference Manual [Levray 2020]. These can be used directly but they have also been proven efficient and useful when used (see below) in an experimental version of the DARE manager located here https://gitlab.com/project-dare/exec-api/-/tree/dkb_incorporation.

The information in the DKB is represented as an ontology, which helps collaborating researchers develop a common vocabulary. There is an initial version on which they build suitable for the communities DARE supports. It is a language that allows for formal specifications of concepts and the relationships between them. The reasons for using ontologies in general are[57]:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge.

The DKB provides functions for directly describing concepts and their properties using Python functions familiar to our user communities without them being aware of the ontological underpinning. They can then build and update populations of instances of *their* concepts capturing whatever data and relationships they require.

This information is stored in SQLite3 via owlready2[58]. The implementation delivers the novel information-structuring mechanism, context, and organises the way contexts use the contents of other contexts. We create all concepts, instances and their relationships within nested contexts, allowing lower levels to work on specialised research and upper levels to hold broadly shared information.

The DKB is very similar to many contemporary knowledge bases, i.e., it organises sets of entities and the relationships between them represented in a graph database often using OWL or other LOD representations. However, the DKB offers a significant innovation, a user or community

[57] https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html
[58] owlready2.readthedocs.io/en/latest/

controlled contextualisation structure, that could, in principle work with or be overlaid on any of those knowledge bases. This delivers substantial benefits in enabling collaboration and independence to coexist, and in supporting the range from established stable knowledge to exploratory experiments.

DKB demonstration

Working closely with WP6 we have developed an experimental demonstration of the DKB working on their MT3D Jupyter notebook (see D6.4 §2.4 [Magnoni *et al.* 2020b]). The goal was to demonstrate improved abstraction and reduced repetition resulting in less detail and less repetition - reducing distractions, workload and learning thresholds. This has been achieved by creating in the DKB, instances of workflows with information about their source code, default parameters,default inputs, etc. needed for their execution. This information can be updated producing new identified versions. This is incorporated into an experimental version of the DareManager helper functions for creating new users, new sessions, etc. Instances are automatically updated with the right information when a workflow is registered. Any information can be viewed via the get and find functions using the names users prefer that only need to be unique and understood within a context. An important feature of the DKB is that it allows for all the versions of an instance to be kept, which as we anticipated proved a significant benefit to users. The result is illustrated in Figure 4.9.

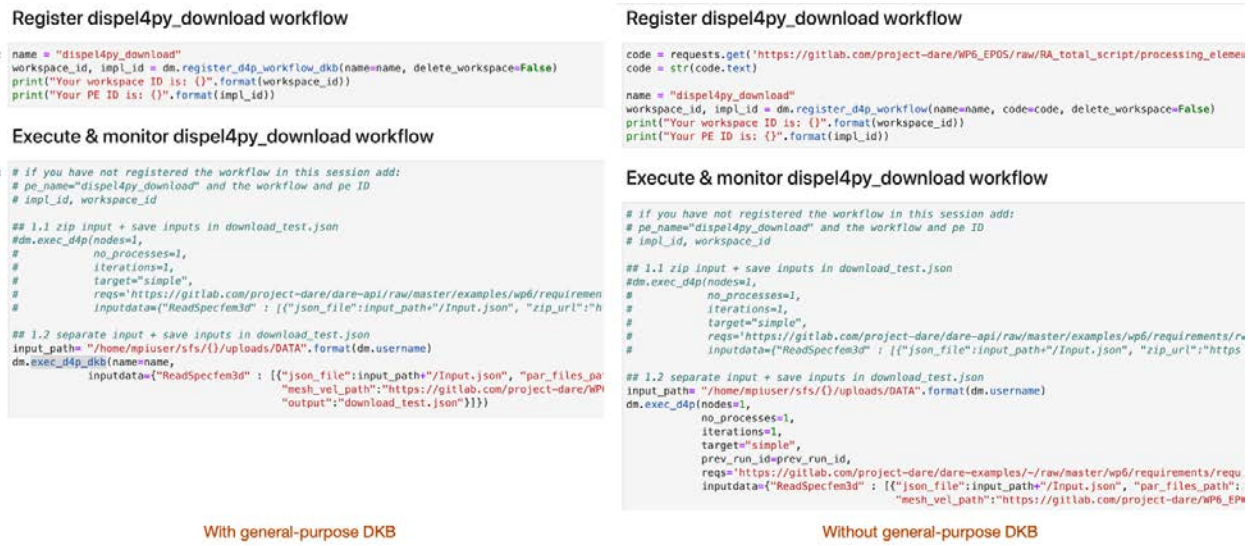


Figure 4.9: *Illustration of the simplification achieved by using the general-purpose DKB to manage information for a user. The two fragments of Jupyter notebook are equivalent. With the DKB (left) the user has significantly less work to do compared with the previous Jupyter notebook (right), but still has full control of everything. This is a preliminary demonstrator that runs but is not yet complete.*

4.3 The P4, tools and interaction interfaces

P4 includes the provenance functionality that enables the acquisition and exploitation of provenance data. DARE focussed initially on capturing lineage information about the execution of a method. This is described by the initial inputs, the method's processing elements, and the

computational resources used. We acquire provenance from different types of systems (CWL and dispel4py). Although both build their provenance on top of W3C-PROV[59], we mapped CWLProv specialisations to S-PROV, in order to be interactively explored and visualised using DARE's S-ProvFlow tools and lineage API. The lineage information recorded from the execution of a dispel4py workflow can be tuned adopting a provenance configuration and contextualisation system developed during the first half of the DARE project [Spinuso *et al.* 2019]. For CWL, we started by focussing on managing provenance information produced by CWL *SPECFEM3D* workflows, implementing a mapping to S-PROV that we have incrementally adapted to support CWLProv in a generic way. The provenance generated by such workflows, can now be sent to the lineage services API and visualised in the S-ProvFlow viewer, as shown in Figure 4.10. These have been implemented according to the requirements of the use cases defined in WP6/WP7 summarised in §3.1 and §3.2. Here, we are interpreting and summarising the CWLProv information produced by such API calls in order for it to be integrated and accessible through S-ProvFlow.

Ultimately, CWL and its provenance component aims at the generation of research-objects[60]. Pursuing this model depends on the implementation and policy of the DKB in terms of the storage and description of the results generated through DARE. We foresee provenance data to be linked from DKB entities describing the final dataset. This will require the proper generation of the results' PID and the reference to the endpoint that will extract from P4 (s-ProvFlow) the full lineage trace associated with the data product. This would comply with the RDA indication for a pattern enabling the linkage between metadata and provenance[61].

[59] W3C-PROV https://www.w3.org/TR/prov-overview/
[60] Research Objects http://www.researchobject.org/
[61] Associating metadata in documents with graph provenance https://patterns.promsns.org/pattern/12

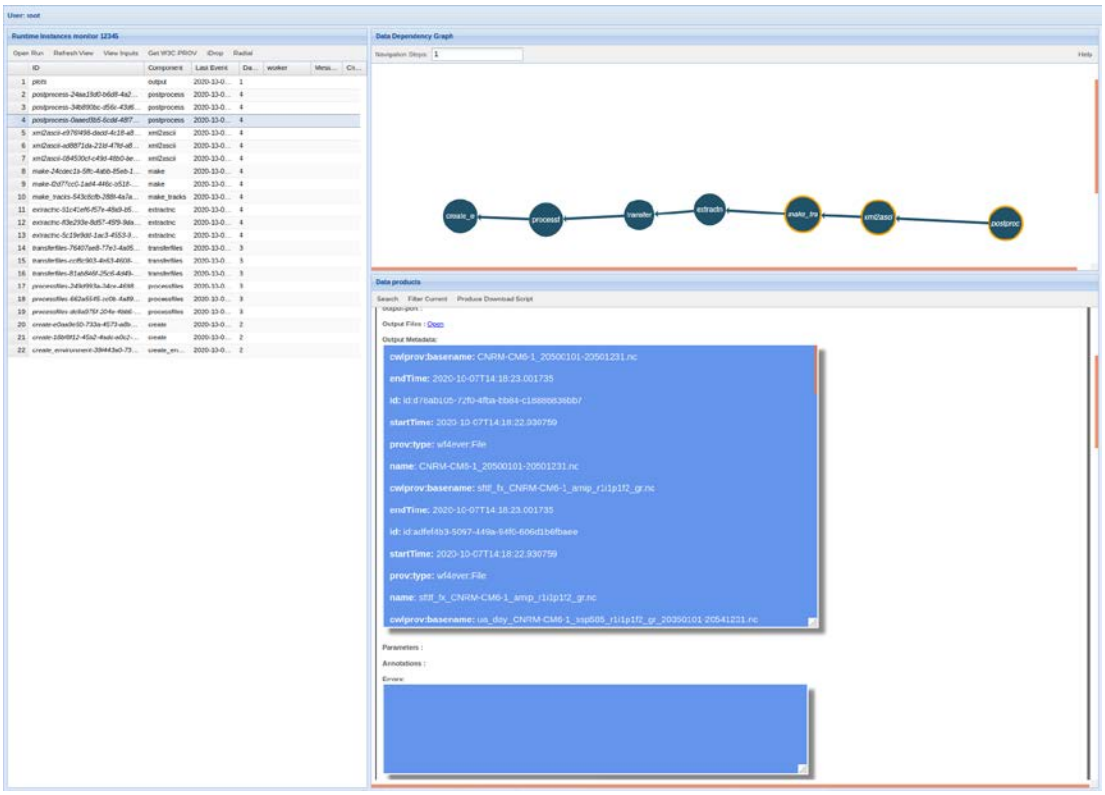


Figure 4.10**:** *Lineage of Cyclone Tracking CWL workflow shown in the S-ProvFlow viewer. Here the workflow processes, their outputs and dependencies are described adopting the metadata vocabularies supported by CWL*.

Once the lineage has been stored we provide three kinds of exploration functionalities. Live monitoring, Lineage queries, Discovery and comprehensive visual Analytics. These have been described in the DARE literature [Spinuso *et al.* 2019, Atkinson et al. 2019, Klampanos *et al.* 2019] from the point of view of their conceptual functionalities and integration within the platform. More technical insights are provided within the DARE deliverable produced by WP3 [Spinuso *et al.* 2020].

Latest developments addressed improvements to the lineage query methods. We provide a more usable and powerful set of methods that will allow users to search over combinations of terms' using value-ranges or value lists. These improvements will be reflected in the s-ProvFlow viewer too. In Figure 4.11 we show the prototypical interface that allows users to search for workflow executions with a short explanation of the new syntax.

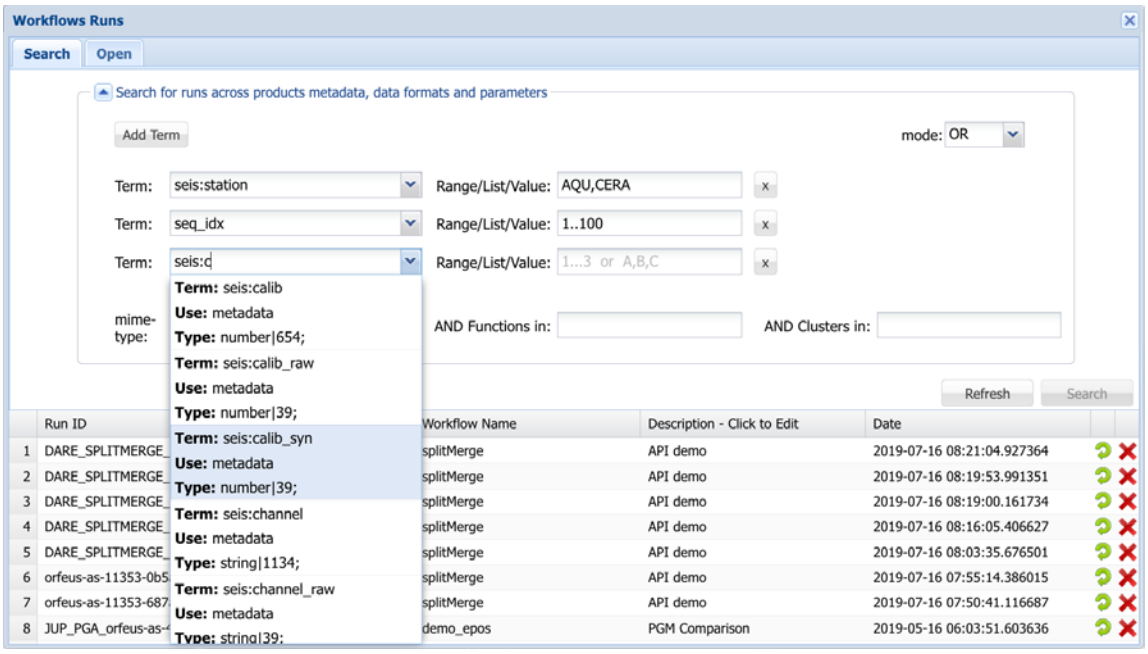


Figure 4.11: *MVV Workflow Execution search panel. The panel allows users to compose the query interactively by specifying more terms and expressions indicating lists or ranges of values. The list presents the user's runs that match the search parameters.*

Recent developments have been largely dedicated to the integration of security mechanisms for the authorised access and storage of the provenance data (AAI). This is pursued in a way that meets the GDPR regulations in terms of the separation, "by design", between the recorded lineage traces and any deducible information about the users themselves (e.g., username, email, identity). In this respect P4 relies on the DARE AAI (see §5.2) and the DKB (see §4.2) for the complete resolution of users' personal information. Implementation details will be reported in the official deliverables about the platform deployment in WP5 [Roth *et al.* 2020] and the updated lineage services in WP3 [Spinuso *et al.* 2020]. In Figure 4.12 we show an updated schema of the services architecture

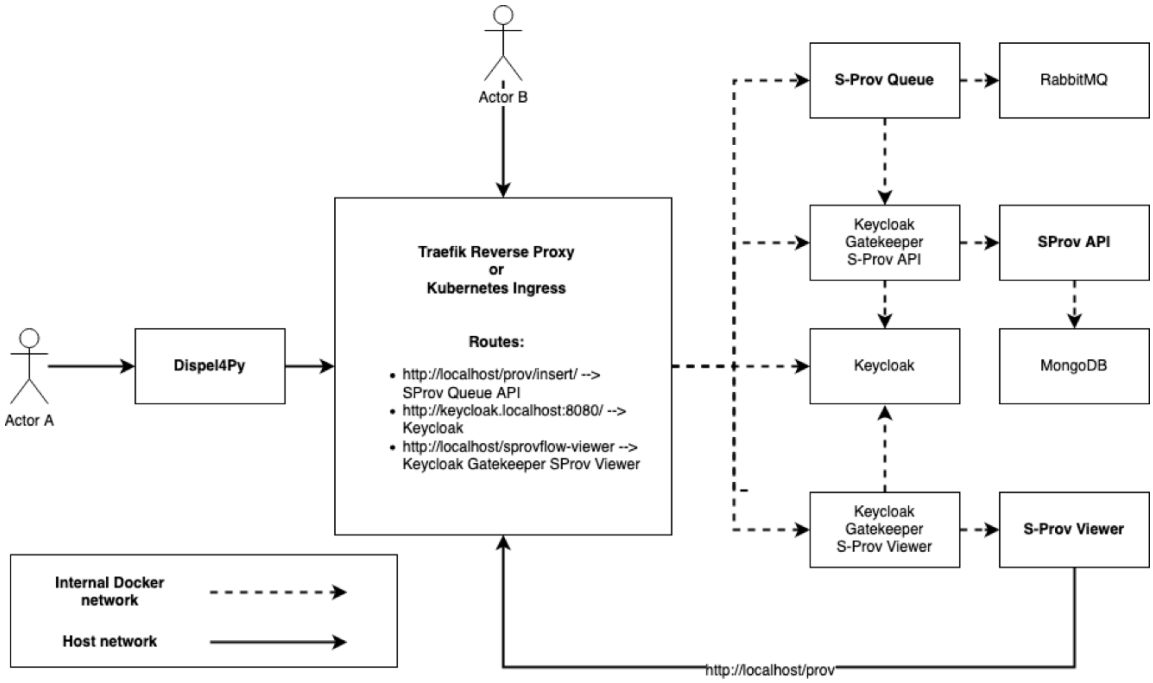


Figure 4.12: *S-ProvFlow integration in DARE. Actor A is the user starting the workflow. She delegates to dispel4py the access to the lineage services. Actor B instead directly accesses the provenance information through the S-ProvFlow viewer. The schema includes the new components managing the authentication of the calls by the different actors (Keycloak and the two dedicated Keycloak gatekeepers). Updates from disple4py are sent to the S-Prov Queue. This module extracts the credentials and performs authenticated insert into the S-Prov API on behalf of the workflow.*

Aspects of P4 also concern direct interaction with the setup and incremental customisation of the development environment, for instance based on Notebook services such as Jupyter. The KNMI is working on a new API, SWIRRL[62] (Software for Interactive and Reproducible Research Labs). This work is conducted in the context of the ENVRIFair project in close collaboration with DARE. The API automatically manages the deployment of a computational environment offering integrated notebook, execution of workflows and visualisation services. The provenance information describing the API actions for the creation and updates of the environment, the execution of data-staging workflows and the generation of repdocuble snapshots are captured within formal provenance documents. These are stored within a dedicated database and made accessible through the SWIRRL API[63]. Ultimately, the provenance data will allow users to trace the evolution of changes within the environment itself and to restore past setups. The latter action may include data and notebook pages, according to a user's needs, fostering reproducibility and sharing of research progress among peers. Aiming DARE at being fully controllable through such interactive notebooks, we foresee great potential for the integration of the services offered by the two projects, with a particular focus over reproducibility and traceability of the research progress. This will extend the period of support and amortise support over a much wider community, as required for sustainability - see §5.

[62] https://zenodo.org/record/4264852#.X6lvNdv_pSw
[63] https://gitlab.com/KNMI-OSS/swirrl/swirrl-api

4.4 Conclusions & Summary

As reported in §3 the DARE platform delivers significant new power to our user communities. It now supports a wider spectrum of user communities and a wider range of applications used by those communities. However, DARE does not have the resources to run these at scale as frequently as users require. As the DARE-enabled methods are polished ready for productive use and as courses are run to expand DARE's user community, this becomes ever more pressing. The DARE platform deployment has been made easily deployable on institutional, regional and European eInfrastructures such as EOSC, and has had a comprehensive and flexible AAI system, to accommodate local practices and those target deployment sites' security requirements (see §5.2). Consequently, users can exploit the full range of computational and data resources that they are entitled to use. This *sometimes* requires enabling relatively inexperienced operational teams to set up, run and help DARE users with a local instance of the DARE platform. The mechanisms are explained in §5.2 and their extension to the wider ENVRIfair community is reported in D3.7-8 [Spinuso & Klampanos 2020,Spinuso et al. 2018].

The three subsystems on which DARE depends are now well integrated, and are brought into a consistent security framework by the login service. Their use is made straightforward through a Python library of helper functions, which encourages research engineers to use them correctly (see §3.4).

The WaaS (§4.1) has been significantly extended to accommodate CWL workflows, piloted by both application communities. The optimisation of dispel4py workflows has been developed [Liang *et al.* 2020]. Workflows grow in complexity by accretion, revision and composition. The current workflows used by WP6 and WP7 have not yet developed that complexity, as they are at the early stages of their life. Consequently, the benefits of that optimisation are not yet demonstrable and synthetic workflows based on the WP6 were used during the research.

The DKB (§4.2) has three mature parts, the two workflow registries, the data catalogue (§4.2.4) and the sophisticated P4 with its tools and visulaisations (§4.3). The integrating and context providing aspect of the DKP (§4.2.1-§4.2.3) is a fully developed prototype with an early pilot study being conducted with WP7. There has not been time to fully explore its potential.

The provenance system, P4 (§4.3), is fully operational and will continue to develop under the aegis of ENVRIfair. It has been substantially extended with accommodation of CWL provenance and more powerful query and visualisation systems.

There is also an ethical challenge to the collaborating federations that use the DARE platform. The purpose of the platform is to help individuals, teams and wider federations pursue complex research campaigns by pooling their ideas and efforts, i.e., we must support their collaboration by facilitating cross-boundary communication and sharing. The DARE platform has a significant Computer-Supported Collaborative Working (**CSCW**) role. But that depends on individuals recognising each other in the system, acknowledging each other's contribution and respecting each other's wishes. This cannot be done if identities are hidden. DARE has not investigated this

issue, but as more dependence on provenance is developed in research communities, facilitated by DARE's powerful provenance-driven tools, we believe it will emerge as an issue in those communities. A solution may build on institutional or public identity providers.

5 Future, Sustainability and Evaluation

Sustainability is crucial for our partners using the DARE platform. They are developing new methods and working practices that depend on the platform. They would suffer severe disruption if the sophisticated software and integrated systems were not supported after the DARE project concludes. There are three aspects to achieving such sustainability:

1. *Minimising the cost and effort* required by using shared systems, standard software and careful engineering; but the cost can never be made vanishingly small - this is addressed in §5.2.
2. *Building the commitment to invest* in the required maintenance by developing expertise and advocates across the user communities - this is addressed in §5.1.
3. *Amortising the costs* widely by expanding the user communities and the number of application areas, organisations and funders who contribute. This depends on and contributes to an improved *return on investment* (RoI); a bootstrap challenge - see §5.1.

The drive for sustainability influences every aspect of DARE's work. Sustainability has long been understood as a pressing issue for software, e.g., quoting an ENVRI report accepted by DARE's two RI communities, EPOS and IS-ENES.

> "**Software sustainability**: ... The decision to depend on software is as important as the decision to depend on an instrument and it should be taken equally carefully. This will lead to an identified list of mission-critical software. Each RI ... should establish mechanisms for determining that critical list. The list should be minimised by careful use of commercial and *well-supported* open-source software. The members of the residual list of software must be maintained or replaced throughout each RI's lifetime. This requires appropriate resources, particularly software engineering staff and processes with appropriate quality controls. Wherever possible these should be met through alliances." (From Section 5.2 "*Impact on Stakeholders*" p193 [Atkinson *et al.* 2016])

One source of underfunding for sustaining critical research software is the lack of realisation of the costs involved, as most people do not have experience of software going into production and being used by multiple users, for many purposes, some not originally envisaged, running in many different and changing computational contexts. The support needed is software maintenance and provision of help to installers and users. Maintenance includes: bug fixes (~10%), accommodation of computational context changes (~50%) and enhancements (~40%).

The lack of research-software sustainability, led to the establishment of the Software Sustainability Institute (SSI)[64]. Its mission is to change the culture so that the Research Software Engineers (RSEs) making and sustaining well-engineered software are respected and resourced. SSI now delivers global leadership for this cause.

[64] Software Sustainability Institute (SSI) https://www.software.ac.uk/

Taking these viewpoints into account, we identify the critical sustainability steps. We combine the viewpoint of RIs and their user communities with the viewpoint of research developers and platform engineering teams.

Research Infrastructures[65], represented by WP6 and WP7 in DARE, need to:

1. Establish an agreed process by which they *decide on which software they will depend on*. This has to balance two factors:
    a. Research agility enabling them to explore new ideas which may depend on and develop new software, and
    b. Dependability from using established software and limiting new software to that which they know they or others can maintain.

    This requires continuous governance and operational procedures. Allowing experiments and exploration, but filtering which are carried through to production with the concomitant obligation for long-term support.
2. Develop a *mutual-respect* ethos when interacting with RSEs, expecting professionalism to develop in software and systems engineering as it does in their own discipline.
3. *Share the responsibility* of finding resources in the short, medium and longer term.
4. *Actively develop broad adoption* of the software they choose to depend on.

The builders of the DARE platform and the research developers pioneering new uses take on the following responsibilities as they proceed:

1. *Minimise the use of bespoke software* so as to reduce the sustainability burden.
    a. This requires re-engineering once requirements and solutions are understood to eliminate the effects of agile processes delivering quick solutions.
    b. This requires broadening the functionality of key elements to avoid additional elements, to take over from bespoke software and to extend amortisation.
2. Use *existing well-maintained software* whenever possible, and build any new software with well-disciplined professionalism, e.g., meeting the standards for research software established by the Software Sustainability Institute.
3. *Deliver self-sufficiency* through intellectual ramps; users start by using your provided solutions with their anticipated variations, but can, with modest effort move to more radical variations when they need to.
4. *Reduce to a minimum* while still meeting all existing and anticipated requirements, the elements in a platform, subsystem or software stack that are included in its sustainability phase, by selecting elements widely used elsewhere, avoiding duplication and weeding out those whose maintenance outweighs their benefits.
5. *Document* with guidelines, patterns, technical information and tutorials the minimum from step 4 for each role that will be involved in use or maintenance.

These considerations should guide all eInfrastructure, platform and generic tool builders, not just those building the DARE platform.

[65] Both EPOS and IS-ENES, in conjunction with their related global and long-running campaigns, recognise the importance of software for their research and have relevant resources.

Evaluation has been limited by the effects of the covid-19 pandemic, as it has not been possible to build up user-communities of significant size and then observe their behaviour and collect their views and insights. However, the rather smaller sample than we would have liked have delivered preliminary information. That is reported and analysed in §5.3. It provides preliminary evidence that users will find the DARE platform easy to use, a substantial step in delivering the power of eInfrastructure to their research and a significant aid to innovation in their research,

5.1 User communities and sustainability

DARE seeks to address needs concerning, the European domain specific e-Infrastructures (such as EPOS, IS-ENES2, ICOS65, SKA66, etc.), science and technology professionals and the "long tail of science", including research institutes, research teams, individual researchers, citizen scientists and SMEs, without them needing to be concerned with technicalities, enabling them to focus on how to improve their methods, results, synergies and innovation potential and develop data-driven services. A crucial element of DARE's sustainability is increasing DARE users' self-sufficiency by reducing technical hurdles and delivering substantial documentation and training kits.

To ensure sustainability, partners will identify exploitable DARE assets, developing a software portfolio, and assuring portability and maintainability under permissive open-source licenses or suitable copyright policies. They will provide user support on a best effort basis, fixing bugs and implementing features and notifying users about revisions. This will create user and developer communities that will invest and innovate using DARE tools.

Extending beyond the IS-ENES and EPOS ESFRIs, DARE initiated talks with additional communities seeking to develop new use-cases pertaining to Eurofusion, Nanomaterials, Atmospheric sciences, Earth Observation etc. DARE also investigated technical integration with EINFRA-21 projects and EOSC. DARE attempted to raise awareness via frequent social media presence, its web site, newsletters and events while it explored the opportunity to publish in Open Research Europe.

Finally, appropriate measures and a supportive environment on all levels must be sustained to foster the effective uptake of new technologies by all relevant economic stakeholders and to facilitate the DARE platform and DARE components as services being adopted by the industry. The aforementioned courses of action are detailed and highlighted in the deliverable "*D8.6 Sustainability, Exploitation and Commercialization Plan*" [Tsilimparis *et al.* 2020].

5.2 Individual and Combined services

Deployment and operations effort can often present an obstacle for the adoption of new platforms like the DARE toolkit. Therefore, care has been taken to design and implement the components of the DARE toolkit from the start in a way that eases the burden on IT-administrators, and that self-empowers non-expert users to run and manage a DARE deployment without the help of IT-operations experts. Using microservices facilitates the deployment of a subset of the DARE

platform, when a community chooses to use only parts of it. In these cases and for the complete platform, virtualisation on multiple levels assists with deployment.

The components, as presented in the preceding sections, are designed as loosely coupled, concise services. Communication / Interworking is realised through APIs, mostly based on REST interfaces using JSON encoding, allowing for distribution and horizontal scaling of the services (e.g. through the automatic functionalities of Kubernetes[66]). DARE is undogmatic when it comes to the microservices philosophy, though. Where it was required to directly address user needs, for example, deviations from this approach were made. For example, to support some simulation codes that use the Message Passing Interface (MPI), more tightly coupled interdependencies are required, e.g. a shared POSIX filesystem between the Kubernetes pods that takes part in a computational job. However, care has been taken to address these cases with native Kubernetes functionality, e.g. in this case with a Kubernetes Operator.

To avoid duplication of effort and sustainability challenges after the project's end, well established open source components have been used wherever possible. Where multiple implementations were available, well-known and widely used community-supported solutions have been preferred. Such components include MySQL, nginx, Virtuoso, Keycloak and Kubernetes.

APIs of the DARE components are designed to be as simple as possible, preferably as REST APIs using JSON for communication. API documentation is available in the form of OpenAPI/Swagger descriptions. Where available, (pseudo-)standards have been used, such as W3C-PROV, DCAT[67] and CWL.

As has been described before, the DARE components make use of operating-system level virtualisation. They are distributed as Containers, and their deployment is managed using the Kubernetes Container Orchestration system. Where possible, existing community-maintained container images are used to avoid replication of work. For custom containers, care has been taken to follow best-practices such as relying on community-maintained base images, carrying only necessary software in small containers running only single applications, using APIs between different applications instead of interweaving, etc, which reduces the effort for maintenance and operations. For deployment, Kubernetes descriptors have been prepared that allow selective as well as collective deployment of the DARE components.

The above described containers and descriptors allow easy deployment on existing container infrastructures, such as managed Kubernetes Clouds (e.g. Google Kubernetes Engine). On top of that, the DARE project provides tools to ease the deployment on IaaS-Clouds such as Amazon and the EOSC-provided IaaS Clouds (e.g. EGI FedCloud). By using the Terraform[68] tool, DARE can provide Infrastructure-as-code level configuration files that allow automation of deployment on various Cloud technologies, both managed and on-premise, e.g., locally installed OpenStack Clouds. For this approach, DARE relies on the work of the Kubernetes project Kubespray[69] to automate the deployment of Kubernetes on Cloud infrastructures and then deploying the DARE components on top.

Most of the DARE components can be used separately as well as in combination as a part of its design philosophy. The components should be as independent as possible, but allow for strong

[66] Kubernetes https://kubernetes.io/
[67] https://www.w3.org/TR/vocab-dcat-2/
[68] https://www.terraform.io/
[69] https://kubernetes.io/docs/setup/production-environment/tools/kubespray/

synergy effects when used in combination. For example, the provenance system can be run independently from the rest of DARE as an autonomous system. However, there are many advantages when it is used with other components. As described in §4, when used with dispel4py or CWL, for example, a lot of provenance information is collected and recorded automatically. Another such case is the Search & Discovery service, which can but does not need to make use of the information collected from the Provenance service. In this way, DARE encourages further use of its components to improve long-term sustainability.

5.2.1 Authentication and Authorisation

Including support for Authentication and Authorisation in the DARE platform is a particular challenge due to the dual objectives described above, as they result in partly contradicting requirements. To give just one example: while on the one hand, local deployments (on-premise) should be independent of external services and separate components should be usable without too many dependencies, typical requirements of community-driven hosted services go in the opposite direction. Users would like to be able to use their existing accounts to log-in and want to benefit from Single-Sign-On solutions instead of entering their credentials multiple times. For this purpose, community-driven infrastructures like the ESFRIs often employ or are on their way to implementing the strategy depicted in the AARC Blueprint Architecture[70]. Examples include DARIAH[71], EPOS[72], LifeWatch[73] and many others. This is also the model that the EOSC-portal[74] is currently using and the proposal for the EOSC AAI from the EOSC-Hub[75,76] project.

To strike a balance, WP5 has evaluated multiple standards (OpenID Connect[77], SAML2[78]) and available implementations (among them Keycloak, Unity IDM, Perun, Shibboleth) and has finally decided to implement a solution based on the Keycloak Open Source Identity and Access Management solution[79]. Keycloak is a widely supported Open Source solution with backing from Red Hat, as it forms the upstream project of their commercial Red Hat Single Sign-On solution. It allows Identity Brokering, acting as an AAI Proxy, Single-Sign On, as well as local user databases. With SAML, OAuth2.0[80] and OpenID Connect, it supports the most important Standards, allowing easy integration and wide compatibility with standards-compliant software and infrastructures (such as EOSC). Additionally, client adapters for multiple programming languages and application servers are available to facilitate integration. For the use with microservices in particular, an authenticating (reverse) proxy called Gatekeeper is available, that can be used to outsource the protocol implementation from the application to this ---proxy. Due to its popularity, a curated helm chart for Kubernetes is available and upgraded regularly on which DARE can easily rely.

On this foundation, DARE uses the OpenID Connect/OAuth 2.0 Standard with access tokens. Even though this meets the requirements nicely, and OpenID Connect has become a widely used standard, there are still challenges to be solved when using it in a scenario such as DARE. These

[70] https://aarc-project.eu/architecture
[71] https://wiki.de.dariah.eu/display/publicde/DARIAH+AAI+Documentation
[72] https://aarc-project.eu/aarc-in-action/epos/
[73] https://wiki.geant.org/display/AARC/LifeWatch+-+Pilot+Overview
[74] https://eosc-portal.eu/
[75] https://www.eosc-hub.eu/
[76] https://confluence.egi.eu/display/EOSC/Authentication+and+Authorization+Infrastructure+-+AAI
[77] https://openid.net/developers/specs/
[78] https://tools.ietf.org/html/rfc7522
[79] https://www.keycloak.org/
[80] https://tools.ietf.org/html/rfc6749

include for example, pure API access (OAuth2.0 is browser-focused) and secure delegation for long-running batch jobs (OAuth 2.0 Token Exchange was still in development, and in draft status until January 2020[81]). The DARE API now uses Keycloak's preview feature of Token Exchange with Offline Tokens to support delegation to the APIs. For increased security, the tokens can be scoped. Keycloak also allows administrators of a DARE instance to either rely on external identities, e.g. social user ids, or to fall back to purely local user accounts, which is helpful for private deployments of DARE.

5.3 Assessment of utility and usability

This section presents the evaluation of the DARE platform from the usability and utility perspectives. The evaluation was conducted in two stages:

1. A questionnaire with international students from KIT's BSc and MSc programmes in Geophysics who had used the volcanic mass transport simulation method (see §5.3.1)
2. Interviews with N research engineers and application-domain experts.

5.3.1 Evaluation with students using the volcanic pilot

This section is also in Deliverables D6.4 (§3.1.2). We advise readers who have already read it to skip to §5.3.2.

Study Design

At this stage we aimed to: 1. investigate the usability of the platform; 2. collect suggestions for improving the platform

Participants. The participants in this study were 10 BSc and MSc (in Geophysics) students from the KIT (Karlsruhe Institute of Technology).

Methods. We used a questionnaire to capture the users' perception of usability (mainly user satisfaction) and also suggestions for improving the platform regarding both usability and functionality. The questionnaire included the System Usability Scale (SUS) which consists of 10 five-point Likert scale questions [Sauro 2011]. The rest of the questions were open ended and aimed to find out what the participants like, dislike and what suggestions they have for improving the system.

Procedure. The study got ethical approval (No 34039) from the School of Informatics, University of Edinburgh. It took place at the KIT (Karlsruhe Institute of Technology) and followed up a session of student training which included using a free exploration of the DARE platform. The course's main aim was to train students in volcanic hazard assessment. At the end of the course all the students agreed to answer the questionnaire, after reading the information sheet and signed the consent form.

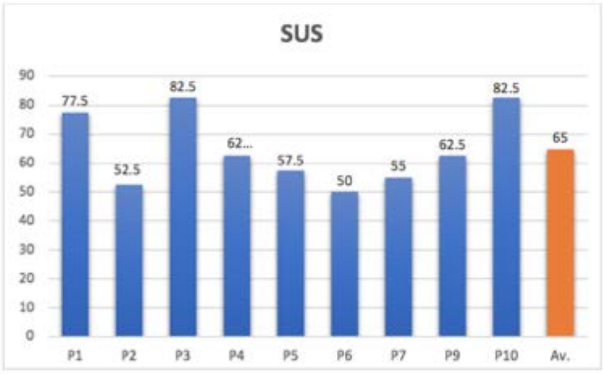
Data Collection and Analysis. The data from the SUS questionnaire were used to obtain the SUS score. This was calculated by combining the ten scores for each question according to [Bangor *et al.* 2009]. This score ranges from 0 to 100, with a score of 68 being considered as an average score. In order to be excellent, a system should score over 80.3. Data analysis of the answers in

[81] https://tools.ietf.org/html/rfc8693

the open-ended questions was inspired by thematic analysis, top-down approach [Brown & Clarke 2006]. We set up three predefined themes: *likes*, *dislikes* and *suggestions*.

Results and Discussion

After analysing the data, we excluded the data from one participant as we found that their scores in the SUS questionnaire were in contradiction with their answers in the open-ended questions. Particularly, the student was very positive in appreciating the platform, but scored the platform extremely low (i.e. 15). Therefore, we only considered the results from nine participants to this study.



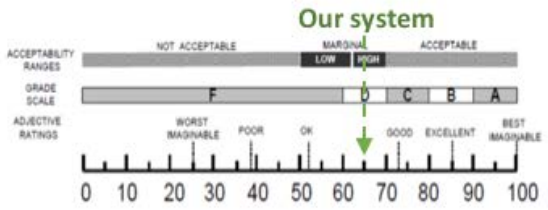


Figure 5.1: [*left*] SUS Scores; [*right*] DARE platform overall score

SUS scores on usability. The SUS scores for the nine participants ranged between 50 and 82.5 (see Figure 5.1 *left*) with an average of 65. Looking at Figure 5.1 *right*, we can see that the maximum score is very close to excellent on the adjective scale [Bangor *et al.* 2009], whereas the minimum is still OK. The average SUS score falls within the upper band of OK and is very close to what is considered the average score which puts the project at 50th percent.

Likes. Most of the students (N=5) considered the platform as easy to use. Other aspects that they liked were: no need for coding[82] (N=1), access from any device (N=1), remote access (N=1) and good structure (N=1).

Challenges. Four students commented that the system crashes too often, when too many cells run at the same time or when too many people were using it simultaneously. Several students (N=4) complained that they needed to wait a bit too long to 'get it going". Also, several students (N=4) considered the platform as being complex. Two students found the structure challenging. Also, one student remarked that some of the results look weird and one commented that the system does not provide feedback for the user's progress[83].

Suggestions. Several suggestions were directed toward improving the usability of the system and they referred to: time indication, status indication (whether a certain workflow run has finished)

[82] As they were new to DARE and in a one-day class preparing for field work, the exercise had to be completed in limited time so the set up did not require them to code; however, this is not typical of DARE use by domain experts and research engineers.
[83] Similarly, professional and experienced users would use DARE's provenance-powered tools for this.

and increasing transparency (–"*Some code blocks can be overseen because they are hidden").* Two students suggested the users would need more tutorials.

Conclusions

The overall outcome indicates a reasonably high-level of usability from the view point of new users. The challenges and suggestions identify opportunities for improvement that should be attempted if resources permit. However, these should be interpreted taking account of what the students were shown, for example, there are interactive methods and tools and access to files and logs via the provenance-powered tools developed by WP3, but they were not introduced in the training. Further detail and analysis may be found in [Constantin 2020].

5.3.2 Evaluation interviewing research engineers

Aims

The interview-study aims were to evaluate:

1. ease of use
2. user satisfaction
3. utility of the DARE platform
4. DARE impact on the speed of the engineers' responses to research requirements
5. impact of the platform on researchers' productivity
6. impact of the platform on the innovation in research community

In addition, the study aimed to collect:

1. usability and functionality issues
2. suggestions to improve the platform

The evaluation interviews aimed to assess the usability of the DARE platform covered both use cases. The analysis therefore provides an overall feedback which applies to seismological and climate-science applications. As a result, this section is also in Deliverables D6.4 (§3.2.2.2) and D7.4 (§3.3). We advise readers who have already read it to skip to §5.4.

Participants

We have recruited 6 participants, from various backgrounds. Two of them helped us to pilot the interview questions. All of them have been involved in the development of the platform to varying extents, and they all have programming skills.

Procedure

We conducted semi-structured interviews to obtain insights arising from participants' perspectives and experiences. The interview questions were developed to probe several areas of exploration, such as the ease of use of the DARE platform, its learnability, its potential for integration with other services, and for automation of research methods. They asked about data-use policies.

Initially, we piloted the interviews with two DARE team members. This tested our interview questions and interview data-collection procedure.

Data Collection and Analysis

The data has been collected online, using the Zoom platform for video-conferencing. The sessions were recorded for analysis purposes. Similar to the previous evaluation (§5.3.1), a thematic analysis top-down approach was employed. Codes were given to the participants (from P1 to P6) to maintain anonymity in reports and publications.

Results and Discussion

Here is a summary of main findings of the interviews, clustered thematically. Readers are referred to [Andries & Constantin, 2020] for a complete report.

Successful experience: overall, all the participants found that the experience of using DARE led to success. Discussing what contributed to their success, they mentioned that developing targets for use cases was accomplished (N=2)[84], and the structure of the platform in itself was deemed to be versatile and functional, because it "*hides some of the complexity of the execution*" (P3). Its components have also been mentioned as contributing to a successful experience with the platform, primarily the workflows that are already set up (N=4), or the fact that DARE sets up the environment with all the libraries that the user may need (P6).

Ease of use: the participants generally found the platform easy to use (N=5). We employ caution when presenting this result as all interviewees were in some way involved in its development. They were able to identify potential areas for improvement, and suggestions for additional features and training to improve the use of the platform.

Less successful platform features: the participants discussed these in terms of features that might be missing and could be added, referring to the lack of maturity of the platform, rather than unsuccessful features. Examples included the lack of a user interface, for finding files and logs (P4). Instead they had to use the DARE API for that purpose (P4). The difficulty of installation (P5), the potential assumption that the users should have some cloud computing knowledge to use the platform (P6), as well as computational and programming skills (P3). There was confusion around the provenance and the API documentation regarding types of data (P2).

Training needed for using the platform: all the participants agreed that training would be beneficial for the users and developers. More specifically, an initial training stage was described as essential as the platform has plenty of functionality to offer, and the opportunity and support to explore that in depth should be provided (N=2). Perhaps some training, more specifically aimed at individuals who may not have a background in computer science (N=2), to provide support with the development of workflows and helper functions would be useful. Providing examples which can be sorted by functions, objectives, etc. was also mentioned (P5)). Videos to explain how to register an application, and videos to introduce each aspect of the platform's functionality (P6).

Responsibility for knowledge transfer: all the participants agree that the developers have the initial responsibility for providing support (e.g. to organise webinars). A collaboration between

[84] Indicates number of interviewees that gave this response, 2 in this case.

research engineers and domain-specific engineers was also mentioned (N=2). Lastly, one of the participants suggested a community forum could be developed over time, similar to Stack Overflow, provided that enough members would actively engage.

Productivity and innovation in communities: all the participants agreed that by using the platform, the productivity of the users would increase. This was motivated by explaining that the platform is aimed at reducing the engineering time by hiding the technical details (N=3), allowing the users to spend more time developing their specific applications and, consequently, on research. More specifically, the platform can be used via an API call, not needing to worry about the workflows or the infrastructure (N=2). Most of the participants (N=5) considered that, by reducing the time previously spent on infrastructure complexities, the platform did accelerate innovation in the users' communities, by enabling them to focus more on research.

Integration with external and local services: all the participants considered that the platform integrated very well with both external and local services, given the services that were tested so far. Examples of good integration included the European seismic archives which could be downloaded for use (P3). DARE was described as modular and independent thanks to Kubernetes (P4) and well connected to the climate computational resources infrastructure (P5). DARE was also described as one of the first platforms to allow the communities to transfer into a cloud, which should be the future of operations (P6).

Automation of research methods and development practices: the participants knew that automation was one of the goals of DARE (to hide some technical details) and they all believe that this was achieved to some extent. More specifically, all resources are shareable between users, by enabling the use and sharing of workflow systems, and large-scale parallelisation without the users' input.

Additional automation and functionality can be added to the platform, depending on community's needs (N=2), e.g., graphical interfaces could help beginners (P2), the addition of a desktop version with the same libraries, for local deployment (P6), as well as a visual representation of the user's repository (P6) may be helpful.

Summary and caveat

We would have recruited more responders and conducted more interviews had face-to-face events been possible. The participants supported the view that the platform is usable, and that it would be easy to use (depending on the users' background and skills). The platform supports automation of some common practices, and it allows additional functionality to be added. Training should be provided by developers in the early stages and extended by the communities themselves, to facilitate understanding of the different opportunities enabled by the platform. Some of the replies indicate limitations in the training, e.g., regarding difficulty in finding files and logs when the data catalogue (§4.2.4) and provenance tools (§4.3) provide these facilities. However, readers are warned that all the interviewees were or had been members of the DARE project, so these results should be treated with caution. Those who had developed a part of the system were unaware of tools and functions that would be familiar to application experts or

research engineers. The study at KIT (§5.3.1) offered significant value as it was with entirely new users with no prior engagement with DARE. A key target in each research community to stimulate induction, so that the active community achieves a critical mass.

5.4 Summary and Conclusions

Sustainability is a crucial issue for DARE, its current and future users and for others who develop sophisticated research environments needed to meet pressing local and global challenges. The mid-term review reinforced its importance. We started §5 by clarifying this sustainability requirement and identifying the critical issues that needed to be addressed.

One of these, the need to deliver sufficient benefits to justify the cost of sustainability, has been demonstrated by the two pilot communities: solid-Earth scientists (§3.1) and climate scientists (§3.2). Their ability to respond rapidly to new issues and to exploit new data and new computational power depends on equipping the research engineers with tools that help them innovate easily (§3.3) and diagnose the causes of technical issues (§4.3).

Sustainability depends on keeping the support-costs sufficiently small (support includes: software and system maintenance, transfers to new technologies, community support and enhancements). The systematic use of standard and widely used subsystems in the platform is key to this (§5.2). This reduces the amount of software the DARE community has to take sole responsibility for. That residual “upper middle-ware and tuned tools” that future DARE communities have to support remains essential for the new ways of working made possible by DARE.

That cost has to be amortised over sufficiently large communities that recognise the platform's utility. Two aspects contribute to this. The outreach and events build those communities and convince them of DARE’s value - a key sustainability step (§5.1). The ability to deploy widely and to fit in with existing communities is also crucial. The arrangements for deployment and coexistence with other services (§5.2) and particularly the flexibility but strength of AAI and security (§5.2.1) are key to this. A noteworthy step in this direction is the use of P4 in ENVRIplus (§4.3).

Usability, measured in terms of ease of use, user satisfaction and learnability. Critically, the ability to incrementally learn how to exploit the new power delivered to researchers and their engineers, enables the use to grow and warrants the investment needed to sustain the platform. The limited evaluation that has been possible (§5.3) suggests that this may be achieved for DARE’s targeted application-domain experts and research engineers. Further work is needed to broaden the usability evaluation to less experienced and less computer literate members of the research communities. The two initial communities were chosen because they had such expertise as it allowed DARE to focus on the key challenge of delivering agility and power to research engineers.

6 Summary, Vision and Impact

The power and usability of the DARE platform and its supported applications is demonstrably significant (§3). It yields substantial improvements in productivity for application-domain experts from improved abstraction. It delivers agility, faster responses to new requirements and opportunities, by delivering accelerated rates of innovation by research engineers from the combination of powerful tools and adaptive mappings to target platforms. This is fully reported by the application communities in D6.4 [Magnoni *et al.* 2020b] and D7.4 [Pagé 2020].

Substantial progress has been made in the final year greatly enhancing many critical factors that combine to deliver the substantial advances in usability, power, sustainability and flexibility. A few are highlighted here.

1. Integration via the DARE manager and the provision of helper functions improving usability (§3.3).
2. Introduction of a flexible and powerful AAI framework via the login manager (§5.2.1) enabling DARE to fit into a wide variety of security regimes using a variety of e-Infrastructure provisions.
3. Automation of container deployment, configuration and orchestration exploiting Kubernetes, reducing complexity while delivering power to research engineers (§5.2).
4. Incorporation of the W3C CWL scientific workflows, widening the communities that will find it easy to migrate to the DARE platform (§4.1).
5. Extension of the provenance-driven tools to deliver more sophisticated user controls and bundled actions to the selected entities (§4.3).
6. The systematic support for Jupyter notebooks, which are in widespread use throughout DARE's target communities (§4.3).

These more than meet the objectives for the final year set out in §6 of ID2.2 [Atkinson *et al.* 2020].

During the same period, exploration of new technologies and pilot implementations have prepared the way for further advances.

1. The dynamic optimisation of fine-grained data-intensive workflows encoded in dispel4py delivers very substantial performance improvements, responsiveness to data-dependent computational costs and scalability potential (§4.1) [Liang *et al.* 2020].
2. The support for user or community controlled contexts in the DARE knowledge base that will help communities manage complexity and allow rapid innovation and experiments to coexist with stable and protected information on which a domain's professionalism depends (§4.2.5).

These will ensure that the DARE platform is ready to accommodate the challenge of the three extremes as its user communities grow and their research develops.

Sustainability is absolutely critical for that potential to be realised and for the investment by DARE's user and developer communities to pay off. Essential steps have been taken, particularly with respect to software and system engineering, deployability, and versatility (§5.2). A key issue is self-sufficiency. An experiment using the volcanology mass-transport simulation as a test case (§3.1.2) provided good evidence of the potential, but revealed some aspects of developing a new

use where help from system, software or data experts was still needed. There will always be residual issues where help is needed, but these should be minimised. Further steps in usability (see above), documentation and recorded training have addressed many of the limits to self-sufficiency.

The DARE platform is a powerful composition of a growing number of microservices (§5.2). The developing research engineer and application expert DARE usage patterns are growing and have increasingly ambitious targets and sophisticated methods, as we intended at the outset of the project. Research leaders have to develop plans, find resources and steer their communities to build on the DARE platform and approach. This needs to harness broad alliances. We see potential for these through EPOS, IS-ENES and IPCC, ENVRI-FAIR and EOSC, with concrete steps having already taken place towards some of them, in line with the *DARE Sustainability Strategy & Roadmap, internal document, 30/1/2020*. DARE already has momentum in each of the large collaborative endeavours, while at the same time it is readily usable in local and institutional installations using their system support and resources.

DARE has pioneered a new approach to supporting demanding collaborative research which will sustain many communities as they combine forces to address today's most pressing data-intensive and computationally demanding challenges.

Acknowledgements

The DARE project is a Horizon2020 project, 777413, funded by the European Union. It builds on a succession of prior EU projects, including: ADMIRE, VERCE, ENVRI, ENVRIplus, and on nationally funded projects. It depends on the coordination of e-Infrastructures by EOSC. The authors thank the research software engineers and systems teams that have delivered and tested a succession of releases of the platform, and the research developers and application-domain scientists who have worked with them to shape and demonstrate the power of DARE's approach and products.

Bibliography

[Aki & Richards 1980] K. Aki & P. G. Richards, *Quantitative Seismology, Theory and Methods. Volume I and II,* 1980.

[Anselemi & Gaujai 2009] Anselmi, J., Gaujal, B.: *Performance evaluation of work stealing for streaming applications*. In: Abdelzher, T., Raynal, M., Santoro, N. (eds.) OPODIS 2009. LNCS, vol. 5923, pp. 18–32.Springer, 2009.

[Atkinson *et al.* 2020] Malcolm Atkinson, Rosa Filgueira, André Gemünd, Vangelis Karkaletsis, Iraklis Klampanos, Antonis Koukourikos, Amélie Levray, Mike Lindner, Federica Magnoni, Christian Pagé, Andreas Rietbrock, Alessandro Spinuso, Chrysoula Themeli, Xenofon Tsilimparis and Fabian Wolf, *DARE Architecture and Technology internal report*, ID2.2, March 2020 DOI 10.5281/zenodo.3697898 URL https://doi.org/10.5281/zenodo.3697898

[Atkinson *et al.* 2020a] Malcolm Atkinson, Amélie Levray and Rui Zhao, *DKB Design*, in preparation, 2020. URL https://docs.google.com/document/d/1hCyoqeB8R0Ov5ZcBZVxyCOAiOST7QEP90n37PnghxGs/edit?usp=sharing

[Atkinson *et al.* 2019] M.P. Atkinson, R. Filueira, I. KLampanos, A. Kourkourikos, A. Krause, F. Magnoni, C. Pagé, A. Rietbrock and A. Spinuso, *Comprehensible control for researchers and developers facing data challenges*, proc. IEEE eScience conf. 2019.

[Atkinson *et al.* 2018] M.P. Atkinson, E. Casaroti, Ewering, R. Filgueira, A. Gemünd, I. Klampanos, A. Koukourikos, A. Krause, F. Magnoni, Pagani, C. Pagé, A. Rietbrock, A. Spinuso and C. Wood, *DARE Architecture & Technical Positioning*, Deliverable D2.1 DARE project. URL https://zenodo.org/record/2613550#.Xe-l5r_grUb

[Atkinson *et al.* 2016] M.P. Atkinson, Alex Hardisty, Rosa Filgueira, Cristina Alexandru, Alex Vermulen, Keith Jeffery, Thomas Loubrieu, Leonardo Candela, Barbara Mangagna, Paul Martin, Yin Chen, Margareta Helström, *A consistent characterisation of existing and planned RIs*. Deliverable 5.1 of the ENVRIplus project URL http://www.envriplus.eu/wp-content/uploads/2016/06/A-consistent-characterisation-of-RIs.pdf

[Atkinson & Filgueira 2020] *Planning dispel4py developments*, in preparation, 2020. URL https://drive.google.com/open?id=1N6UGU8J47FHpnWaLTNpRLCAFguGIQYmanmO4pyB8ixM

[Bangor *et al.* 2009] Bangor, A., Kortum, P., & Miller, J.. *Determining what individual SUS scores mean: Adding an adjective rating scale*. Journal of usability studies, 4(3): 114-123, 2009.

[Bell *et al.* 2013] Bell, Ray & Strachan, Jane & Vidale, P.L. & Hodges, Kevin & Roberts, Malcolm. (2013). Response of Tropical Cyclones to Idealized Climate Change Experiments in a Global High-Resolution Coupled General Circulation Model. *Journal of Climate*. **26**. 10.1175/JCLI-D-12-00749.1.

[Braun & Clarke] Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2): 77-101.

[Casarotti *et al.* 2019] Casarotti E., Magnoni F., Pagé C., Filgueira R., Klampanos I., *D8.4 Training and Consulting Report I*, DARE D8.4, July 2019.

[Constantin 2020] Aurora Constantin, *Report on the first usability evaluation of DARE platform (24th July 2020)*, DARE technical report, https://doi.org/10.5281/zenodo.430971

[Andries & Constantin 2020] Valentina Andries and Aurora Constantin, *Evaluating user experience with the DARE platform*, DARE technical report, 2020, URL

[Corcho 2019] Oscar Corcho, *Management of versions of ontologies*. Personal communication, 2019

[Doltz *et al.* 2018] Dolz M.F., Del Rio Astorga, D., Fernández J., Garcia, J.D. and Carretero, J., *Towards automatic Parallelization of stream processing applications*, IEEE Access, Vol. 6, pp 39944-39961, 2018. DOI 10.1109/ACCESS.2018.2855064. URL https://doi.org/10.1109/ACCESS.2018.2855064.

[DXWG 2020] W3C Data Exchange Working Group, *Data Catalog Vocabulary (DCAT) - Version 2 W3C Recommendation* 04 February 2020, URL https://www.w3.org/TR/vocab-dcat-2/

[Edwards 2013] Edwards, P.N., *A Vast Machine: computer models, climate data and the politics of global warming*, MIT press, 2013.

[Filgueira *et al.* 2017] Rosa Filguiera, Amrey Krause, Malcolm Atkinson, Iraklis Klampanos and Alexander Moreno, dispel4py: *A Python framework for data-intensive scientific computing*, The International Journal of High Performance Computing Applications 2017, Vol. 31(4) 316–334. DOI: 10.1177/1094342016649766. URL https://journals.sagepub.com/doi/pdf/10.1177/1094342016649766

[Filgueira *et al,* 2016] R. Filgueira, R. Ferreira da Silva, A. Krause, E. Deelman, and M. Atkinson, *Asterism: Pegasus and dispel4py hybrid workflows for data-intensive science*, in 7th International Workshop on Data-Intensive Computing in the Clouds (DataCloud'16), 2016, p. 1–8.

[Folch *et al.* 2009] A. Folch, A. Costa, & G. Macedonio 2009*. FALL3D: A computational model for transport and deposition of volcanic ash, Computers & Geosciences, 35.6: 1334-1342.*

[Frigo *et al.* 1998] M. Frigo, C. E. Leiserson, and K. H. Randall. *The implementation of the cilk-5 multithreaded language*. In Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '98, pages 212–223, ACM 1998

[Garijo *et al.* 2019] Daniel Garijo, Maximiliano Osorio, Deborah Khider, Varun Ratnakar and Yolanda Gil, *OKG-Soft: An Open Knowledge Graph for Describing, Composing and Reusing Software*, IEEE eScience Conf., 2019

[Klampanos *et al.* 2020] Iraklis Angelos Klampanos, Chrysoula Themeli, Alessandro Spinuso, Rosa Filgueira, Malcolm Atkinson, André Gemünd and Vangelis Karkaletsis, *DARE Platform: a Developer-Friendly and Self-Optimising Workflows-as-a-Service Framework for e-Science on the Cloud*, Journal of Open Science Software, 5(54), 2664. https://doi.org/10.21105/joss.02664 2020

[Klampanos *et al.* 2019] Iraklis Angelos Klampanos, Athanasios Davvetas, André Gemünd, Malcolm Atkinson, Antonis Koukourikos, Rosa Filgueira, Amrey Krause, Alessandro Spinuso, Angelos Charalambidis, Federica Magnoni, Emanuelé Casarotti, Christian Pagé, Mike Lindner and Vangelis Karkaletsis, *DARE: A Reflective Platform Designed to Enable Agile Data-Driven Research on the Cloud*, in BC2DC workshop proc. IEEE eScience conf., 578-585, 2019. DOI 10.1109/eScience.2019.00079

[Klampanos *et al.* 2015] Klampanos, Iraklis Angelos, Martin, Paul, & Atkinson, Malcolm. *Consistency and Collaboration for Fine-Grained Scientific Workflow Development: The dispel4py Information Registry*. Zenodo. http://doi.org/10.5281/zenodo.3361395.

[Lannon *et al.* 2020] Larry Lannom, Dimitris Koureas, and Alex R. Hardisty, *FAIR Data and Services in Biodiversity Science and Geoscience*. Data Intelligence 2 (2020), 122–130. doi: 10.1162/dint_a_00034

[Levray 2020] Amélie Levray, *DARE Knowledge Base User Manual,* DARE technical report. 2020. URL https://docs.google.com/document/d/1u62251KnRURztDyBbxo77yfGGBpjxjzlWBlSinArxX0/edit?usp=sharing

[Liang *et al*. 2020] Liang Liang, Filgueira Rosa and Yan Yan, *Adaptive Optimizations for Stream-based Workflows*, in proceedings WORKS 2020, Oct. 2020.

[Magnoni *et al.* 2020a] Magnoni F., Grund M., Spinuso A. and Rietbrock A., *D6.2 Requirements and Test Cases II*, DARE D6.2, July 2020.

[Magnoni *et al.* 2020b] Magnoni F., Lindner M., Gottschämmer E., Rohnacher A., Constantin A., Andries V., *D6.4 Pilot Tools and Services, Execution and Evaluation Report II*, DARE D6.4, December 2020.

[Magnoni *et al.* 2020c] Magnoni F., Pagé C., Gottschämmer E., Lindner M., Tsilimpari X., Galifianaki E., Constantin A., Andries V., *D8.5 Training and Consulting Report II*, DARE D8.5, December 2020.

[Magnoni *et al.* 2019a] Magnoni F., Casarotti E., Artale P., Lindner M., Rietbrock A., Klampanos I., Davvetas A., Spinuso A., Filgueira R., Krause A., Atkinson M., Gemund A., Karkaletsis V., *DARE to Perform Seismological Workflows*, IN13C-0726, American Geophysical Union, Fall Meeting 2019.

[Magnoni *et al.* 2019b] Magnoni F., Casarotti E., Davvetas A., Klampanos I., *D6.3 Pilot Tools and Services, Execution and Evaluation Report I*, DARE D6.3, July 2019.

[Martin *et al.* 2019] Paul Martin, Laurent Remy, Maria Theodoridou, Keith Jeffery, and Zhiming Zhao, *Mapping heterogeneous research infrastructure metadata into a unified catalogue for use in a generic virtual research environment*. Future Generation Computer System, 101, 1–13. http://doi.org/10.1016/j.future.2019.05.076

[Mattheis *et al.* 2012] Sebastian Mattheis, Tobias Schuele, Andreas Raabe, Thomas Henties and Urs Gleim, *Work Stealing Strategies for Parallel Stream Processing in Soft Real-Time Systems*, in proc. Architecture of Computing Systems -- ARCS 2012, 172-183, Springer 2012

[Myers *et al.* 2015] James Myers, Margaret Hedstrom, Dharma Akmon, Sandy Payette, Beth A Plale, Inna Kouper, Scott McCaulay, Robert McDonald, Isuru Suriarachchi, Aravindh Varadharaju, Praveen Kumar, Mostafa Elag, Jong Lee, Rob Kooper and Luigi Marini, *Towards sustainable curation and preservation: The SEAD project's data services approach*, in: e-Science, IEEE pp. 485-494, 2015.

[Navarro *et al.* 2009] Navarro, A., Asenjo, R., Tabik, S., Cascaval, C. *Analytical modeling of pipeline parallelism*. In: International Conference on Parallel Architectures and Compilation Techniques (PACT).IEEE (2009)

[Pagé 2020} Christian Pagé, D7.4 Pilot Tools and Services, Execution and Evaluation Report II, DARE deliverable 7.4, Dec. 2020 URL https://docs.google.com/document/d/1bzcwKCaI-i9u3XDScH6873kXPD9EHPOCM0G__-LQ6JE/edit#

[Pagé 2019] Christian Pagé, *D7.3 Pilot Tools and Services, Execution and Evaluation Report I*, DARE deliverable 7.3, July 2019. URL https://docs.google.com/document/d/1Y9Ap10SM5RvyL3a91HksYl17pED-c0mpz0mantGbInE/edit?usp=sharing

[Pagé *et al.* 2019a] Pagé, Christian; Plieger, Maarten; Som De Cerff, Wim; De Vreede, Ernst; Drost, Niels; Klampanos, Iraklis Angelos; Karkaletsis, Vangelis; Atkinson, Malcolm and Pivan, Xavier. *Climate Data Access: Re-thinking our Data Analysis Workflows*, poster, in Proc. Scientific Gateways Conference (2019), DOI: 10.17605/OSF.IO/T8Y3H URL: https://zenodo.org/record/3546232#.Xfeh4S2ZOUk

[Pagé *et al.* 2019b] Pagé, Christian; Som de Cerff, Wim; Plieger, Maarten; Spinuso, Alessandro; Pivan, Xavier, *Enabling Transparent Access to Heterogeneous Architectures for IS-ENES climate4impact using the DARE Platform*, in Proc. IEEE eScience Conf. (2019) DOI: 10.17605/OSF.IO/WKM93, URL: https://zenodo.org/record/3546219#.XfelrS2ZOUk

[Python IDEs 2019] The Python IDEs, *The Python Language Reference* URL https://docs.python.org/3/reference/

[Ramakrishnan 2018] V. Ramakrishnan, Gene Machine, Oneworld Publications, 2018.

[Rietbrock *et al.* 2018] Andreas Rietbrock, Federica Magnoni and Emanuele Casorotti, Alessandro Spinuso and André Gemünd, *D6.1 Requirements and Test Cases I*, DARE D6.1, July 2018. URL https://drive.google.com/open?id=1ZISbDjphDR7gYNiQ24TQOM-npKxx169A.

[Rodriuez & Buyya 2018] Maria A. Rodriguez and Rajkumar Buyya, *Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms*, Future Generation Computer Systems, **79**, 2018, 739-750

[Roth *et al,* 2020] Malin Roth, Horst Schwichtenberg and André Gemünd (eds), *Platform Infrastructure, Usage & Deployment II,* DARE deliverable D5.2, December 2020.

[Rule, Adam, *et al*. 2018] "T*en simple rules for reproducible research in Jupyter notebooks.*" *arXiv preprint arXiv:1810.08055* (2018).

[Sauro 2011] Sauro, J. *A practical guide to the system usability scale*. Denver: Create Space, 2011.

[Spinuso *et al.* 2020] Alessandro Spinuso, Chrysoula Themely, Iraklis Klampanos, D3.8 *Integrated Monitoring and Management Tools II.* DARE deliverable D3.8, December 2020 URL https://docs.google.com/document/d/1vRy0R-E2waESqTd7FIkY1N4yzu4m05d4x71UNJFrAcs/edit?usp=sharing

[Spinuso & Klampanos 2018] Alessandro Spinuso and Iraklis Klampanos, *D3.7 Integrated Monitoring and Management Tools I,* DARE Deliverable D3.7 2018, http://project-dare.eu/wp-content/uploads/2019/03/D3.7-Integrated-Monitoring-and-Management-Tools-I_final_draft.pdf

[Spinuso *et al.* 2019] Alessandro Spinuso, Malcolm Atkinson and Federica Magnoni, *Active provenance for Data-Intensive workflows: engaging users and developers*, Proceedings of the BC2DC workshop IEEE eScience conf. 2019. URL https://bc2dc.github.io/presentations/ActiveProvenanceASpinuso.pdf **Replace with DOI**

[Spinuso & Klampanos 2018] Alessandro Spinuson and Iraklis Klampanos, *Integrated Monitoring and Management Tools*, D3.7 DARE project deliverable URL https://drive.google.com/open?id=1QRDAQOoyPX13Uue_L3c_WnLNhapQlD3z

[Trani 2019] Trani L. *A methodology to sustain common information spaces for research collaborations*, PhD thesis, University of Edinburgh, 2019.

[Trani *et al.* 2018] L. Trani, R. Paciello, M. Sbarra, D. Ulbricht and the EPOS IT Team, *Representing Core Concepts for solid-Earth sciences with DCAT – the EPOS-DCAT Application Profile*, Geophysical Research Abstracts 2018.

[Trani *et al.* 2018] Trani L., Atkinson M.P., D. Bailo, R. Paciello, Filgueira, R., *Establishing core concepts for information-powered collaborations*, Future Generation Computer Systems 89, 421–437, 2018.

[Trani *et al.* 2018] Trani, L., Paciello, R., Bailo, D., and Vinciarelli, V. (2018). EPOS-DCAT-AP: a DCAT Application Profile for solid-Earth sciences. In 2018 Fall Meeting AGU. Abstract IN31B-33.

[Tsilimparis et al. 2020] Xenofon Tsilimparis, …, *D8.6 XXXX, DARE deliverable D8.6, December 2020.*

Appendix 1 Abbreviations and Definitions

Table A1.1: *Abbreviations used in this document*

Abbreviation	Meaning
§	Section or paragraph
AAI	Authentication Authorisation and Identity
API	Application Programming Interface. the means by which software and developers use the capabilities a software subsystem or services offers
C3S	Copernicus Climate Change Service a service run by Copernicus
C4I	Climate for Impact a service run by IS-ENES
CMIP	Coupled Model Intercomparison Project (IPCC, 2018: Annex II: Acronyms)
CSCW	Computer-Supported Collaborative Working
CWL	Common Workflow Language a W3C standard https://www.commonwl.org/
DEM	Digital Elevation Models (topography information)
DCAT	Data Catalogue, a W3C standard describing the content of data catalogues
DKB	DARE Knowledge Base an open-ended place to leave and access information
DXWG	Data eXchange Working Group a W3C group developing a vocabulary to describe data, DCAT
ECMWF	European Centre for Medium-Range Weather Forecasts
EPOS	European Plate Observing System
ERA5	Data provided by ECMWF which contains hourly estimates of a large number of atmospheric, land and oceanic climate variables
ETOPO1	ETOPO1 is a 1 arc-minute global relief model of Earth's surface that integrates land topography and ocean bathymetry.
FALL3D	Ashfall 3D simulation code (original Fortran code)
FALL3DPy	Ashfall 3D simulation code (python port of the original Fortran code written in context of a Master`s Thesis at GPI-KIT)
FDSN	Federation of Digital Seismometer Networks that deploy seismometers on a long-term basis to collect and make available their wave-form observation time series
IS-ENES	InfraStructure for the European Network for Earth System Modelling
KB	Knowledge Base an organised repository of information used by people and software

LOD	Linked Open Data used to represent the semantic web
MIP	
MPI	Message Passing Interface used in HPC systems for parallelisation
MS	Mile Stone that marks project or research campaign progress
MT3D	Moment Tensor in 3D, a seismological method
NetCFD	Network Common Data Form https://www.unidata.ucar.edu/software/netcdf/
NOAA	National Oceanic and Atmospheric Administration
P4	Protected Pervasive Persistent Provenance a means of recording what has been done
PE	Processing Element, a computational process or processes forming part of a data-streaming workflow
RaaS	Reproducability-as-a-Service the collection and use of provenance to facilitate repeating a computational experiment or analysis
RA	Rapid Assessment a seismological method estimating ground motion
RDF	Resource Description Framework (RDF) a W3C standard for the semantic web
Registry	The dispel4py Information Registry that manages information about dispel4py workflows and their component PEs
RI	Research Infrastructure computational, storage and networking facilities to support research or domain specific facilities to support research
RoI	Return on Investment the value obtained compared with the effort or cost needed
RSE	Research Software Engineer a designation of competence awarded by SSI
SSI	Software Sustainability Institute https://www.software.ac.uk/
SWIRRL	Software for Interactive and Reproducible Research Labs (ENVRIfair project)
UTC	Coordinated Universal Time https://en.wikipedia.org/wiki/Coordinated_Universal_Time
VC	Volcanology Test Case
WaaS	Workflows-as-a-Service an automated support for authoring and using formalised methods
WMS	Workflow Management System that supports developing and running workflows
WPS	Web Processing Service an OGC developed standard for geospatial data
URI	Universal Resource Indicator a W3C standard
URL	Universal Resource Locator a W3C RDF-related standard
URM	User Reference Manual