H2020-EINFRA-2017

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

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

DARE Architecture & Technical Positioning

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D2.1 Dare Architecture and Technical Positioning
Work package	WP2: Architecture Specification and Innovation
Tasks involved	T2.1: Architecture Specification, Tools and Components
Type	R: Document, report
Dissemination Level	PU = Public
Due Date	30/11/2018
Submission Date	31/12/2018
Status	Draft V1.0
Editor(s)	Malcolm Atkinson (UEDIN)
Contributor(s)	Malcolm Atkinson (UEDIN) Emanuele Casarotti (INGV) Malin Ewering (FRAUNHOFER) Rosa Filgueira (UEDIN)

	André Gemünd (FRAUNHOFER) Iraklis Klampanos (NCSR-D) Antonis Koukourikos (NCSR-D) Amrey Krause (UEDIN) Federica Magnoni (INGV) Andrea Pagani (KNMI) Christian Pagé (CERFACS) Andreas Rietbrock (KIT) Alessandro Spinuso (KNMI) Chris Wood (UEDIN)
Reviewer(s)	Antonis Koukourikos (NCSR-D) Andreas Rietbrock (KIT)
Document description	An architecture to shape an integrated, extensible, flexible framework for supporting DARE applications, balancing immediate priorities with long-term considerations. This will lead to a clarification of the currently available components and technologies, together with a plan for developing these further over the next twelve months. This builds on two internal deliverables: ID2.1-M4 and ID2.1-M8.

Document Revision History

This history builds on and follows the history of ID2.1-M4 and ID2.1-M8. The contributors above are accumulating over the three iterations leading to D2.1.

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
ID2.1-M8 V0.1	27/06/2018	Initial structure	M. Atkinson (UEDIN)
ID2.1-M8	29/07/2018	Shared via Office 365	Ready for ArchTF call and to enable concurrent update merger.
ID2.1-M8	19/09/2018	Revised structure added detail	Ready for ArchTF call 19/09/2018 to explain new sections and extension to include benefits.
ID2.1-M8	06/10/2018	Final version	Internally released for review and comment. Approved by 7th ArchTF meeting 02/11/2018 with agreed plan of changes for D2.1.
D2.1Draft1	02/11/2018	Initial version + plan	Shared draft for all in ArchTF to edit according to an agreed plan of suggested changes and schedule supplied as a companion document.
D2.1Draft2	29/11/2018	Integrated set of amendments	Multiple inputs from contributing authors plus responses to the meetings at Demokritos and KNMI considering the technological-architectural interplay.
D2.1Draft3	05/12/2018	Editorial	All sections sent for internal review
D2.1Draft4	17/12/2018	Respond to reviews	Improvements to Section 8.3.2 and responses to suggestions from two internal reviewers plus revised Section 11.

Executive Summary

This document is designed to help shape the work of the DARE project, project-dare.eu. It addresses long-term strategic issues and technical opportunities while clarifying these using immediate requirements of the agile task forces and development teams. This is the third iteration and the first formal deliverable. It develops those viewpoints together for DARE. This document has been produced by the combined efforts of the Architectural Task Force (ArchTF) – see below – with help from other DARE colleagues.

DARE's architecture is intended to provide a framework and design pattern for many future applications that need to extend their capabilities in handling the scale of data, the computational challenges or the complexities inherent in their work and federations. These federations collaborate to address clusters of research and analytic challenges. They grow in size and diversity, in geographic distribution and recruited skills to meet their challenges. Their success depends on effectively sharing information, methods and resources; pooling skills, knowledge and insights to meet progressively more demanding challenges. DARE's architecture will help them do this and will deliver its benefits throughout long-running research campaigns. The crucial ideas and initial versions will be tested by addressing the requirements of DARE's two user communities: computational seismologists in EPOS and climate-impact modelling in IS-ENES.

The ArchTF will consider operational issues, prior investments, established working practices and external guidance. As such it must reconcile conflicting requirements, acknowledge constraints and resource limitations and still be ambitious. This requires that the architects facilitate a conversation with trusted representatives of *all* of the stakeholders. The architects' vision should shape these discussions but all stakeholders: user communities, resource providers and technology experts, should feel that their input and concerns are properly considered. The four-monthly cycles developing editions of this document are intended to stimulate the required discussions and to provide a framework from which significant decisions emerge. This is the third edition and it considers the next 12 months (2019) of DARE R&D.

The primary goal of each version of the architecture deliverable is to identify the best plan for the next cycle of DARE development. This should lead to the allocation of available resources to make the greatest amount of progress possible when considering the *priority* requirements of the two user communities, EPOS Computational Seismology (WP6) and IS-ENES/Climate4Impact (WP7) and the need for sustainable good quality software and systems engineering to ensure the delivered platforms are sustainable.

The secondary goal of the architectural reports is to achieve a significant long-term impact from DARE. This requires that future users of DARE products in new application domains should be considered. Their research campaigns will demand capacity for expansion in all three dimensions that DARE addresses: data scale, computational scale and complexity. This requires sustainable software and methods that provide a platform for exploiting future technologies. The cost of sustainability will be reduced by alliances with suitable computational and data-intensive R&D. These alliances will amortise future support, maintenance and development over wider communities. However, a residual component of DARE's innovations will require their own sustainability plan.

Table of Contents

List of Figures

List of Tables

List of Terms and Abbreviations

Abbreviation	Definition
AAAI	Authentication, Authorisation, Accounting and Identity
ADS	Application Domain Scientist
AFS	Archive FAIR Service
AGU	American Geophysical Union
API	Application Programming Interface
ArchTF	Architectural Task Force
ATF	Agile Task Force
AQ	Architectural Question that needs resolving in each application context and should have consistent answers
BDI	Big Data Europe Integrator
BDV	Bulk Dependency Visualiser (used with provenance data)
C3S	Copernicus Climate Change Service
C4	Common Conceptual Core Catalogue a way of communicating about everything consistently
C4I	climate4impact
CERFACS	Centre Européen de Recherche et de Formation Advancée en Calcul Scientifique
CF-Convention	Climate and Forecast Metadata Convention
CMIP5	Coupled Model Intercomparison Project Phase 5
CMOR	Climate Model Output Rewriter
CSCW	Computer-Supported Collaborative Working
CWT	Compute Working Team
d4p	dispel4py
da	DARE Action (used as a da_ prefix for Python methods)
DAP	Data Access Protocol
DC	DARE Concept (used as a DC_ prefix for Python classes)
DCAT	Data Catalogue Vocabulary
DEM	Digital Elevation Model
EGI	European Grid Initiative
EGU	European Geophysical Union
EIDA	European Integrated Data Archive
ENVRI	Environmental Research Infrastructure
EoS	End of Stream (in data-streaming workflows)
EOSC	European Open Science Cloud
EPOS	European Plate Observing System
ERM	ENVRI Reference Model
ES	Ensemble Simulation of seismic events
ESFRI	European Strategic Forum on Research Infrastructures
ESGF	Earth System Grid Federation
EUDAT	European Data

FAIR	Findable, Accessible, Interoperable, Re-usable[1]
FEM	Finite Element Model
FDSN	Federation of Digital Seismological Networks
GB	Gigabyte
GCMT	Global Centroid Movement Tensor Catalogue
GEF	Generic Execution Framework
GIS	Geographic Information System
GHG	Greenhouse Gas
GUI	Graphical User Interface
HCI	Human-computer interaction
I^3	Intent, Interpretation & Incentive
INGV	Instituto Nationale di Geofisica e Vulcanologia
IoT	Internet of Things
IS-ENES	Infrastructure for the European Network of Earth System Modelling
LOD	Linked Open Data
ML	Machine Learning
MVV	Monitoring and Validation Visualiser (used with provenance data)
NCAR	National Centre for Atmospheric Research
ncl	NCAR Command Language
nco	NetCDF Operators
NetCDF	Network Common Data Form
NLI	No-Longer Interested (used for shut down propagation in data streaming)
O-O	Object-Oriented
OGC	Open Geospatial Consortium
OPeNDAP	Open-source Project for a Network Data Access Protocol
ORFEUS	Observatories & Research Facilities for European Seismology
P2P	Peer-to-Peer
P4	Protected Persistent Pervasive Provenance an authoritative and controlled record of all that has happened
PE	Processing Element a data-streaming process in dispel4py
PID	Persistent IDentifier – a means of identifying something for as long as required
PPR	Price-Performance Ratio
RA	Rapid Assessment of earthquake sources and their expected impact
RCP	Representative Concentration Pathways
RD	Research Developer
RDF	Resource Description Framework
RI	Research Infrastructure
RM	Reference Model
RoI	Return on Investment
SEM	Spectral Element Model

[1] Force 11 FAIR https://www.force11.org/group/fairgroup/fairprinciples

SHACL	Shapes Constraint Language
SKA	Square Kilometre Array - a global radio astronomy research infrastructure
SoS	Start of Stream (In data-streaming workflows)
SS	Seismic Source characterisation
TB	Terabyte
TDMT	Time Domain Moment Tensor Catalogue
TDS	THREDDS Data Server
UI	User Interface
UDF	User-Defined Function (that can be shipped into a database or close to data)
UoD	Universe of Discourse
URI	Universal Resource Identifier
URL	Universal Resource Locator
VM	Virtual Machine via any virtualisation mechanism, e.g., containers
VO	Virtual Organisation
VRE	Virtual Research Environment in this context a synonym for science gateway
WaaS	Workflows-as-a-Service a means of getting actions done
WMS	Web Map Service
WP	Work Package
WPL	Work Package Leader
WPS	Web Processing Service

1 Introduction

This deliverable describes the architecture for the DARE platform. The architecture is intended to help steer the DARE work and be part of the DARE legacy.

1.1 Purpose and scope of the architecture

This document presents the DARE architecture. The architecture addresses the three DARE challenges of extending the scale of computation and data, handling the growth of complexity while delivering research agility. The architecture presents researchers and research developers with a new mode of working. They still feel in control and take responsibility for correctness, but they work with names that abstract over data, methods and services. The architecture maps these to enable their work, interpreting it consistently and efficiently as the target platforms evolve. This extends the value of their investment in methods, increases their productivity and accelerates innovation. This is necessary for sustainable data-driven science, to broaden accessibility, to cope with scale and to limit energy consumption. To achieve this the architecture introduces a comprehensive central catalogue, with a sufficient range of core concepts each well described for both human and machine use. That catalogue will be tailored for communities and extended by innovators. Interaction, processing, working practices and data handling will all consult and often update the catalogue.

The architecture will shape the work of the DARE project, project-dare.eu. It embraces a long-term vision, exploits technical opportunities and addresses the immediate requirements. It is intended to provide a framework and design pattern for many future applications that need to extend their capabilities. It will help them form multi-disciplinary information-sharing productive consortia.

The DARE architecture is being tested by two user communities: computational seismologists in EPOS [Rietbrock *et al.* 2018] and climate-impact modelling in IS-ENES [Pagé & Spinuso 2018]. The architecture considers development and operational issues, prior investments, established working practices and external guidance. It must reconcile conflicting requirements, acknowledge constraints and resource limitations and still be ambitious. This requires that the architects facilitate a consultation with representatives of *all* stakeholders. DARE revises the architecture every four months. This has delivered two internal deliverables: ID2.1-M4 and ID2.1-M8 [Atkinson *et al.* 2018a, b]. The Architectural Task Force (ArchTF) – see below – organises the consultations.

1.2 Methodology and Structure of the Deliverable

The architects will take account of contemporary solution strategies and develop guiding principles introduced in Section 2. Section 3 identifies the anticipated benefits of adopting the DARE architecture. Section 4 introduces the three major features of the DARE architecture and their functions. Section 5 presents the strategy to enable communities, organisations and individuals to adopt DARE's approach incrementally to minimise risk and disruption. Section 6 presents the current status of the DARE platform and its extensions for each user community. Section 7 summarises the architectural implications of the requirements of the two user communities based on analysis by Task 3.1 [Spinuso & Filgueira 2018]. The architecture is prototyped as a framework of core functions, concepts and information sharing presented in Section 8. Each subsystem is analysed to identify the functions, interrelationships and technological constraints it must meet. Section 9 has a parallel structure to Section 8 and covers technical options. Section 10 identifies the impact on and response from each work package. The deliverable concludes with a summary and longer-term vision in Section 11.

In summary, the architecture is presented in three phases:

1. A high-level view in Section 4.
2. An engineering view in Section 8.
3. A technological view in Section 9.

1.3 Approach and relationship with other Work Packages and Deliverables

The architectural analyses and processes need the pooled insights and judgements of a group that has the relevant breadth and expertise. This includes representatives of each user community (WP6 and WP7), representatives of the technology developers (WP3 and WP4) and those deploying and operating the DARE platform (WP5). The Technical manager also represents WP1 and the WP8 outreach and training is also engaged. To pool their expertise an Architectural Task Force (ArchTF) has been formed that holds monthly tele-conferences. The preparation for, and records of these meetings can be found in folder WP2 Meetings, in the DARE googleDrive[2].

Table 1: Individuals and their roles in the ArchTF at M8 (August 2018)

Name	Full Name	DARE role	Architecture role
Malcolm	Malcolm Atkinson	WP2 leader	Architectural lead & deliverable editor
André	André Gemünd	WP5 leader	Expertise on operations, deployment and platforms
Iraklis	Iraklis Klampanos	Technical manager	Coordination with the other activities in DARE and expertise on big data technologies
AntonisK	Antonis Koukourikos	WP4 leader	Expertise on big data systems, particularly BDI and Semagrow
AntonisL	Antonis Lempesis	Athena rep.	Exareme and big data-streaming expertise
Byron	Byron Georgantopoulos	GRNET rep.	Cloud deployment and platform operation
Chris	Christopher Wood	Deputy T2.1 leader	Deputy to architectural lead and deliverable editor
Christian	Christian Pagé	WP7 leader	Climate modelling research challenges
Andreas	Andreas Rietbrock	WP6 leader	Computational seismic research challenges
Alessandro	Alessandro Spinuso	WP3 leader	Expertise on provenance, tools & VREs

At its meeting on 2/11/2018 the ArchTF agreed the contributors to and structure of the D2.1. The work of the ArchTF will be organised using the following procedures:

1. A monthly virtual meeting[3] for consultation to resolve issues and agree priorities.
2. An extraordinary meeting if any urgent issue requires it.
3. Observers and additional contributors are welcome and designated delegates normally find substitutes if they are unavailable.
4. A technical meeting of a subgroup when necessary to address specific topics.
5. Progressive development of the next edition of the "*DARE Architecture & Technical Positioning*" deliverable, with a presentation to DARE's technical board at least a month before it is due, for approval, comment and suggestions.
6. A draft for internal review one month before a full deliverable is due. Internal deliverables will be presented to the first DARE (tele-)plenary meeting following their completion for approval.
7. Explanations of the architectural decisions at F2F meetings and to external interested parties. These will include talks and papers.
8. Investigation of any issues the DARE technical board identifies and allocates to architecture with best efforts to respond quickly.

[2] These records are available on request.

[3] A F2F instance of this meeting will be scheduled during each DARE project F2F.

2 DARE's architecture: motivation, structure and concepts

The architecture for DARE should address three questions:

1. What aspects should we focus on as we develop systems that meet DARE's goals?
2. What will those parts do to meet DARE's goals?
3. How should those parts be assembled, how should they interact with their digital environment and user communities, and how should they interrelate?

Developing good answers to those questions that are widely applicable is a key element of DARE's research. In this section we introduce three top-level parts as an initial stab at question 1 and briefly consider question 2 for them. Sections 4, 8 and 9 present initial answers to question 3. The architecture is shaped to address the needs of demanding data-driven application communities typified by the needs of DARE's two engaged communities: computational seismology (WP6), see deliverable D6.1 [Rietbrock *et al.* 2018], and climate impact (WP7), see deliverable D7.1 [Pagé & Spinuso 2018]. Their needs have been analysed and consolidated by Task 3.1 in ID3.1 [Spinuso and Filgueira 2018]. Contemporary deliverables D3.5 [Klampanos *et al.* 2018] and D3.7 [Spinuso & Klampanos 2018] provide preliminary prototypes on which future steps towards the DARE will build. DARE will continuously test and refine its architecture by intensively engaging in co-design and co-development with its two user communities.

The background to DARE's architecture is a synthesis of experience sharing data, methods and resources in research federations and through science gateways over several decades [Atkinson *et al.* 2013]. It has recently been sharpened by experience in VERCE [Atkinson *et al.* 2015] and in the two EU-supported ENVRI projects pioneering approaches to supporting all stages of the data lifecycle for environmental research infrastructures. The ENVRI work led to the ENVRI reference model[4] shaped by the requirements of 23 environmental ESFRI-endorsed research infrastructures[5] including the two that are partners in DARE: EPOS and IS-ENES.

2.1 Purpose of the DARE architecture

The DARE architecture presents a design and construction strategy for systems that support data-driven application domains with distributed communities. The resultant work environments must help all aspects of such a community, the domain experts, the systems and software engineers and the managers as they collaborate to achieve their community's goals. Therefore, the architecture must meet the goals of the DARE project. These are:

1. To deliver *research agility* to data-driven application domains.
2. To significantly *extend the capabilities* available to such domains in respect of:
    a. the *scale of data* they can handle,
    b. The *amount of computation* they can apply and
    c. The *complexity* they can manage.
3. To have a *sustainable* and widely applicable *legacy*.

Delivery in the DARE context requires an intimate combination of user-engagement and technological strategies so that professionals working in the application domains adopt and benefit from DARE's innovations. This requires that the DARE architecture respects their concerns, empowers their judgement and works with the technologies they use on the platforms they have access to. Consequently, it is essential to support incremental adoption and co-evolution without unnecessarily disrupting established commitments, methods and working practices. *Benefits must be self-evident to motivate adoption*. While non-disruptive introduction mitigates professional concerns about risks it may not always be achievable.

Consequently, shaping the architecture so that it meets these demanding goals within the socio-economic and engineering constraints is a research challenge. Through intensive co-design and co-development with DARE's two application communities, the architectural strategies will be

[4] The ENVRI reference model: https://wiki.envri.eu/display/EC/ENVRI+Reference+Model

[5] European Strategic Forum on Research Infrastructures (ESFRI) https://ec.europa.eu/research/infrastructures/index.cfm

developed and tested. This experience will help shape the design patterns for the architecture and steer the development of a reusable framework that supports those patterns.

2.2 Constraints on the DARE architecture

To achieve its target the architecture considers the following:

1. Deep HCI[6] to ensure that all professional roles and activities are well supported and collaboration between those roles is encouraged; this includes:
    a. Ensuring professionals feel in control, well-informed and able to apply their own judgement.
    b. Maintaining a balance between sustaining existing working practices and empowering innovation.
    c. Lowering barriers to incremental adoption – taking full account of the range of adopters[7] – see Section 5.
    d. Establishing mechanisms to improve and validate the quality of evidence derived from data.
2. Independence from technological detail so that the existing diversity of technologies in use can be effectively exploited via the framework and newly available technologies can be harnessed rapidly.
3. Adaptations for a diversity of scientific, organisational and governmental drivers that span from multi-decade campaigns to immediate response to events.

These issues are considered in the context of the large consortia that are necessary to address today's challenges. Such consortia form loosely coupled federations to build support for a cluster of related data-driven activities. These federations gather required skills and resources. They need to grow and adapt to meet emerging needs. As a result, they become large, multinational and multi-disciplined. They involve many organisations, groups and individuals who have their own independent goals and responsibilities. Their communities and stakeholders develop diversity, from those whose focus is funding to those keeping their technology running – with experience ranging from novices to leading experts. Many need to encourage the work of citizen scientists. DARE directly addresses the complexity inherent in such federations as these become prevalent in application domains.

Many professional roles need to be respected. In each application domain there are researchers who are expert in aspects of that domain, its theory, models, methods and standards. Enabling them to employ their knowledge and experience is a priority. Many will focus on routine work because it is needed for longitudinal studies or delivered services. A few will explore new possibilities. One or two will be research leaders, identifying new targets, gathering the required resources, steering the effort and guarding their campaign's reputation. These application roles will be supported by teams, including those building the required infrastructure and software; the systems engineers, operations teams, software engineers, data architects, data scientists, simulation-system engineers, numerical analysts, statisticians and mathematicians. These in turn depend on a similar diversity of roles in supplier organisations. Funders and governments may set goals, specify regulations, assess achievements and require compliance. These multiple viewpoints are essential, but they are a source of complexity as well as resources.

The architecture must enable three **I**s (I^3) for this combination of roles in order to be adopted [Schubert & Jeffery 2015]:

1. **Intent**: It should be easy for each professional to express their intent clearly.
2. **Interpretation**: That intent should be interpreted consistently as it crosses between disciplines, as it passes between sites, as it passes from people to systems, from system to

[6] The experience afforded as all the professionals interact with services built using the architecture is critical to success. That may be termed Human-Computer Interaction (HCI) and all aspects of it are important. We focus on what we call *deep* HCI here – that is the ways in which humans and systems share responsibility and who is in control.

[7] They span the software-adoption curve https://en.wikipedia.org/wiki/Technology_adoption_life_cycle

system and from software to people. That interpretation should be preserved through time unless it is explicitly changed.

3. **Incentive**: Individuals, groups, organisations and federations must experience immediate benefits sufficient to warrant the perceived risk of disruption from using new technology; promises of *eventual* benefits when changes are completed will not convince users.

2.2.1 Catalogues delivering I[3]

For consistent understanding of users' intent, a catalogue must capture all the vocabulary used and describe its interpretation. For consistent and sustained interpretation, a catalogue must describe the elements of supporting systems to which that interpretation is mapped. We introduce the nature and role of a logically central comprehensive catalogue. The content and performance of this catalogue is critical to the success of the DARE platform for the following reasons:

1. It provides the Universe of Discourse for humans as they communicate with collaborators via DARE, as they organise and control their work and as they communicate with the software systems that perform their work. As such, it must facilitate their understanding and consistent usage. It must enable them to accurately express their intent in terms that make sense to them.
2. It mediates the interpretation of their intentions by software. The software uses the information it contains to consistently interpret users' intentions as it maps their actions to multiple evolving computational contexts. The metadata about data, actions and targets in this knowledge base must contain sufficient information and be kept up to date.
3. It must support typical scientific practices, such as selecting or building up collections of data items and applying actions to those collections, as efficiently and as promptly as required.
4. It must prepare for a data-rich future where the majority of research procedures have to be supported by automated methods.
5. It must facilitate incremental change by incorporating mechanisms for unpredictable extension and revision.

2.2.2 Universe of Discourse covering everything needed

We use the phrase "*Universe of Discourse*" (UoD) to capture all of the things people using or software in the DARE platform are communicating about, to quote Collins Dictionary,

"*the complete range of objects, events, attributes, relations, ideas, etc, that are expressed, assumed, or implied in a discussion*"

For the system, it is individuals, organisations, projects, their credentials, authorisation and entitlements, software elements, subsystems, services, data sets, workflows, provenance records, and so on.

For the experts in a particular application community, it is all the concepts, terms, data sources, standards, conventions, working practices, methods, archival stores, instruments, observations, data streams, time series, intermediate results, etc. Each application community will have a different set of concepts, with potential overlaps with other communities. DARE encourages that overlap to facilitate sharing of R&D and sustainability effort, and to aid inter-community collaboration. In particular, it recognises the overlap of geospatial and environmental concepts and the commonalities in scientific models and methods.

Advances in science and changes in UoDs are deeply coupled. To make its researchers and developers *agile,* DARE needs to capture and exploit many external changes to reveal their power to its communities and their supporters as quickly as possible. To empower its communities and developers to contribute their innovations, DARE must support boundary developments, where groups explore new ideas with their associated UoD changes. When an innovation proves its value, paths for its promotion and adoption must be facilitated. However, these must also avoid

unnecessary disruption to existing vocabularies and established working practices to preserve intellectual investment and retain users' confidence.

The catalogue is the medium for accomplishing this in DARE. By organising its knowledge structure via the explicit recognition of concepts and focusing on their different descriptions and usage patterns, DARE makes handling this complexity feasible. The catalogue becomes an information resource that enables users, systems engineers, developers and software to handle this complexity consistently and incrementally.

2.3 DARE's architectural focus

DARE is working in a multi-faceted, dynamic and complex socio-technical environment. To contribute DARE needs to focus on particular challenges within this digital environment and depend on contemporary R&D to contribute in other areas; drawing on those areas when it needs to. We set out the topics DARE's architecture addresses and the contextual issues that it does not address but may draw on.

DARE's focus – the topics where it will invest effort and contribute advances.

1. Get users working with names and identities rather than location-specific file paths and URIs so that their intent can be preserved while interpretation adapts to technical advances and economic necessities[8].
2. Get users actively using catalogue functionalities (see Sections 4.3 and 8.3)**,** including domain-specific metadata to gain the incentives reported by Jim Myers *et al.* [2015] and further demonstrated by Spinuso's work [2018]. This shapes WP3 delivering improved productivity, processes and metadata through the power of its tools and methods[9].
3. Providing a framework for sharing concepts, to pool information about definitions, patterns of use and interpretations of these concepts so that they are more consistently used and better understood. That consistency must persist across technology changes and as work passes between nations, organisations and disciplines. It must survive the introduction of contemporary advances in neighbouring areas. However, concepts need to evolve and adapt to match the changing needs of science and groups. In most professions their agreed form is inculcated through training. This determines a well-defined initial culture. DARE provides a framework for capturing and recording that information and revising it when necessary. EPOS (WP6) developed this approach [Trani *et al*. 2018][10].
4. Facilitating the extension of concept use across discipline boundaries when that is needed, e.g., when an informatician works with an application domain, or when one discipline, e.g., geodesics, shares information and methods with another discipline, e.g., seismology. This is a major motivator. DARE recognises that each professional community is self-sufficient, autonomous and co-evolving. But they need help with the rarer interactions across disciplines. This is being pioneered in EPOS (WP6) [*ibid.*][11].

[8] This is a major incentive for funders and managers – saving costs and improving productivity. As data becomes too large to download, it is an incentive for leaders as it enables their research to continue. Citizen scientists can engage without needing their own resources.

[9] Professionals are incentivised by the automation removing chores, summarising and visualising progress to help them manage their work and understandable paths when they investigate an issue or re-evaluate evidence.

[10] The incentive is significant improvement in intellectual RoI. Creation, refinement and encoding of methods survives as the digital and scientific contexts change. Learning how to use methods pays off for longer and in new contexts.

[11] Such boundary crossing is essential to address societal challenges, to capitalise on our growing wealth in data, or to conduct research in the face of Earth-systems' complexity.

5. Advanced frameworks for defining and describing actions and methods (WP4)[12]. Their use and form may be explicitly constrained; e.g.,
    a. restricting their applicability to appropriate inputs and matching authorisation,
    b. limiting resource usage rates,
    c. specifying consistency in terms of the conceptual framework, e.g., when handling a time series from a seismometer (WP6) one source of Python methods is used and data from particular archive services is catalogued as delivering instances of this concept, whereas,
    d. when handling data from a climate simulation (WP7) a different source of Python libraries is specified as relevant.
6. Exploit the power of provenance to enable:
    a. Users to walk through steps in a method to better understand what it does and how to use it,
    b. Developers and specialists to improve methods by studying how they are used,
    c. to assess the quality of evidence by reviewing past procedures, and
    d. to expose issues by analysing the historical records looking for anomalies[13].
7. Facilitate the co-existence of different work contexts[14], e.g.,
    a. production contexts where stability, performance, costs and reliability are priorities,
    b. diagnosis contexts, where the interrogation and analysis of provenance has a high premium,
    c. development contexts, where experiments, testing and exploration are well supported and have minimum constraints,
    d. training contexts, where learners are presented with opportunities to learn without risks to themselves or to research processes.
8. Support a diversity of organisational structures sustaining interdisciplinary and multi-national evolving agreements.
9. Embrace co-working between traditional and DARE-mediated methods so that incremental adoption is encouraged.

Contemporary work will be conducted by others – DARE will invest effort to import additional[15] technology and methods from these contexts, when necessary. These imports may need adaptation and extension. The architecture should always support such imports and extensions.

1. Provide the underpinning services and resources other than:
    a. the extended catalogue functionality,
    b. the enactment of actions until they can be delegated and
    c. the pervasive capture and use of provenance records.
    DARE depends on and maps to all other resources, e.g., for storage, data access and transport, AAAI, computation, etc.[16]
2. Support R&D for individual communities and disciplines. DARE assumes they are self-sufficient, with established data and metadata standards supported by their services and tools. It encourages them to bring these into DARE's framework because of the protection from technology change provided by DARE's mapping services, and for assistance with domain boundary crossings. Further incentives come from automated validation of correct

[12] With sufficient abstraction and support, experts in aspects of a domain may directly create, explore and refine methods. With a suitable framework for composition, experts in different fields can combine their contributions. As these move into production, data-scientists, systems experts and resource providers contribute optimisations.

[13] The immediate incentive is the power of provenance-driven processes and tools. Longer term, the value of provenance as evidence, to protect organisations and individuals, and to facilitate review will be significant.

[14] These are needed to deliver good working environments for a variety of activities; key to usability and acceptance.

[15] In many cases DARE already has key technologies or methods developed by earlier R&D (see Section 6).

[16] Prototypes of the DARE platform produced by WP4 and run by WP5 include such resources and the lowest-level elements. DARE's architecture must remain independent from particular choices. Much contemporary work in big-data processing systems will generate important innovations but we cannot predict their form. Consequently, the architecture must be open to describing them and mapping to them.

usage where provenance is adopted, as well as improved productivity from catalogue and provenance-driven tools and improved confidence in the validation of evidence. The disciplines themselves have to push this agenda. DARE has a few resources to help them *start* doing this, but insignificant resources compared with the scale of the task.

3. Provide innovative methods, models, simulations, software or representations – communities and research leaders have to find resources for their innovations. However, the DARE architecture should facilitate that innovation by providing a productive development context and support methods (promotion workflows) to safely introduce such innovations into production contexts after validation. They should then interoperate with the relevant variety of existing subsystems and methods. In DARE, WP2, WP3 & WP4 may do a little innovation to demonstrate how the DARE architecture helps. DARE should also demonstrate how to import the work of external research communities.
4. Tailor DARE's generic platform to suit a particular community. DARE will demonstrate how this is done for two disciplines, thereby honing the tailoring methods and pioneering some supporting tools. Within DARE, this will not be completed, even for the two chosen communities. But enough demonstrable benefits and momentum will be achieved to motivate continuation in those two communities.
5. Develop ways of describing most of the relevant concepts. Others are doing this for almost every concept, and we should import their work. The architecture should relate relevant descriptions to the concepts and entities it supports and uses. However, we will find that in some cases further information is needed, e.g., to extend consistent interpretations, improve usability or support new optimisation opportunities.
6. Develop ways of doing data translation and harmonisation. DARE should import what is relevant, e.g., from VRE4EIC[17], which we may access via Semagrow (see Section 6).
7. Develop scientific workflow languages and strategies for their reliability and optimisation. There is a great deal of contemporary work here which we should import and build on. DARE will build the initial framework and perform the initial tailoring with dispel4py [Filgueira *et al.* 2016a] and Python.
8. Develop DB systems for any form of scientific data or description technology. These we should import, but not lock into a particular form for the architecture, though we may for the DARE platform.
9. Require total transfer from current working practices and computational/data environments to DARE's platform and methods. An incremental path is essential (see Section 5).

3 Benefits from the DARE architecture

The architecture presents a consistent, integrated and understandable view of multiple data resources and powerful operations using those resources while exploiting the latest technological advances. It recognises that users in various roles must feel they are in control of the aspects of a community's work they take responsibility for. DARE needs to support existing working practices as well as encourage innovation. The DARE platform API will make the integrated functionality available to tools tailored to the various roles and varieties of expertise [Klampanos *et al.* 2018].

We present, with examples, the benefits of this approach for various research, development, engineering, operations and leadership activities. The outline of the architecture, particularly with respect to its novelty, is presented in Section 4. The DARE architecture is a dramatic step from today's research environments. This is necessary to achieve the features presented above while capitalising on new technologies, growing data resources and elastic resource allocation. Consequently, incremental adoption is essential. The strategy for doing this – working alongside communities' current systems, supporting their established working practices and incrementally migrating as the incentives become compelling is presented in Section 5.

[17] VRE4EIC https://www.vre4eic.eu/

3.1 The benefits for application-domain researchers

Typical domain researchers have to do the same tasks repeatedly. They use their expertise to apply methods to new observations, to produce data products related to long-term phenomena or to respond to requests. Their expertise includes gathering the relevant data, making judgements about its quality, preparing it for their task, applying established methods, steering those methods, reviewing results, shaping and embedding the presentation of results, assessing and specifying the quality of the resulting evidence. A suitably tailored mature instance of the DARE platform would support all of these steps for the domain professionals. It would have their community standard data sources and established methods already set up, and would provide a consistent understanding of the terminology, authoritative standards, representations, methods, tools and presentations adopted by their community. This is mediated via the catalogue, which lets them use established terms and identifiers to express their intentions as actions being applied to and producing named entities.

We illustrate this by imagining a seismologist, Ann, using an instance of the DARE platform that has been tailored for computational seismologists via a user-interface tool such as a Jupyter notebook to perform a rapid assessment (RA) of strong ground motion – see Section 7.1. She sets the geographic area she is working on, as an input parameter for her GIS system and data visualisation by supplying latitude and longitude bounds.

new GeoArea ga = GeoArea(36, 41, -20, -16)

Next, she chooses a wave-propagation model for the area she is about to work with from the catalogued wave-speed *SEModels*.

new SEModel wsm = SEModels **where** name = “southernItaly”

She reviews the suitability of *wsm* by mapping its visualisation in the *ga* and interactively probing some of its properties.

wsm.visualise(ga)

She judges that it is not entirely suitable and decides to load an alternative model she has in her file system.

wsm = **ingest** SEModel **from** /RA/INGVmodels/em123

This she knows from experience to be suitable, so she does not re-run her evaluation, instead she chooses a seismic source from the reference sources[18], GCMT and TDMT, which are mapped to deliver instances of the concept *SeismicSources* into the DARE context where she is performing this role via an authorised *Session*. She specifies her choice by selecting by time.

new SeismicSource ss = SeismicSources **where** time = 2017:07:13:18:23

Again, she checks her choice by visualising it. Each concept has a default visualisation adjusted to suit her community’s normal requirements. Extra parameters would allow her to override these. The GIS system used by visualisers allows viewing controls and can yield a result corresponding to a selection as we see when she modifies the content of *targets* interactively below.

ss.visualise(ga)

She decides, after using drill-down to see more details, that it is suitable. She then sets up a list of *Seismometers* she wants to use in the comparisons between simulation outputs and observed ground motion.

new Seismometer[] targets = Seismometers **where** geoLocation **in** ga

[18] Today researchers deal directly with their reference sources, learning how to interact with each one, establishing access and usage rights if necessary, by direct and immediate interactions. They marshal and harmonise data via explicit downloads and explicitly applications of translation processes. DARE would set up these arrangements and mappings, recording them in the catalogue when it is asked to **see** a source as a means of obtaining instances of a concept. It postpones mapping until it is marshalling data to a location where it will be processed – done today in *some* workflows.

targets = targets.visualise(ga)

She starts a run of the workflow already stored in her context in a ready-to-run form to simulate the propagation of seismic waves from *ss* and *wsm*.

new GroundMotionTimeSeries gmts = forwardWaveSimulation(wsm, ss, 600, 0.1, targets)

She specifies the *SEModel*, the *SeismicSource*, the duration in simulated seconds and the time step, as well as the set of *Seismometers*, *targets*, where she will compare the simulation products in the collection of results *gmts* with the seismic observations at those target locations.

ss.visualise(ga) **with** gmts.visualise(ga)

She visualises the wave propagation surface motions emerging from the simulation and makes a judgement about whether the results are usable.

The DARE platform was able to do all of this while only bringing to the platform itself what was needed to supply Ann with her requested visualisations – small fraction of the size of the data being visualised in many cases. When actions requiring significant computation and substantial data are initiated, e.g., the wave-propagation simulation using SPECFEM3D (see Section 6.2), the DARE platform is able to choose where to conduct the run and to handle the data with minimum cost. However, it still allows Ann to supervise and apply her judgement. Triggers supported by the provenance system enable workflow developers to arrange that progress information can be streamed to Ann to enable her to judge whether continuing the simulation is worthwhile.

She obtains and prepares the relevant seismic traces for her set of seismometers, *targets*, for a comparison with the simulation results; where the collection SeismicTraces includes the mapping of queries to the relevant EIDA seismic-archive registries.

new SeismicTrace[] st = SeismicTraces **where** st.instrument **in** targets **and** st.start < ss.time **and** st.end ≥ ss.time+600

new PreppedTrace[] pt = prepTrace(st)

She now does a precise comparison between the simulation results.

new GMdiffs[] gmd = compareGMs(gmts, pt)

gmd.visualise(ga)

These give her confidence in the results and she computes a ground motion shake map based on the integration of observed and simulated values of the ground motion parameters as this has more coverage of the area of concern. In steps involving collections DARE has inferred that iteration is required.

new ShakeMap sm = computeShakeMap(gmts, st)

sm.visualise(ga)

This now becomes available in the set of catalogued *ShakeMap* entries that could be found and used by others by searching the automatically maintained collection of concept instances:

new ShakeMap sm = ShakeMaps **where** source = ss **and** model = wsm

The fetching of seismic traces, their preparation and their comparison with simulation results are all combined, parallelised and distributed on chosen Cloud resources by the DARE platform, for economy or speed of response. Ann only sees the visualisation start more quickly.

Some days later Bob tells her that two seismometers, *seis1* and *seis2*, were giving incorrect responses during the period 1 March to 15 October 2017. Ann thinks she may have used one of these in the last month. Fortunately, DARE's pervasive provenance, together with her team's embedding of domain metadata in the provenance stream lets her check quickly.

new Run[] r1 = Runs **where** seis1 **in** targets **and** run.time ≥ 20181001

r1.list

new Run[] r2 = Runs **where** seis2 **in** targets **and** run.time ≥ 20181001

r2.list

She sees that *r1* is fortuitously empty but *r2* has one *Run* in it calculating the above example. She adjusts targets to remove seis2. She then re-runs the steps affected, which she could obtain from the provenance trace, but she understands this issue and works directly.

targets -= seis2

st = SeismicTraces **where** st.instrument **in** targets **and** st.start < ss.time **and** st.end ≥ ss.time+600

new PreppedTrace[] pt = prepTrace(st)

gmd = compareGMs (gmts, pt)

gmd.visualise(ga)

sm = computeShakeMap(gmts, st)

DARE should be able to avoid reperforming preparations and comparisons for sources that have not changed by caching and checking with the provenance and catalogue that things they depend on have not changed. Ann is not expected to keep track of everything others do. For example, DARE notes when the shared *ShakeMap* is revised to a new version. It would normally use the new version automatically but can be configured to consult before inserting revised versions into an action. Users can register interest in items, so that they are notified when new versions are inserted into the catalogue by others.

The working professionals get on with their work, still supervise and control things, but they do this through named concepts like *Seismometer*, *SeismicSource*, **ingest**, **new**, *ga*, *ss*, s*eis1*, *seis2*. Some of the terms are communally agreed and some are specific to them or their group. The integration of this set of names and their consistent interpretation as technology varies is a significant advance. It enables application groups to build and control sets of names and collections of these with an agreement about what each name that is shared means precisely and with guidance and checks on how it may be used. This enables them to build and control a virtual consistent world without having to warehouse data to make it conform. This saves many chores and errors as DARE takes care of the housekeeping. It is made possible by the underlying catalogue containing enough information about concepts and the meaning, representation, uses and properties of every instance of a concept, and about actions and how they are to be used and interpreted to preserve their semantics.

DARE keeps careful records, *provenance*, of what has happened. Interrogating these enables users to review and understand what has been done and what has yet to be done. The DARE platform uses provenance records to automate re-running with modifications and to avoid redundant work. It accelerates recovery from errors and facilitates the exploitation of new information. Figure 1 shows the nature of the harmonised conceptual space used by Ann.

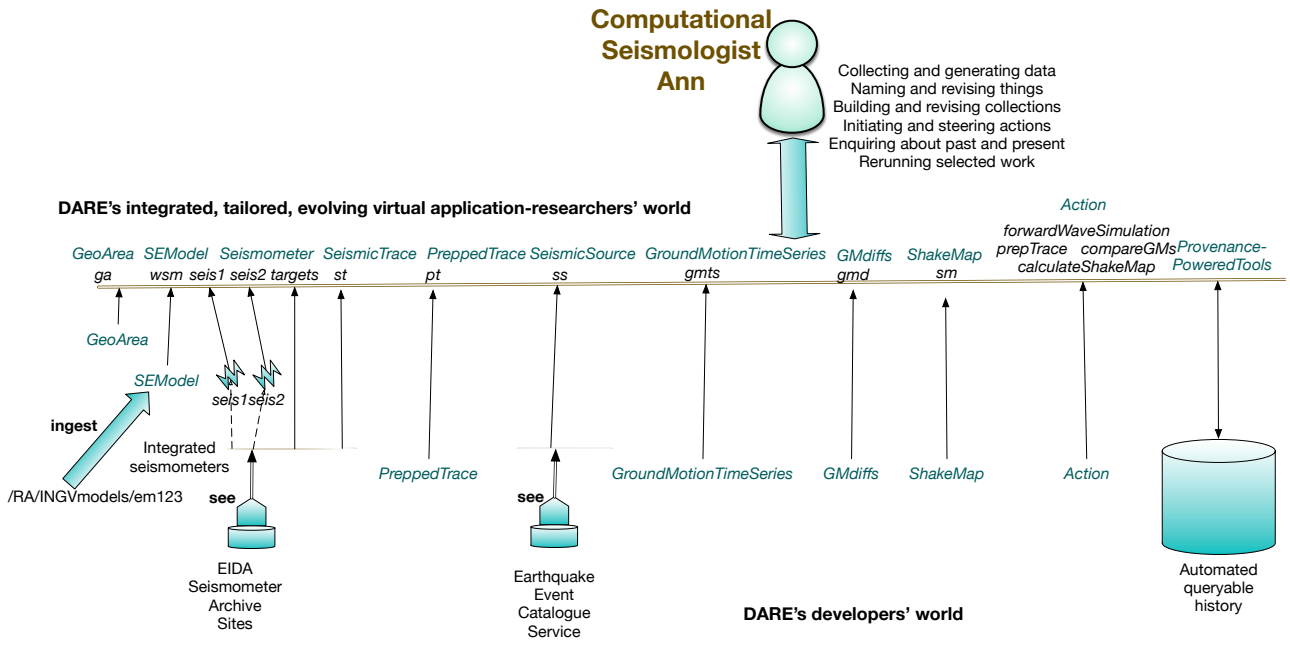

Figure 1: The understood name space used by a computational seismologist, Ann, appears above double line. DARE protects her from the underlying complexity where research developers work.

If the professional researchers had to build their DARE virtual world from scratch, they would never attempt it. It would be too daunting and the payoff too distant. DARE overcomes this in two stages:

1. Professionals from various disciplines act as pioneers and use DARE's virtual world to *tailor* DARE to provide a harmonised information, methods and facilities for their community.
2. To make that feasible, the developers of DARE pre-populate DARE's virtual world with an evolving *core* of concepts and a population of useful entities required by multiple federations. Communities will choose when they install releases of that core into their developer and production contexts. DARE's benefits to an application domain's professionals are:
    1. Their world persists and evolves under their control so that their investment in learning about it and in developing methods and concepts for it retain their value almost indefinitely.
    2. They work using terms, annotations, methods and programs they understand. These do not directly specify the choice of technology. Established practices rarely need revision as the capabilities of new technologies are exploited. The methods execute faster and handle data and computations that they could not continue to manage on their own resources.
    3. The ability to see the system as fully integrated and to enquire about the metadata and managed history allows domain researchers to organise long-running campaigns without onerous housekeeping – it saves them tracking and analysing progress.
    4. It lowers the intellectual energy needed when novices get started.
    5. It allows rapid investigation of issues via the provenance system.
    6. It provides convenient ways of redoing selected work with optional modifications.
    7. It documents data dependencies to help them assess the quality of data produced.
    8. It acts as a laboratory notebook, making it easier to document simulations and computations and it assists users in sharing the results and computational algorithms.

These benefits will be sufficient incentive for adoption provided that process is incremental.

3.2 The benefits for research developers

While most activity is routine work with minor incremental refinements, progress in handling new data, in exploring new theories and in exploiting the new computational power depends on

innovation. That innovation is triggered by application-domain scientists (**ADS)** developing new insights. They, sometimes in with a research developer (**RD**), will explore this idea developing and testing prototypes until they believe their innovation is ready for production. Achieving production ready implementations, often requires interdisciplinary teamwork to address the scale challenges, to achieve sufficient accuracy, reliability and usability on multiple platforms. In this context, DARE focuses on harnessing the skills of research developers. Investment from management and other engineers may also be needed – see below. These RDs also need to help sustain and support their products. Therefore, DARE envisages RDs operating in two modes:

1. *Collaboratively pioneering* new software destined to run for one or more application communities.
2. *Investigating and repairing or extending* software when a user community has an issue.

DARE assists in both of these cases by letting RDs access all aspects of the *production* research context while helping them innovate in their *development* context. These work contexts are carefully separated to facilitate each role's behaviour while avoiding accidental disruption of production work. DARE provides a controllable bridge between these contexts to accelerate and improve ADS and RD collaboration. An example of work in the development context was presented in ID2.1-M8 [Atkinson *et al.* 2018b].

The benefits of the DARE architecture here include:

1. Protection of production context so that ADSs enjoy a stable and reliable work environment.
2. Freedom for developers to proceed by any means and to use any development tools and strategies they wish.
3. Controlled support for RDs seeing relevant parts of production spaces, enabling them to investigate and understand what is required rapidly.
4. Sharing of vocabularies and instances across the inter-disciplinary boundaries, thereby, improving the rate and depth of mutual understanding.
5. Controlled visibility of changes across the boundary, so that each side is able to produce things to help the other.
6. Sophisticated workflows (implementing **promote**) supporting the promotion of innovations and repairs into relevant contexts, alerting those who are concerned (inferred because of use discovered from provenance records or those who explicitly expressed interest).
7. Community agreed mechanisms for ensuring quality in the elements that are moved into production via the **promote** workflows.

These combine to improve research developer and researcher productivity while retaining agility. They also help support teams deliver good service to extensive distributed collaborations – illustrated in detail in ID2.1-M8 [*ibid.*].

3.3 The benefits for research leaders, managers and funders

For the e-Infrastructure and all the intermediate software and data to be sustained, maintained and supported over the duration of research campaigns it is essential that costs are considered[19]. How costs are measured is an open question, they can be in financial terms, in energy terms or in human terms, recognising the diverse skills needed. The DARE architecture takes this into account in several ways:

1. by directly reducing costs,
2. by analysis of a community's history to identify opportunities for improvement, and
3. by establishing a conceptual and practical framework that facilitates interdisciplinary collaboration.

Socio-economic judgements are also important, to keep a community collaborating effectively, to find opportunities to amortise sustainability costs, and to balance opportunities for innovation with

[19] Sustaining the software and e-Infrastructure on which researchers depend is just as important as sustaining observational instruments or experimental systems – Software Sustainability Institute https://www.software.ac.uk/. If it stops working their research suffers. If it is not upgraded they lose their competitive edge.

stability for productivity. Examples below show how the DARE architecture provides a platform that has the potential to address these issues.

DARE provides each user with a persistent coherent view of their workspace so that they can repeatedly interact over long periods without disruption. They interact with a virtualisation of their world without marshalling their data –they must still feel they *control* and *own* their work. The DARE platform will optimally map work and data onto available resources. These mappings will change as the priorities and trade-offs change. The users should not notice any variation in the semantics of their actions. This is beneficial for a number of reasons:

1. *Direct data movement*. It avoids multiple-hop data movement.
2. *Releasing users from local provisioning*. When data are marshalled and processed locally, users have difficulty obtaining sufficient resources. DARE removes the need for local support teams and local capacity. Citizen scientists particularly benefit.
3. *Specialist stores and systems*. The mappings include access to specialist shared facilities, for example, if some aspects of the data contain personal information, management can ensure that data is stored in locations where the security has been approved as being sufficiently robust.
4. *Exploiting new opportunities*. The digital technology available (software and hardware) all change. The ability to remap makes it possible to try these immediately.
5. *Reliability and accountability*. Governance, ethical goals and the requirement to analyse the quality of evidence result in the need for pervasive collection of histories as provenance, protection of data and of accounting, e.g., a community may commit to limiting its total GHG emissions, c.f., SKA. When data are downloaded and manipulated outside of the system, none of these goals are monitored automatically. Working entirely within DARE provides continuous protection and monitoring.

The balance between stability and rate of scientific advance is difficult to maintain. The DARE platform capability that allows one DARE context to see another DARE context but work independently can be used by governance to accommodate some potential stress points. They need to maintain and evolve a *consistent conceptual space* that extends across their full community of established terminology, adopted standards, methods and reference data. But they also need to let subcommunities and projects they trust emerge who extend and potentially modify that common conceptual space. They may permit branches to develop that **see** and can synchronise with their governed evolving core conceptual space. If and when a branch has extensions or improvements that it wishes to offer to widely used common core, governance will review its quality and value. They then have the option of permitting those running a branch to **promote** selected information from their context to the widely shared context. This is illustrated in Figure 2.

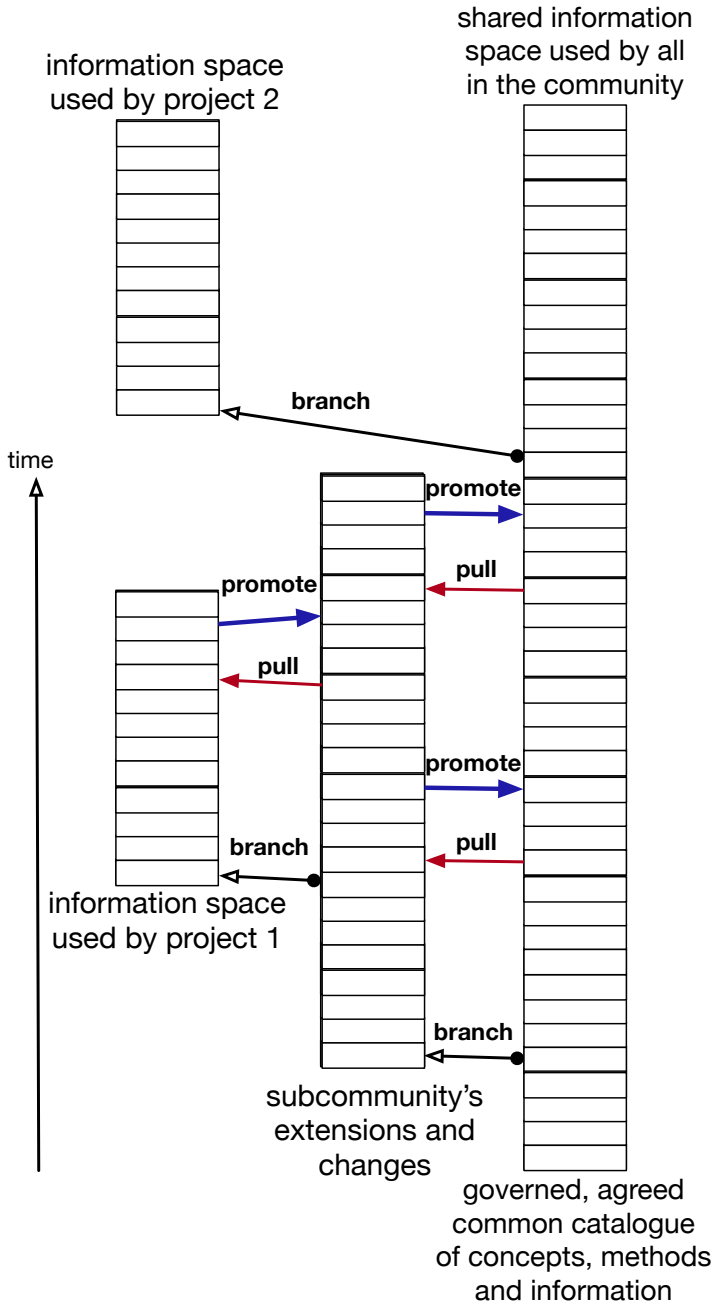

Figure 2: Subgroups and projects may be permitted to branch from the agreed common core.

4 Key architectural innovations and structure

To test and refine the architecture, DARE needs to deliver robust prototypes. To build and refine these rapidly, DARE will develop a framework. That framework will be pre-structured with an initial common conceptual space. This will be pre-populated with built-in information, methods, tools and services. That common conceptual space and its pre-population we refer to as the *DARE core*.

Instances of the DARE framework and core tailored for a particular application community are called the *DARE platform*. Initially we are developing two instances of the DARE platform, one for the computational seismologists (WP6 – see Section 7.1) and one for the climate-impact scientists (WP7 – see Section 7.2). The seismologists need to run computationally demanding simulations to develop improved methods of predicting the impact of an earthquake, of characterising earthquakes and of characterising the errors inherent in such processes. The climate-impact modellers use the output from large simulations to determine more specific and localised consequences with clarity about inherent errors. In both cases, the domain researchers control the work through selecting

inputs to workflows and judge the validity of evidence produced. The DARE platform must provide them with the controls and capabilities in an easily used form that can support long-term research into mitigating the impact of environmental hazards. To do this the DARE platform will:

1. ensure that as much as possible is shared between application communities, and
2. enable rapid exploitation of the DARE architecture for new applications.

The framework delivers DARE's architectural goals by organising the integration and use of an open-ended set of ICT components, tools, services and systems. It builds on a massive body of work developed in previous e-Infrastructure and science gateway R&D projects, including its own systems, to deliver high-performance computation, data handling and communication.

At the top level it comprises three integrated services presented through one API:

1. *Workflows-as-a-Service* (**WaaS**) that performs the actions required by a data-driven application domain including all the organisational and technical supporting activities.
2. *Protected Pervasive Persistent Provenance* (**P4**) that keeps records of all that is done and supports multiple uses of those records.
3. *Common Conceptual Core Catalogue* (**C4**) that helps researchers build and share a harmonised view of multiple information sources and services covering whatever they need. It holds all of the items of interest to them. Or more precisely, it knows how those items are referenced, where to find them and how to use them. It enables users, projects, groups and communities to build and re-use their information space, supporting progress and collaboration. It holds the information needed by software and specialists to interpret users' intentions consistently and correctly.

These are introduced in more detail below. Instances of the DARE core will be tailored to support each application domain and its community. Figure 3 illustrates this for some of the roles involved.

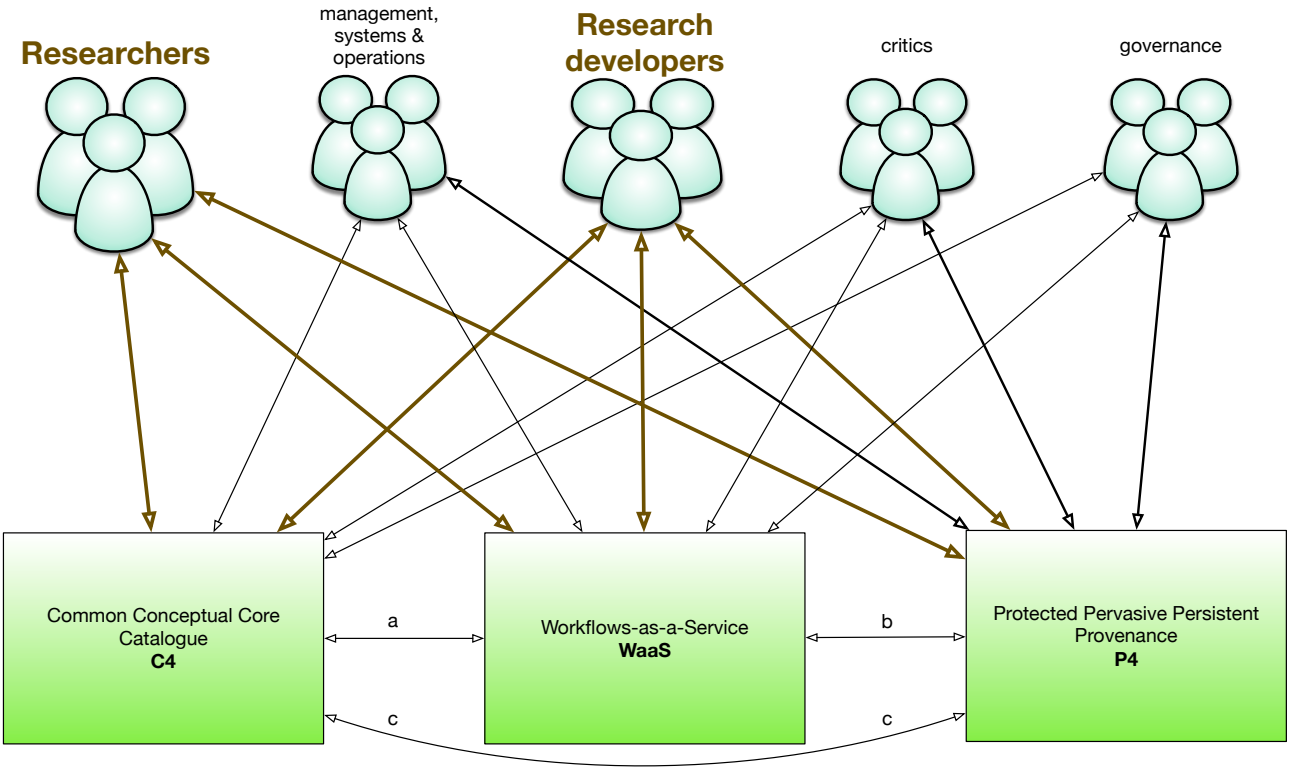

Figure 3: The principal data-intensive ICT pillars supporting instances of the DARE architecture. It shows services delivered to five clusters of roles required for large-scale and long-lived endeavours to address the Earth's pressing research challenges. Consortia of many autonomous organisations are formed to muster the breadth of skills and resources required. The DARE project focuses on just the two emboldened clusters of roles by collaborating with two ESFRI consortia: EPOS and IS-ENES.

4.1 Workflows as a Service (WaaS)

The architecture connects its users to continuing advances in data-driven computation by delivering Workflows-as-a-Service (**WaaS**). **WaaS** enables easy composition and deployment of data-intensive workflows on cloud platforms in a scalable, efficient, optimised and robust manner. Since all the software needed for running workflows and their dependencies should be prepackaged (e.g., by using containers), it substantially reduces the time (and the possible human errors) spent by scientists building such systems by themselves, which consequently allows them to focus on their research.

A WaaS handles any form by which users specify what they want done, i.e., their *intention*, e.g., by using Python, R or scientific workflows in any language. Tools will request actions via APIs, which may convert those actions to workflows; the underlying mechanism should not disrupt users' perceptions. Some professionals, especially research developers, may explicitly author methods as scientific workflows. Teams may then combine and refine these formalisations to improve the methods. To improve productivity, extend the retention of meaning (interpretation of intent) and to lower adoption barriers, WaaS minimises the direct use of technical and organisational details from all professionals except system specialists[20]. This will be achieved by developing and deploying relevant actions, abstractly named to reflect their intention. These actions will include those already coded, those coded by research developers, those that support the data lifecycle and those that enable the use and management of DARE platforms. A few *primitive* actions will be fundamental, authoritative (cannot be redefined) and embedded in the core framework. The remainder will be looked up in the catalogue to obtain currently available implementations that fulfil the functional and operational requirements expressed by the invoking user. This will occur whatever the tool or language is being used to activate those actions, though caching or pre-binding may avoid this happening repeatedly during one enactment[21]. This ensures that repairs as the digital environment changes, e.g., an external service changes its response encoding, only need to be performed once.

WaaS chooses optimum enactment pathways within set constraints and organises the required mappings, deployments, executions, record keeping and user feedback, i.e., it consistently *interprets* their expressed intentions. For researchers, the *incentive* for adoption comes from increases in power and productivity. For funders, providers and governments the *incentive* is improved return on investment (RoI). The implementation during the DARE project will clarify the properties that a WaaS must deliver and demonstrate they are achievable in the specific cases selected to meet the needs of the two application domains. WaaS will deliver advances towards goals 2.a and 2.b (see Section 2.1). Internal use of WaaS may hide complexity contributing towards goal 2.c.

The technical requirements for subsystems fulfilling the WaaS role are presented in Section 8.1.

4.2 Protected Pervasive Persistent Provenance (P4)

The second major element of a DARE platform is Protected Pervasive Persistent Provenance (**P4**). As will be explained in Section 9.2, this builds on the S-PROVflow system [Spinuso 2018]. P4 organises and controls the recording of what is happening in a W3C PROV standard form and preserves (and if necessary, protects) those records for as long as they may be useful. It performs two roles:

[20] However, the requirement for incremental adoption and minimal disruption means that DARE must continue to enable the continued explicit use of details by application professionals, particularly research developers. Some application-domain experts also take responsibility for mappings to chosen facilities. They too need to see what is happening and potentially direct mappings. Consequently, the shared-information space supported by C4 is a continuum, with no explicit boundaries.

[21] For example, if the request is encoded in Python, the names will be methods in the *C4* class, and the local instance of C4, called *theC4*, will contain the current version of those actions with the same names, if they are concerned with the use of C4.

1. It provides *definitive evidence* of what has happened which can be used for many purposes (see below).
2. It acts as a *lingua franca* for communication from software elements to humans (via tools, visualisation and search systems). Previous systems had developed consistent interpretation from intentions to ICT-enabled actions. DARE leads the way in completing the interpretation consistency goal by using W3C-provenance standards in the response path.

P4 builds on the work in the VERCE project and its successors [Atkinson *et al.* 2015] and on Spinuso's research [Spinuso 2018]. These built on the provenance work of W3C, digital librarians and the scientific workflow community developing the PROV standard[22]. DARE is extending the collection and control mechanisms in P4 to make provenance pervasive, i.e., captured for every significant[23] action in method and data lifecycles. Any gaps leave gaps in professionals' ability to optimise, verify procedures and review the strength of methods and evidence – assessing the *intent* or the *interpretation*. The *incentives* for adopting provenance, i.e., making it pervasive, include reduction in chores, powerful tools [Spinuso 2018] and preparation for meeting publication and curation requirements [Myers *et al.* 2015]. However, premature and unauthorised visibility of provenance records acts as a severe disincentive; therefore, the architecture provides professionals with control mechanisms. The value of provenance to be used as evidence in defence of one's actions depends on its protection[24], so that you cannot be framed by someone tampering with the records.

The technical requirements and approach to achieve the P4 goals will be presented in Section 8.2.

4.3 Common Conceptual Core Catalogue (C4)

The explicit inclusion of Common Conceptual Core Catalogue (**C4**) is a significant innovation, although there is much prior work on catalogues, ontologies and metadata on which it builds, e.g., [Quimbert *et al.* 2018]. C4 holds a dynamically changing population of *entries identifying*, *naming* and *describing* all of things a federation needs to share in order to collaborate. It potentially extends to hold the work of organisations, subcommunities, groups and individuals. A local view provides the context for each worker. As they work it gets updated to record their progress. This persistent state allows them to resume from where they left off when they reconnect. If they update shared entries, then their updates are visible to those who share the entry[25].

C4 contains a sufficient diversity of things, e.g., *data*, *collections*, *services*, *people*, *sessions*, *methods*, *actions* and *runs*. The logical interpretation of C4, independently of the actual technologies involved and the means of orchestration of the storing mechanisms as well as the integration of existing collections, is that of a single, unified catalogue as catalogued entities exist within a unified universe, referencing one another and being subject to a common set of consistency and integrity restrictions. Crucial differences in treatment are captured by assigning each catalogue entry a *Concept*[26]. A *Concept* reflects the human and software requirements for differences in the interpretation of entries. This may be refined to support any required detail by introducing new concepts that are subtypes of existing concepts.

Concepts are introduced as a form of knowledge organisation. They permit users to focus on and understand the concepts they need to use. They enable software to factor out common information

[22] Provenance standards https://www.w3.org/standards/techs/provenance#w3c_all

[23] Communities choose what they consider significant. DARE should support their choices. It should encourage comprehensive detailed recording by providing visualisations and summaries that make the detail worthwhile.

[24] Protection against update might depend on signatures of run traces being stored in a block chain.

[25] When a user creates a new entry, it is normally in their context. The use of the same name by others is disambiguated by an implicit prefix, just as it is with file naming. With authorisation the scope may be widened to a shared space; again analogous to shared file systems.

[26] Concepts start with an uppercase letter, c.f., class names in Python.

that applies to every instance of a concept. Humans find it helpful to consider concepts in three steps [Trani *et al.* 2018]:

1. *Concept* The name and nature of the concept – what kind of thing does it denote;
2. *Representation* The representation, properties and uses of the concept; and
3. *Population* The population of instances of the concept that are relevant for their work.

C4 encapsulates the specification of each concept, either by explicitly defining it or by incorporating and appropriately extending its already existing definition (provided and exposed by authorities external to DARE).

Every entry in C4 is an instance of a Concept. Many entries refer to data held elsewhere to avoid redundant data movement. This widens the scope for optimising placement.

Logically, any user or system using an instance of the DARE platform sees one instance of C4 presenting their view of and context in DARE's virtual world – see Figure 1. That view includes an integration and harmonisation of many resources. Some of these may be other C4s. The technical approach to C4 will be presented in Section 8.3. Here we introduce some of C4's potential features that will be implemented incrementally in DARE, when they are needed and can be evaluated.

4.3.1 Starting to tailor an instance of DARE platform

Professionals who are setting up a federation will pre-populate C4 with concepts and entities their community has agreed is the *common conceptual core* needed for effective collaboration and support for their work[27]. This extension of the *DARE core*, which is already rich in concepts, information, methods and resources, to meet requirements of an application community, we call *tailoring*. Professionals conducting their routine work will then use this *tailored DARE platform* to perform their actions. The implementations of these actions (often via a WaaS) will consult C4 about how to interpret their intention. Many actions will find method encodings, data, parameters deployment requirements and orchestration scripts from C4 and will add further entries to C4. The descriptions will contain access and visibility rules, as well as authorisations, so that professionals can control when and by whom their entries are seen. Part of the initial shared information context in C4 will be common to all instances of the DARE platform, and therefore be pre-established. For example, the descriptions and encodings of all the built-in, *primitive*, and under the hood, *internal*, actions will already be in C4.

4.3.2 DARE helping professionals handle complexity

C4 is a substantial step in tackling the complexity that threatens to overwhelm professionals as the research challenges they face become more multifaceted, as methods and models draw input from more fields and as consortia respond with a growth in their diversity. It is key for achieving and sustaining the *consistency of interpretation* required. It provides a foundation for delivering agility by allowing subcommunities to emerge and explore rapidly using their own C4 only where they differ. The view of C4 maintained as standard by a community delivers stability to the majority of professionals who want it.

Today, researchers integrate multiple sources of information by personally keeping track of many sets of concepts, naming systems, instance identifiers and annotations. They master complex relationships and technical detail to directly handle their data, and to initiate and oversee separate implementations of method steps. They often manually record progress, marshal and transform intermediary data and explicitly find and manage resources. They are forced to assume that their co-workers from other fields have a consistent *interpretation* where their concept spaces overlap. Within a profession training develops this consistency.

DARE sets out to relieve them of these chores and need to understand technical detail. It does this via C4. C4 helps researchers cross discipline boundaries, accelerating the induction of new specialisms. Specialists supporting their campaigns focus on key aspects of descriptions within the

[27] The DARE platform helps this process in two ways. It is already equipped with a substantial common core, that is needed by most communities, and it has actions and tools to support this tailoring process.

part of the concept space they work on. It normally takes a substantial period of collaboration for mutual understanding to develop sufficiently between disciplines. By combining multiple viewpoints into one supported framework, we hope to accelerate this process. It will still be necessary, but it should be more focused and better supported with accessible definitions of concepts and consistent tools for using them.

Multinational bodies develop bundles of *agreed* concepts, instances and representations for substantial patches of the conceptual space. Updated curated instances of these bundles are often supported by archival services used directly by researchers. When their work uses multiple services, these may adopt different standards. Those setting up the federation need to understand and exploit such pre-fabricated bundles. They may exploit multiple, incompatible bundles, selecting and mapping subspaces linked together with conceptual-space patches[28]. As their federation evolves, they may import further bundles. Meanwhile the external organisations revise their bundles and the federation organisers need to decide when and how to import these changes. The tools and functions of C4 cannot automate such conceptual processes that depend on long-running negotiations to formulate agreements [Trani *et al.* 2018]. However, the tools and conceptual framework of C4 should support their work well. This can only be evaluated in the long run and in large scale collaborations. However, DARE should observe such work in the larger communities behind WP6 (EPOS is using Trani's approach to concepts and catalogues [ibid.]) and WP7 (IS-ENES using a large NetCDF formulation of relevant concepts and associated metadata). DARE should implement C4 so that it accommodates their current bundles with a design to eventually accommodate evolution of adopted standards.

Professionals also use version management systems that support collaboration, e.g., GitHub, shared information spaces in a VRE [Martin *et al.* 2018, Pierce *et al.* 2018], workspaces on services and directories on local file systems. They build collections of files, often encoding significant semantics in file names or as annotation embedded in files. Researchers also draw on component libraries, simulation systems and data services that evolve independently. C4 will present a consistent view that maps a coherent integration of this complex space[29]. It will support the operations by users and software via this view to help tools, individuals and communities achieve consistent interpretation.

To enable professionals to control incremental adoption they choose which target collections they map and how they use them. Three modes of inclusion are supported.

1. **see**: when developers choose to **see** another 'catalogue', e.g., one supported by an archival service, they **see** a specific catalogue by supplying its *referenceURI*. They may specify a selection expression to see only a view of that source and specify transformations on the metadata and data used from that source[30]. The user or community must be entitled to access the source. The user may not modify the source through this view. There may be usage constraints and accounting.
2. **use**: enables developers to see and update another 'catalogue'. Similar selection and transform mechanisms to those supporting **see**[31] are provided, but now transformation is two way. Similar controls and accounting may be required. The catalogue may be a file path specified by regular expressions, so the subset of files that is incorporated. This will be used

[28] Scientific insights introduce new ways of combining information from multiple methods of observation and multiple models that then have a variety of simulations and simulation outputs. Research scientists spot these links and invent new methods to answer questions they recognise as critical. The prefabrication of C4 and the embedded conventional compositions it is tuned to support must not inhibit exploration of these new possibilities.

[29] For example, it may catalogue versions of SoftwareStacks to denote the evolving context in which work progresses. These versions would also be used by P4.

[30] A library of such transformations will be built and shared [Martin *et al.* 2018]

[31] This time they need to be two way.

to permit co-existence of work using shared-file or version-management systems with work using DARE.

3. **ingest**: enables users to upload data from their own or acquired sources in any initial form, for example, records from a temporary deployment of extra instruments. Again, the set of entries ingested, their concept, their metadata and other descriptions will be specified and honoured by the ingest mechanism. Users must be entitled to use this data; often it is their own files. In this case any referenced data are ingested and stored according to the current **preserve** settings – see Section 8.3. From then on, the users have direct access and update powers over their ingested items unless they specify restrictions. This is a one-off operation making a copy in the DARE platform's remit[32]. A later ingest may add new entries or update an existing entry, e.g., a collection.

C4 also supports direct creation of entries and the addition of annotations to existing entries. If they are entries derived via a **see** relationship, the annotations are added via a local C4 proxy instance. APIs and tools for searching, browsing, visualising and updating, similar to those for P4 will incentivise adoption by delivering convenient and relevant power[33] [Myres *et al.* 2015]. Community standards adopted by those who tailor C4 to set up a federation will have a dominant influence on use and entries.

4.3.3 Humans and Software communicating via the Catalogue

Once items are in C4, they may be used by any human or software element that has access to the C4 instance. Operations include running any permitted action on any combination of entries, such as deletion, annotation, modification and export. Implementation strategies for these operations and for C4 are considered below. The concept of an entry determines the permitted operations and the interpretations of actions on that entry. The metadata, mostly inserted automatically, will include human and machine identifiers, synonyms, links to semantically equivalent or similar entries, version chains and *interpretation* rules. The *incentive* for adoption will be the reduction in the need for humans to create and remember such things, leading to improved productivity and fewer errors. The simplification of specifying mappings and revisions in one place will be the incentive for research developers and software engineers. The machinery for managing C4 will become a key tool for those responsible for setting up and sustaining inter-disciplinary and inter-site collaborative behaviour.

4.3.4 Supporting the systematisation of Concepts

The concept mechanism supported by C4 is an abstraction of types, classes, kinds, etc. to allow people and machines to recognise a taxonomy that helps them understand and use the items in C4[34]. Items with the same concept have a specified subset of information elements in common and have a similar range and interpretation of permitted actions. The concepts are organised by an **isa** hierarchy, that governs the information they carry and the actions which may be applied to them. The top level of this hierarchy is predetermined in the DARE core, as illustrated in Figure 4 and described in Section 8.3.2[35]. The set of predefined concepts is usually extended as the DARE core is tailored to shape a DARE platform for an application community. They then extend it with further specialisations of the concepts to suit group and individual needs and to capture aspects of semantics vital to their work and practices. It is important that users can introduce conceptual refinements they need as they conduct their work.

[32] Communities will normally have **export** mechanisms to perform the inverse, delivering data in a variety of forms.

[33] Eventually, provenance trails will be accessed via entries in the C4. In consequence, the same tools will allow selection, clustering and visualisation of the entries, including provenance.

[34] It explicitly introduces a taxonomic framework to help communities discuss and manage their data. By conceptualising sub-populations of data items, they can agree on the requirements, behaviour and typical properties of all data items denoting that concept. Leaving only individual variations to be handled.

[35] The actual set of core concepts is not yet fixed; it will be developed incrementally during the next 12 months.

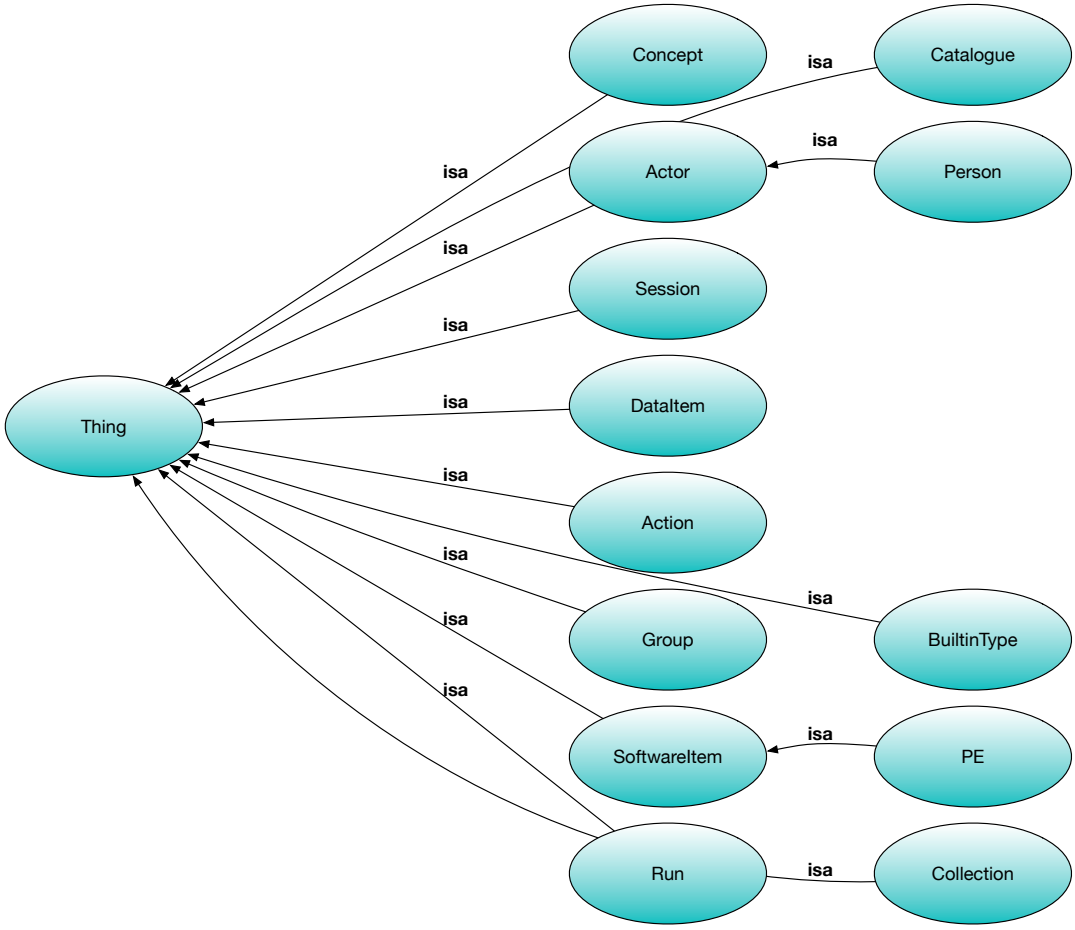

Figure 4: an example of the top-level concepts delivered as primitives in the DARE core. Further conceptual structures are developed in Section 8.3.

When users or software insert an entry into C4, they specify or infer its concept. Concepts' names begin with an uppercase letter, e.g., *Workflow*.

The built-in concepts shown in Figure 4 will be supported in all instances of the DARE platform. In DARE there are further built-in categories that are subcategories of these categories, e.g., *Human* **isa** *Agent*. These built-in categories are *authoritative*; that is their definitions may not be changed or over-ridden except in a developer context – see 4.3.6. Those shown have the following interpretations:

1. **Thing** the minimum contents and behaviour of every entry managed by C4.
2. **Concept** the name for recognised commonality of behaviour and content.
3. **Catalogue** is a description of a view of another collection of information indexed via a catalogue that contains metadata about the available information and describes how it represents that information as data, e.g., by exploiting the DCAT standard[36].
4. **DataItem** the data that is being acquired, input, produced, managed, used, preserved and archived –the whole lifecycle should be supported as in the ENVRI RM[37].
5. **Service** an often independently run facility for delivering resources, such as data, storage and computation, and for performing actions, such as issuing a PID.
6. **SoftwareItem** any means of representing actions that can be created, moved, stored, edited and run in an *appropriate* context, e.g., a Python program, a *PE* and a dispel4py workflow.
7. **Action** any operation or graph of operations that may be carried out. They are used to perform all the actions requested by users and all the sustainability and housekeeping

[36] Data Catalogue Vocabulary DCAT https://www.w3.org/TR/vocab-dcat/

[37] ENVRI Reference Model (RM) https://wiki.envri.eu/display/EC/ENVRI+Reference+Model

actions. They may be *primitive*, i.e. built-in and *authoritative*, or *composite*, i.e., composed of primitives. Actions may be user or externally defined.

8. **Actor** is any entity capable of carrying out an action, such as a *Person*, an *Organisation*, a *Group* or a software *Agent*.
9. **Run** the start of the records needed to support human and software requirements for reviewing or reperforming previous actions. These allow direct recording of events in the C4 and deliver strong linkage between C4 and P4.
10. **BuiltinType** a number of primitive types already known to the system with corresponding literals and handling functions in the parallel Python representation so that developers don't have to develop them individually. Instances of these **BuiltinType**s may appear in descriptions, in parameters and as results from functions. They do not require container objects with explicit entries in C4. They may also be values of entries in C4.
11. **Collection** will be small set of supported collection types, e.g., **list**, **set**, of anything that could be the subject of an entry in C4. A collection may constrain the **Concept** of their members.

4.3.5 Supporting Collections

Both humans and software work with collections of items. They build them, often incrementally and iteratively. For example, a set of data items that will form the input to some process or a set of contributions being accumulated to establish sufficient evidence. They then iterate over members of the collection, performing actions, such as viewing, listing, cleaning, mapping, analysing, computing and data handling. Making such collections first-class citizens in C4 enables users to build and revise collections without having to marshal them locally on their own resources. They should feel they control such collections. C4 will tentatively support three forms of collection:

1. **list** denoted by **[]** is an indexable list, c.f., Python. Specification of the concept of objects it may contain can occur, e.g., an instance, *st,* of *SeiesmicTraces*[] could be holding a sequence of *SeismicTraces* a user is building to test her methods and models. She could then run actions applying to every member of *st*, or to a selection of *st* members.
2. **set** denoted by **{}** is an unordered collection of unique elements.
3. **dictionary** denoted by **dictionary**(*tag1 value1*, *tag2 value2*, …) is a means of building an explicit set of name, value pairs, e.g., as a parameter set or parameter defaults, c.f., Python dictionaries.

The collection constructors can be used recursively, e.g., a set of lists of tables.

These will operate at the concept and the instance level. The exact choice is very much up for discussion at present. The crucial point is to allow users to perform all the actions they wish on them or their members without requiring co-location of their representation or members. They should be able to use their normal tools and methods to work with collections.

There are some implicit collections. At present, these are all sequences. For each concept, the identifier using the same sequence of letters except that it begins with the corresponding lowercase letter is an instance of that class, e.g., *workflow* that iterates over the sequence of all instances of that class in the order they were defined. That sequence of instances is named by adding '*s*' to the concept name, i.e., *workflows* denotes all workflows and *concepts* denotes all concepts[38].

There will be further implicit sequences to support iteration, in particular, over responses to queries. Collections may be streamed, distributed, partitioned and handled incrementally, without the collection being locally materialised.

4.3.6 Enabling DARE to support developers

C4 is also the foundation for achieving agility while preserving stability. Moderate innovation in methods and procedures does not require special facilities as it does not have far reaching effects. However, substantial changes, such as repairing or improving a workflow fragment embedded in

[38] Suppose there is a category *X*, then *x* denotes an item such that *x* **isa** *X*. The list of all *X*s in C4 is denoted by *xs*.

many methods or changing the way a widely used primitive action, such as **preserve**, is authorised, parameterised and implemented may take a substantial time to accomplish and have significant effects. This is addressed by different *work contexts* as shown in Figure 5. The majority of professional activity is confined to the *production context*.

Projects, because they are focused on innovation, tend to disregard this difference. We will ensure that this does not mislead DARE's design and evaluations. When innovation is required the developers require freedom that could not prevail in production to change fundamentals, such as the implementation and mapping of built-in *authoritative* definitions, methods, the deployments they require, their mappings, and the VM images in which they are enacted. Developers need to install or change key software components. They need to observe and examine runs in deep detail. As far as possible in the development context the interpretation of C4 entries yields the same data, methods and resources as it does in production. But changes that will modify persistent external stores must be emulated otherwise test runs will leave a legacy of phantoms. A particular case is the updates to P4. P4 will be a critical source for understanding the *a priori* state, methods and history. It may also provide an important window into the runs during testing, but those, often incomplete and incorrect runs should not be recorded undifferentiated from production runs with normal visibility. Consequently, the differentiation between production and developer contexts affects the whole DARE platform as shown below.

Modification of critical mappings in the C4 that accumulates the developers' work will establish this different context. It will **see** the relevant production context, since its C4 provides a catalogue of all that is relevant. Eventually, after testing (often after independent validation to attain approval), the developers will select a subset of their products that are ready and **promote** them to the C4 that supports production. Where these modify an existing entry, this will produce a new version with a reference back to the prior version. As existing work maps to explicit versions (identities) those not wanting the change may avoid it. However, humans and software will be selectively notified, or be able to enquire about, the new facilities – they can selectively rebind to adopt the revisions.

The automated access to see the production context where an issue occurred reduces a developer's set up time[39]. The management of persistent changes as a consequence of replaying actions also reduces the requirement on them to contrive safe re-runs. Workflows supporting **promote** will handle the consistent extraction for a set of changes and build the corresponding updates in the target C4s. Tailoring of those workflows for particular communities will introduce validation and governance requirements. This should accelerate the release of repairs and extensions, reduce installation errors and help both production professionals and research developers. It is the foundation for delivering research agility.

[39] A fully developed DARE platform will include the ability to **see** external catalogues, e.g., those provided by archival services, so that the contents of those catalogues appear to be in the DARE platforms remit, even though they are independent. The DARE API will present the catalogue functionality of a DARE platform instance. Combining these, the development context will **see** the production platform without perturbing it. The promote process requires a workflow to be run once permissions have been negotiated.

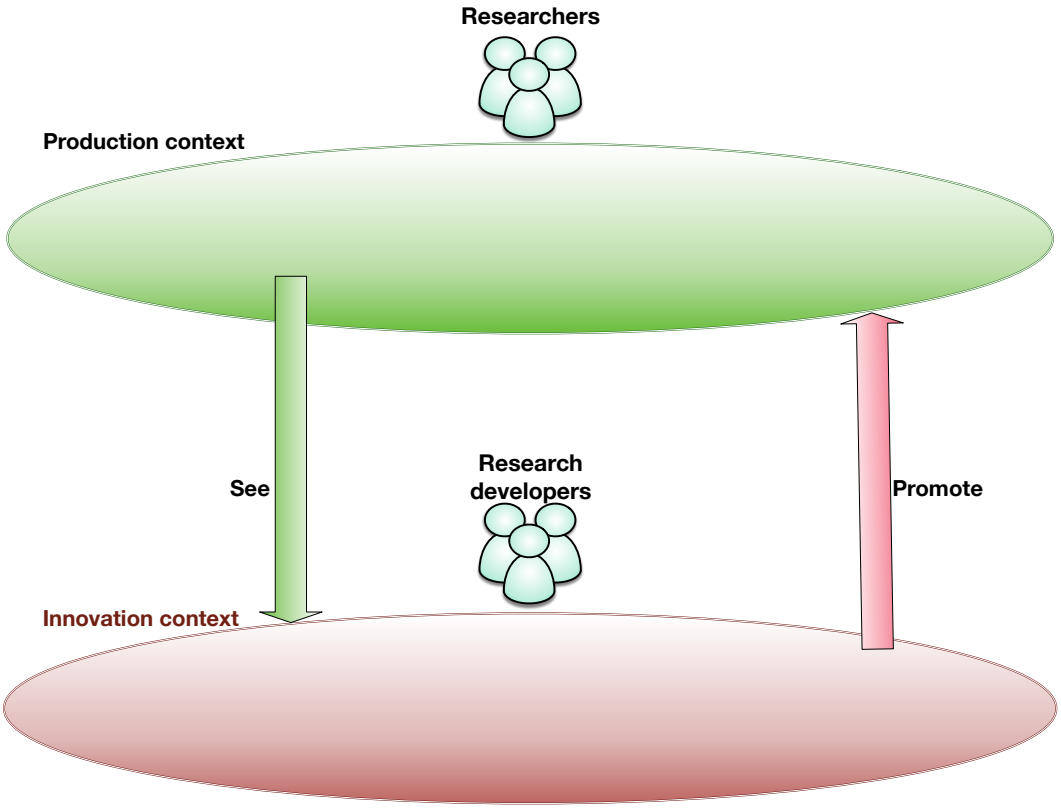

Figure 5: Differentiating and supporting fluent transitions between production and development is key to enabling rapid response to opportunities or to problems.

4.4 Delivering Computer-Supported Collaborative Working (CSCW) using DARE

Each consortium will set up, tailor and maintain an instance of the DARE platform to support its communities and their data-powered work – see 4.3.1. Systems, data and software specialists may be aware of the internal structure just described. Certainly, research developers will be. These will use the tools, libraries and APIs delivered and if necessary, internal functions supported by specific pillars. They will work on populating C4 with what their communities need.

In contrast, the majority of professionals and all external users will use an instance of the DARE platform constructed using the DARE platform as an integrated whole and be unaware of internal subsystems. They will work through an API for the whole system [Klampanos *et al.* 2018]. Tools and portals will develop against that integrated view.

Each consortium, working on its cluster of data-driven R&D campaigns, will set up, tailor, manage and maintain such an instance of the DARE platform. These will have a number of DARE primitives in common as well as a substantial common subset of C4 needed to meet recurrent data lifecycle tasks, as identified by ENVRI[40], to support common geo-political, geophysical, statistical, data-science, mathematical, scientific and presentation needs. This common foundation can be cloned to make new instances for new communities that exploit the DARE framework and strategy. Their support team will then add to this to meet their community's needs. As their community changes its goals, composition and methods they will continue this revision. Consequently, the different instances of the DARE platform will diverge and push for different revisions. Further needs for revision will come from changes in the digital context, both from technological advances and from the revision of provider and funder business models.

An open project supporting DARE will therefore be needed to deliver two things to sustain the DARE-dependent consortia:

[40] ENVRI Reference Model (RM) https://wiki.envri.eu/display/EC/ENVRI+Reference+Model

1. Successive versions of the implementation of the DARE core mapped to available services with mechanisms for installing and migrating each instance's state and user community.
2. Successive versions of the DARE core[41] to offer the new capabilities and conceptual improvements with mechanisms that allow each federation to selectively and progressively adopt the advances.

To gather adopters and to achieve longevity sufficient for long-running research infrastructures and campaigns, this plan for DARE maintenance is critical[42]. To make that feasible, the architecture must use shared solutions that either have a well-proven maintenance strategy or have a sufficiently strong base of open-source collaborators that maintenance is well amortised. The architecture must also keep the remaining integrating software simple and well-structured to make long-term maintenance feasible [Atkinson *et al.* 2016].

5 Strategy for incremental adoption

New capabilities delivered by DARE motivate research leaders to push their groups to adopt DARE (see Section 3). For professionals to feel and be in control they must be able to choose when and how to adopt the DARE approach. For application-domain experts to avoid unacceptable disruption or perceived risks, they must be able to adopt the DARE approach when they choose. On the other hand, the economies and productivity gains promised by DARE result in managers and funders also pushing users to adopt DARE. Practitioners responsible for quality and consistency of their analyses and data products may be reluctant to change any aspect of their established working practices. Compromises based on incremental adoption should help communities navigate these adoption challenges[43]. This incrementality must take two forms for individuals, groups, projects and communities:

1. *Co-working*: adoption for parts of work, i.e., some of the activities continue with prior methods while others take advantage of the DARE platform. This is likely to endure for a long period as the intellectual and technical investment in established methods is very significant and it will require continuing effort to wrap them in the DARE framework when that is prioritised by a user community.
2. *Mixed communities*: a subset of a community adopts the DARE indirect-action ethos while others retain direct control managing their own use of technology. This is orthogonal to and will overlap with the co-working incrementality.

Consequently, the DARE architecture must function effectively in a 'mixed data economy', where direct management of data, with explicit application of computation, runs alongside DARE's conceptual virtualisation, with smooth interworking between the two systems for as long as is required. This may add complexity for the implementers of DARE, but it should not introduce complications that inhibit adoption. DARE's approach to this combines three tactics, each of which has socio-economic and technical aspects intertwined.

1. *Intellectual ramps*, allowing users to incrementally understand and use the system.
2. *Persistent mappings*, systematic ways of bringing external persistent state into DARE.
3. *Action across boundaries* making the actions available outside DARE include external presentations of DARE supported actions and enabling external actions to be discussed and performed within DARE.

[41] The set of *Things* used by a large proportion of the set of application domains for which DARE has been tailored – initially computational seismology and climate-impact modelling. This will include all the built-in authoritative items, the common tree of concepts, the common set of actions, the common population of terms, the common sets of data, the shared documents and commonly required background data.

[42] Software Sustainability Institute https://www.software.ac.uk/

[43] These are characterised by the technology adoption curve https://en.wikipedia.org/wiki/Technology_adoption_life_cycle.

We review the DARE approach to each of these in turn.

Intellectual ramps: early adopters, support teams and the application-tailoring process will have developed and tested some key activities that many people in the community need to undertake. By setting these up in a form that can easily be copied and edited to incrementally adjust them to a user's purpose, a user has to learn very little to get started. For example, a procedure using an established method on a set of data with typical control parameters can be set up as a Jupyter notebook[44]. After success with making small changes the user incrementally learns more to explore and exploit the power of DARE for their community. Direct support and encouragement from peers who have travelled further up a ramp or have explored an adjacent ramp is an amplifier of adoption. This needs to be supported and encouraged.

Persistent mappings: the identification of external entities outside the DARE platform so that they may be used within it and for some of these the possibility of actions conducted under the aegis of DARE changing those entities. The latter is relatively rare as the external world for any data-driven science is always large and under diverse independent governance models, so permission to change things is usually limited. The former is a dominant part of DARE, as reference to external repositories and services is key to avoiding unnecessary data handling and excessive local resources. Consequently, individual entries in C4 refer to external entities. These can include the parts of work not yet incorporated during co-working and the work of colleagues who have not yet joined the DARE camp in mixed communities. This can be handled on an item-by-item basis, but that would be very laborious and so DARE provides various ways of specifying choices generically. These include:

1. Mapping external collections into the DARE context in a way which makes them **see**n by actions performed using the platform. Where collections are behind a service that supports queries any action on a Concept yielded by those queries, may request whether the collection has relevant entries and then arrange that the workflow includes those entities. Often external information is organised by file-system structures, e.g., for individuals, for VRE shared spaces and for collaborative development environments, such as GitHub. In these cases, the generated query may use aspects of file names or examine metadata embedded in files. Typically, the selected files then become entities to be used in setting up or running the workflow. These arrangements can be encoded in the definitions of concepts. The choice of concepts then supports incrementality in bulk, with a coverage of all future instances of a concept. We expect to see communities progressively bringing more concepts into the DARE framework.
2. Wrapping external services inside process steps that are described and available as actions within DARE, e.g., the allocation of PIDs for DARE generated entities in the official service chosen by a community or the archiving of products in a community's established archival and curation services. These actions may then be used directly via users' GUIs or embedded in the encodings of methods that users use.
3. Using external persistent systems, such as databases, archival stores, filing systems and so on, as if they were within DARE via a two-way mapping process that brings them into the DARE platform's remit and allows them to be worked on contemporaneously by those using DARE and those working external to DARE modulo concurrency and consistency constraints. These will typically be file systems or collaborative development environments directly mapped to yield *DataItems* and collections of *DataItems* or to concepts that are represented by underlying *DataItems*. Updates may be controlled in either direction, e.g., with **pull** and **push** actions.

Action across boundaries initiated from within DARE can be handled by wrapping external services and tools as outlined above. But actions initiated externally to DARE should also be well supported. This will depend on DARE presenting a sufficiently rich API to external services and

[44] Jupyter notebook tutorial https://www.dataquest.io/blog/jupyter-notebook-tutorial/

tools. These should be able to carry out any DARE action subject to security and AAAI checks. These are considered in Section 6.1.

In summary, users of DARE individually or as groups and organisations should be able to choose when they start working via the DARE platform. When they do, they should be able to continue doing some of their work outside DARE and link easily with that work. It is anticipated that the partial-population adoption of new technology normally experienced will prevail. DARE should enable the adopters and the non-adopters to continue to collaborate. The design and structuring of C4 is intended to facilitate this.

6 Current DARE platform status

This section describes the state of operational systems, technologies, software systems, services and platforms from which the next phase of development starts. It covers the aspects of the current digital, technical and organisational context that constrain, enable or in any other way influence the architecture. It focuses on the architectural aspects and the longer term. It is informed by current work delivering prototypes to the two user communities.

6.1 DARE platform

Initially, the platform is a framework for setting up the services that the two use cases need so that testing and evaluation can be carried out at challenging scales. The exact requirements that are common across application domains will emerge. By the end of DARE this platform will be a consistent and integrated environment in which application communities can develop and run software, services, data and tools that meet their communities' needs, accommodating the multiple roles and modes of use introduced in Sections 2 and 3. The consistent relationships between the three digital pillars will be sustained by this framework so that adoption by new communities can be rapid. The DARE architecture will yield this common framework, with good properties of convenience for both research developers and research professionals conducting their work. Teasing out the composition of this common platform and the service boundaries, APIs and non-functional qualities is a research activity DARE will sustain. At the present time the current composition is determined by the low-level building blocks, the required interfaces to contextual e-Infrastructures and services that may eventually be considered specialised.

It is intended that the final DARE platform will build on top of the current and next generation e-Infrastructures deployed in Europe. But to ensure that the available infrastructures for the development and testing for relevant components meet the requirements of the development teams, WP5 started to combine the resources of the partners GRNET and SCAI. Both partners provide resources in the EGI Federated Cloud[45], which is part of the European e-Infrastructures. In a liaison with available services from the e-Infrastructures: EGI[46], EUDAT[47], INDIGO-DataCloud[48], EOSC-hub[49] and PRACE[50] the DARE resources will be extended.

Altogether, WP5 will mobilise computational resources to serve:

- the software development pre-release infrastructure for DARE's software products,
- the infrastructure required for the final deployment of the DARE platform.

[45] EGI Federated Cloud: https://www.egi.eu/federation/egi-federated-cloud/

[46] EGI: HYPERLINK "https://www.egi.eu/services/"https://www.egi.eu/services/

[47] EUDAT: HYPERLINK "https://eudat.eu/services"https://eudat.eu/services

[48] INDIGO-DataCloud: https://www.indigo-datacloud.eu/

[49] EOSC-hub: https://confluence.egi.eu/display/EOSC-hub+service+catalogue

[50] PRACE: http://www.prace-ri.eu/

The environment for developing DARE's software products has already been set up. This environment is available on GRNET's infrastructure okeanos, a Cloud platform based on Synnefo[51] technology. DARE developers use it directly.

As is envisaged for the final DARE platform, this development environment uses of the Docker ecosystem[52] due to its flexibility, reusability, popularity and compatibility with all major Cloud-provision services. Docker makes it possible to isolate and package ("containerise") the DARE software. This facilitates the deployment and execution of developed applications. The clustering and scheduling tool Docker Swarm offers scalability, interlinking of containers, networking among containers, resource management, load balancing, fault tolerance, failure recovery and log-based monitoring. Docker Compose enables multi-container Docker applications to be run, modelling interdependencies and connectivity among containers, in an easy-to-understand and write YAML[53] description. In the future, it is planned to replace Docker Swarm with Kubernetes[54] that is supported by a larger community and offers a more flexible use and additional features. More details referring to this are provided in D5.1 Platform Infrastructure, Usage & Deployment.

The Big Data Europe Integrator (BDI) platform [Auer *et al.* 2017] that builds the next level of the DARE platform builds upon the Docker ecosystem as well and taking advantage of it for application provisioning. The BDI platform has been developed as part of the Big Data Europe project and is powering several use-case pilots responding to societal challenges, ranging from emergency response during radiological events to transport. The BDI platform is a customised, cloud-ready and modular integrator platform, bringing together commercial and research, production-ready components for big-data analytics. It offers an easy-to-deploy, easy-to-use and adaptable framework for the execution of big-data applications and supports a wide range of common tools for these applications as ready-to-use Docker Compose files. It contains tools to help with composing and configuring BDI instances/environments, and monitoring them during deployment, test and production. Furthermore, it can provide data management and analytics functionality. In the scope of WP5 the BDI was already installed at the development infrastructure and is available to all DARE partners. The next step will be to integrate the available DARE technologies such as S-ProvFlow. All these technologies are described below.

Next to the development infrastructure WP5 deals with the setup of the infrastructure required for the deployment of the DARE platform in production. At this point, WP5 is dependent on third parties. To get access to the necessary Cloud e-Infrastructure, it will be obligatory to enter into discussions with responsible parties from EGI, EUDAT, INDIGO-DataCloud and EOSC-hub.

With regard to authentication, authorisation and identity management, DARE expects to rely on existing solutions, such as AAAIs of the community infrastructures or e-Infrastructures such as the EPOS AAAI[55] and ESGF OpenID[56]. Beyond that, relying on the EGI Check-in[57] service will offer federated authentication and authorisation using multiple sources of identity. That service allows researchers to authenticate by e.g. using their institutional credentials (if their institute participates in eduGAIN[58]) or social identity providers such as Google or Facebook. Backwards compatibility is assured by providing the possibility to identify using an IGTF X.509[59] certificate. Depending on the

[51] Synnefo: https://www.synnefo.org/about/

[52] Docker: https://docs.docker.com/

[53] YAML: http://yaml.org/spec/1.2/spec.html

[54] Kubernetes: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[55] EPOS: https://www.epos-ip.org/sites/default/files/repository/images/ICS-TCS-INtegration-Guidelines-Level-2.pdf

[56] ESFG OpenID https://www.earthsystemcog.org/projects/cog/openid

[57] EGI Check-in: https://www.egi.eu/wp-content/uploads/2017/09/Check-in.pdf

[58] eduGAIN: https://edugain.org/about-edugain/what-is-edugain/

[59] IGTF X.509: https://www.cilogon.org/globus-with-incommon-ca

source of identity, the user is assigned different Levels-of-Assurance, which service providers can use to grant different access permissions. As the service offers interfaces compliant with the popular OpenID-Connect[60] and SAML2 protocols[61], compatibility with a wide range of existing technologies is assured.

The entire approach WP5 follows, including the process of transiting software, computational and data assets from development to release packages and the process of deploying the DARE platform and associated components on publicly available infrastructure will be described in D5.1 Platform Infrastructure, Usage & Deployment.

6.1.1 DARE platform subsystems undergoing development

We list here subsystems included in the DARE platform that are currently undergoing integration and development. In each case we identify its current capabilities and planned role. These are presented with details of the currently prioritised enhancements that will be conducted along with the realisation of auxiliary registries, such as the processing elements registry, and technologies. These comprise the generic data-intensive subsystems and application domain software, developed by their communities. The latest versions of the generic software listed below are available in the DARE GitLab group: https://gitlab.com/project-dare. They are already in the DARE platform and are in the process of being integrated, extended and strengthened. The climate community's global and European collaborations are developing software available from ESGF-hosted repositories (see Section 6.3). Similarly, the computational-seismology community has a number of software sources, including institutional repositories and targeted collaborations (see Sections 6.2 and 7.1). These application-domain software systems are developed concurrently, independent of DARE. DARE will demonstrate how these can be brought into the remit of a DARE platform and used effectively to meet the application domains' challenges.

dispel4py: A Python framework for describing abstract stream-based workflows for distributed data-intensive applications that was developed in VERCE [Atkinson *et al.* 2015] and subsequent projects [Filgueira *et al.* 2016a]. For development and testing this will run on a laptop, and then be shipped unchanged to the production environment – this fluent movement between development and production is key to delivering productivity and agility. Workflows coded in dispel4py can be enacted on a wide variety of platforms: multi-core large shared memory architectures or distributed-memory architectures such as HPC clusters and Clouds running data-intensive systems. Optimised mappings for shared memory, Apache Storm and MPI deliver production performance. New tools will be developed by DARE addressing the mapping to Exareme, and the automatic optimisation of dataflows and deployment to enactment platforms. These new DARE tools will take advantage of data-lineage information that is collected by S-ProvFlow which is available integrated with dispel4py.

S-ProvFlow combines a set of components in support of Reproducibility as a Service (RaaS) [Spinuso *et al.* 2013, Spinuso 2018]. It includes a NoSQL document-store for the storage of the provenance and lineage metadata, a service layer in the form of a Web API and a suite of interactive provenance access and visualisation tools. The data-model specialises the W3C-PROV recommendation for data-intensive applications (S-PROV). RaaS addresses the limitations of grids and computational infrastructures in terms of flexible lineage metadata management services and tools, from its acquisition and representation to its rapid exploitation. Data lineage information, stored and accessible through the RaaS layer, can be used at any stage of the cycle. During the usage of experimenting with tools and analysis software, for the iterative and preliminary validation developed with the DARE technology, until the production of outreach and summarisation reports, eventually feeding the mechanisms that enable the re-enactment of the experiment.

Exareme: A system for large-scale dataflow processing on the cloud [Kilapi *et al.* 2011, Chronis *et al.* 2016, Kharlamov *et al.* 2016]. It offers a declarative language to the users. Exareme is a highly

[60] OpenID Connect: http://openid.t/connect/

[61] SAML2 Protocols: http://saml.xml.org/protocols

configurable system with new functionalities added frequently. These functionalities include federation, stream processing, compatibility with Apache Spark, lossy and lossless streaming compression, privacy preserved data mining, etc. Exareme and its components have been used in several projects including Optique, the Human Brain Project and OpenAIRE.

Semagrow: Semantics and linked-data support [Charalambidis *et al.* 2015, Konstantopoulos *et al.* 2016]. The Semagrow query engine has been developed as part of the FP7 Semagrow project, and has been deployed and improved in later projects, such as Big Data Europe. The Semagrow query engine acts as a query federator between heterogeneous linked-data sources, enabling complex queries. Semagrow features a sophisticated source selection and query optimisation to decide where and which subqueries must be generated against which underlying data sources. Semagrow also tackles the issues of semantic heterogeneity, that is, sources make use of various vocabularies to express the relationships between their data. Semagrow has been used to support complex queries on the metadata for bibliographic publications and on the metadata of agricultural experiments and their results. Moreover, in the context of the BigDataEurope project, Semagrow has been used in various pilots of the BigDataEurope project as a federator between linked-data, geospatial data and other data stored in NoSQL and Big Data stores. It will support the resolution of data and resources based on high-level queries and metadata.

Big Data Europe Integrator (BDI) platform [Auer *et al.* 2017]. The BDI platform has been developed as part of the Big Data Europe project and is powering several use-case pilots responding to societal challenges, ranging from emergency response during radiological events to transport. The BDI platform is a customised, cloud-ready and modular integrator platform, bringing together commercial and research, production-ready components for big-data analytics. It contains tools to help with composing and configuring BDI instances/environments, and monitoring them during deployment, test and production. BDI is based on the Docker platform due to its flexibility, reusability, popularity and compatibility with all major cloud provision services. The BDI platform can provide data management and analytics functionality. It will be extended to form the basis on which DARE components will be deployed. Such extensions will look at the progress of related prototyping work taking place on the e-infrastructure level, such as the EUDAT GEF, and the support of connectivity to HPC resources. DARE will evaluate whether these efforts can be efficiently reused and integrated to meet the DARE's challenging requirements of composability, data-transfer and monitoring of its hybrid computational platform, at scale.

6.2 Computational-seismology components

A main component for the computational seismology use case is the eScience scientific portal developed during the European FP7 project VERCE and refined and rolled out in the European FP7 project EPOS-IP. This platform has been developed to allow both expert and less expert users to quickly perform simulations of the seismic wavefield generated by an earthquake and to easily manage post-processing and analyses of the output data. The portal functionalities are carried out through four principal workflows:

1. the Simulation Workflow that allows users to select the simulation region with a corresponding seismic wave speed model (by choosing among an implemented library or uploading bespoke models), select the earthquake to be simulated and the seismic stations, and to finally launch the simulation run;
2. the Download Workflow permits users to query seismological European archives for raw recorded seismograms corresponding to the simulated waveforms;
3. the Processing Workflow allows them to apply typical seismological steps to both observed and simulated traces in order to prepare them for comparison;
4. the Misfit Workflow offers different procedures to calculate the misfit values between recorded and simulated seismograms, fundamental to study, e.g., the model behaviour or to approach waveform inversion; this workflow has been significantly improved and updated in the current release of the portal.

The main software implemented in the VERCE portal for waveform simulations is SPECFEM3D [Peter *et al.* 2011], both its version for local/regional simulations and the one for regional/global

scale. This is a Fortran 95 code tested worldwide and scalable to a huge number of cores and also adapted to exploit GPU resources. Moreover, for the misfit calculation two other software options are already implemented in the portal: the code pyflex[62], a Python port of the Fortran 95 code FLEXWIN (Maggi *et al.* 2009), that selects time windows on the seismograms where it calculates cross-correlation misfit parameters between observed and synthetic traces, and the Python code developed by Kristekova *et al.* [2006, 2009] that calculates time-frequency misfit criteria on full seismic waveforms.

The intensive numerical calculations of the VERCE portal are performed exploiting HPC resources of EGI and/or PRACE computing centres. Recent updates tested the readiness of different cloud providers of the EGI Federation to support the EGI Virtual Organisation (VO) through which the portal is running.

The functioning of the VERCE platform is controlled behind the scenes by another fundamental component, the cross-platform processing framework dispel4py [Filgueira *et al.* 2016a]. This is a Python library specifically designed to describe abstract workflows for distributed data-intensive applications and to allow their execution in a large variety of parallel environments. This component thus represents the base of the VERCE platform workflows, orchestrating the management of input and output data, the connections and relationships between the different workflows, down to the definition of the fundamental pipelines that constitute the single processing steps within the platform. It is its level of abstraction and granularity that guarantees the strong flexibility of the portal allowing for easy customisation of the procedures by the users and for a continuous update of the portal functionalities in order to support the evolving requirements from field researchers. It is clear that dispel4py can cover a key role both for the seismological use case and for the climatological one.

In this framework, Python and especially its package ObsPy[63] are also essential components for computational seismology applications. ObsPy is a widely adopted Python framework for processing seismological data; it provides parsers for common seismological file formats, clients to access data centres and fundamental seismological signal-processing routines which allow the manipulation of seismological time series. The seismology-specific processing elements managed by dispel4py are all written in Python using the ready-to-use functions specifically designed to meet the needs of seismological researchers by ObsPy.

Finally, the VERCE portal also depends on the functioning of external services that allow for gathering the input data used within the portal. For example, the web services of the Federation of Digital Seismographic Networks (FDSN) is an option implemented in the portal to collect the parameters of the earthquakes and stations to be used in the simulations, or in the Download workflow. The observed seismograms for waveform comparisons can be searched and downloaded through ORFEUS/EIDA nodes.

All the above described components are already in place and operational under EPOS Seismology and are considered fundamental for the development of the seismological use case within DARE. In addition, other required components are some of the seismological software packages already used by the seismic community to address the main tasks planned in the EPOS Use case (WP6 – Section 7.1), some example are:

a. pycmt3d[64] is the code that we plan to use for point-like moment tensor source inversions with 3D wave-speed models;

[62] Pyflex, L. Krisher; http://krischer.github.io/pyflex

[63] ObsPy http://doi.org/10.5281/zenodo.165135

[64] pycmt3d https://github.com/wjlei1990/pycmt3d

b. other codes or libraries for seismic source inversions modelled as point sources e.g. pyTDMT[65] and instaseis[66];
c. codes for finite source inversions e.g. Dreger *et al.* [2005];
d. code for shakemap calculation (e.g. the one implemented at INGV[67]);
e. some state-of-the-art library for machine learning analysis, useful for Ensemble Simulation (ES) analyses, such as scikit-learn[68].

6.3 Climate-impact modelling components

We list here the services, software and methods currently operational (within or external to DARE) that the climate-impact modelling community uses and that should be continued, upgraded or introduced into DARE in the next steps. This includes their current operational portals, data services, data-transport services, simulation systems and libraries. This information was compiled by WP7 with assistance from T3.1.

1. *Basic components: NetCDF, OpenDAP, THREDDS*
   - **NetCDF** (Network Common Data Form)[69] is an interface to a library of data access functions for storing and retrieving data in the form of arrays. It is the major, if not only, file format used for storing and exchanging climate modelling simulations. It is self-describing. The names of variables and dimensions should be meaningful and conform to any relevant conventions. Dimensions should have corresponding coordinate variables where sensible.
     Attributes play a vital role in providing ancillary information. It is important to use all the relevant standard attributes using the relevant conventions. A number of groups have defined their own additional conventions and styles for NetCDF data. Descriptions of these conventions, as well as examples incorporating them can be accessed from the NetCDF Conventions site[70].

     A large number of tools can be used to manipulate NetCDF files, most of them are open-source. Standard interfaces to access NetCDF files exist.

   - **OPeNDAP**: is an acronym for "Open-source Project for a Network Data Access Protocol," an endeavour focused on enhancing the retrieval of remote, structured data through a Web-based architecture and a discipline-neutral Data Access Protocol (DAP). Widely used, especially in climate sciences, the protocol is layered on HTTP, and its current specification is DAP4, though the previous DAP2 version remains widely used.
     It supports different operations, such as sub-setting, and DAP clients can access those files as if they were accessed as local files.
   - **THREDDS**: The THREDDS Data Server (TDS) is a web server that provides metadata and data access for scientific datasets, using a variety of remote data-access protocols. Notably, it provides catalogue services for NetCDF datasets.
2. *ESGF Data Nodes and Computing Nodes and OpenID*
   The Earth System Grid Federation (ESGF) Peer-to-Peer (P2P) enterprise system is a collaboration that develops, deploys and maintains software infrastructure for the management, dissemination, and analysis of model output and observational data. The ESGF maintains a global system of federated data centres that allow access to the largest

[65] pyTDMT http://webservices.rm.ingv.it/pyTDMT/

[66] insstaseis http://instaseis.net

[67] INGV shakemap http://shakemap.rm.ingv.it

[68] scikit-learn http://scikit-learn.org

[69] NetCDF https://www.unidata.ucar.edu/software/netcdf/

[70] NetCDF conventions http://www.unidata.ucar.edu/netcdf/conventions.html

archive of climate data world-wide. Data is accessed through any of the ESGF Data Nodes. In the near future, some pre-processing will be possible near the data storage using ESGF Computing Nodes that will implement the Compute Working Team (CWT) API.

The ESGF uses OpenID as its authentication mechanism.

3. *climate4impact (C4I) and pyWPS, OpenID delegation, processing delegation*
   - The climate4impact (C4I)[71] Platform provides easier access to data stored in ESGF and any OpenDAP/THREDDS public server. C4I provides search, visualisation, download, and processing services. Visualisation is using OGC WMS, processing services are using pyWPS (OGC WPS) standards. Recently interfaces between C4I and the future computing nodes have been developed as a prototype. Also, a prototype of using the EUDAT GEF to perform processing delegation onto the EGI FedCloud has been developed. Authentication/authorisation to ESGF is done using certificate delegation.
4. *CERFACS icclim, nco, NCAR ncl*
   - CERFACS icclim: open-source software to calculate climate indices and indicators, including simple statistics
   - nco: tool to perform operations and calculations on NetCDF datafiles
   - NCAR ncl: scripting language to perform calculations and operations on NetCDF data
5. *EUDAT GEF, B2SHARE, B2STAGE, B2DROP, B2HANDLE, B2ACCESS*
   - GEF: docker-based workflow system for remote execution
   - B2SHARE is a user-friendly, reliable and trustworthy way for researchers, scientific communities and citizen scientists to store and share small-scale research data from diverse contexts.
   - B2STAGE is a reliable, efficient, light-weight and easy-to-use service to transfer research data sets between EUDAT storage resources and high-performance computing (HPC) workspaces.
   - B2DROP is a secure and trusted data exchange service for researchers and scientists to keep their research data synchronised and up-to-date and to exchange with other researchers.
   - B2HANDLE is the EUDAT PID service
   - B2ACCESS is the EUDAT authentication service
   - B2NOTE is an annotation service
6. *EGI FedCloud* The EGI FedCloud provides computing and storage capabilities using Virtual Machines.

6.4 Summary and Architectural implications

The diversity of technologies currently available, as listed in Section 6.1, and as required by the application domains, see Sections 6.2 and 6.3, indicate the complexity and diversity the DARE architecture has to cope with. It illustrates well the realistic complexity of the digital environment from which resources are drawn. It also exemplifies the complexity inherent in each application domain's data-driven science systems and services. The complexity will grow as research and communities mature. Most changes are conducted by autonomous organisations. Therefore, sustained engagement in and support for data-driven R&D has to adopt strategies to cope with the complexity and evolution. One, exemplified the global consortia supporting climate research, is to form large federations to muster sufficient skills, effort and resources. This requires political leadership, governance, coordination and continuing commitment. The other, they are complementary, is exemplified by DARE. Use knowledge organisation to structure and maintain the body of information that accrues as the complexity is addressed in many separate contexts. Then

[71] C4I https://climate4impact.eu/

use this organised information to minimise the costs of accommodating change and to automate the propagation of those changes to the places where they are needed.

7 Requirements

This Section presents an analysis and distillation from the user stories collected by WP3 Task 3.1 and via embedded task forces and other investigations. It draws on analyses in internal deliverable ID3.1-M6 [Spinuso & Filgueira 2018]. It includes refinements from two face-to-face meetings in November 2018.

The seismologist present three overlapping data-driven computational challenges that would significantly enhance their capabilities. These are analysed to reveal detailed requirements and identify overlap. This leads to a commitment to prioritise work on the rapid assessment of ground motion as a result of an earthquake – information needed for those advising hazard-response teams.

The climate-impact modellers anticipate the scale of simulation results from the next phase of climate modelling. As a result, they recognise that the current *modus operandi* of their community of researchers will become impractical. Instead the researchers will need to be well supported running their analyses close to the simulation results – the output from analyses is invariably very much smaller to make the evidence assimilable by humans. They identify and analyse a representative researcher-led analysis using current smaller data sets and commit to pioneering solutions that will continue to work well as the data grows.

7.1 Requirements for computational seismologists

Based on the work in the first few months of the project and following Deliverable D6.1 [Rietbrock *et al.* 2018], the main test cases that compose the general EPOS Use Case have been identified delineating the underlying workflows and the principal requirements. Deeper analyses and improved understanding will be developed through co-design and co-development, causing the requirements to evolve contemporaneously.

In the framework of EPOS Use Case within DARE, seismologists are primarily interested in:

1. designing and implementing methods for Rapid Assessment (**RA**) of strong ground motion after large earthquakes also in the context of emergency response;
2. the rapid characterisation of Seismic Sources (**SS**) to evaluate the impact on an earthquake's wave propagation and to support decision-makers in localised hazard assessments;
3. on-demand Ensemble Simulations (**ES**) which are required for statistical analyses of the ground motion parameters and their uncertainties exploring the variability of the input models.

In view of these tasks, there is a strong demand for robust provenance-driven tools to organise, explore and reuse the results, with flexible management of metadata for detailed and *ad hoc* validation of methods. To address these requirements, DARE should provide a holistic system that will facilitate comparative studies and will complement the rapid response to societal demands with trustworthy evidence and advice. Moreover, we can benefit from the strong experience matured during the development of the VERCE portal [Atkinson *et al*., 2015] in the framework of the VERCE and EPOS-IP projects.

The **RA** of strong ground motion is considered the primary objective since most of the needed components and tools are implemented on the VERCE platform and therefore the focus can be put on integration and extension of the capability of the newly deployed DARE platform. The aim of this first test case is to quickly analyse earthquakes and produce rapid on-demand estimates of parameters such as the peak values of velocity or acceleration of the ground motion or the intensity of ground shaking. Output products such as waveform propagation snapshots and especially maps of ground-motion parameters are fundamental for a visual representation of the earthquake. They are also useful in the framework of emergency response and can be compared with maps based on recorded ground motion data, so-called Shakemaps [e.g. Michelini *et al*., 2008].

The specific steps and requirements in this case include:

1. Selecting the models to describe the region where the seismic wavefield is simulated geometrically and physically. This can be achieved by choosing a model from a library of available models or by uploading customised models. This is already implemented in the VERCE platform and EPOS-IP will extend the library of models.
2. Selecting the seismic source parameters that describe the earthquake to be simulated. This can be achieved by collecting information from national and international archives (e.g. GCMT, TDMT by INGV) or uploading customised models. Both point-like seismic sources and extended fault descriptions are possible. This is already available in the VERCE platform except for finite seismic sources; arrangements for their use still needs to be implemented.
3. Managing the numerical simulation software. In general, the seismological use case can use the code SPECFEM3D, already incorporated in the VERCE platform, both for global and local/regional seismic waveform simulations.
4. Accessing the suitable computing resources on-demand, to produce the simulated output data as quickly as possible. These data are both numerous small (tens of KB) files in ASCII format (eventually converted into mseed format) for the seismograms and a smaller number of bigger (MB to GB) files in binary format for the visualisation outputs. Again, this is already operational in the VERCE platform, but actual on-demand requests have still to be incorporated.
5. Rapid transfer of the input/output data between heterogeneous co-opted execution environments and storage systems, including Cloud resources.
6. Organisation and exploration of the runs and results based on their metadata and provenance information, for easy discovery and combination of the outputs from simulations with different inputs. This includes management tools that allow users to summarise the ground motion features, combining outputs from multiple runs. Whereas, in the VERCE portal so far, only one-to-one comparisons between synthetics and data are supported.
7. Gathering of corresponding observed data from national and international archives (e.g. EIDA, INGV Shakemaps); these data can be seismograms in mseed format (as already managed by the VERCE platform) but now also binary files containing information on the strong ground motion parameters extrapolated from the analysis of observed data like Shakemaps [Michelini *et al.*, 2008].
8. Managing the tools requested in Section 6.2 for the comparison and combination of synthetic outputs on earthquake strong ground motion and the corresponding information based on observed data. The flux of input and output information exchanged during these procedures is usually encoded in ASCII, XML or JSON/GeoJSON files.
9. Handling the storage requirements. For a complete RA experiment, the volume of data to be stored can reach a maximum of tens of terabytes per user in the production context.
10. Handling the computing demand. For a complete RA experiment, the computational resource requirements can reach a maximum of tens of millions of CPU hours per user in the production context.

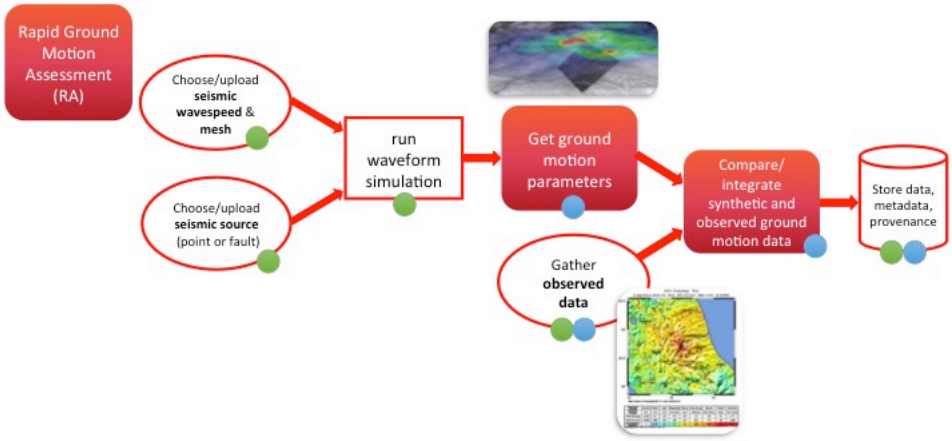

Figure 6: Steps of the workflow for the Rapid Assessment test case. The coloured dots associated with each element indicate the steps that are in common with the other proposed test cases: green is for the Seismic Source (SS) analysis test cases, blue for the Ensemble Simulation (ES) test case.

An earthquake is usually modelled as a slip on a fault, that is a discontinuity in displacement across an internal surface in a medium. This earthquake source can be mathematically described by a system of equivalent forces represented by the moment tensor. The components of the moment tensor describe the mechanism of the rupture along the earthquake fault and the displacement along this fault, and are therefore fundamental parameters to describe the earthquake source solution. (For more detailed and technical explanations see [Aki & Richards 1980]). The **SS** analysis aims at characterising the parameters of earthquake sources including the earthquake location, magnitude and rupture mechanism represented by the moment tensor. While, in case of modelling the earthquake as an extended fault, the parameters include the values and direction of the displacement that occurred on this fault. In SS analysis the simulated synthetic waveforms for an initial model of the seismic source are compared with the observed data in order to invert for improved values of the source parameters (minimising this misfit) and to estimate the associated uncertainties. These parameters and uncertainties characterise the seismic sources and are fundamental for further hazard assessment analyses.

The RA and SS cases have in common the steps and requirements described at points 1-7 above. Then the impact caused by the seismic source on the ground motion parameters is analysed. In the framework of seismic source analysis, we plan to distinguish four different test cases depending on the model chosen for the earthquake source and for the wave-speed structure:

1. study of point-like seismic sources using 1D wave-speed models;
2. study of point-like seismic sources using 3D wave-speed models;
3. study of seismic sources modelled as a slip on a fault with finite dimensions using 1D wave-speed models;
4. study of seismic sources modelled as a slip on a fault with finite dimensions using 3D wave-speed models.

With respect to RA, these cases involve additional simulations or access to pre-calculated basis-function libraries required by the inversion procedures implemented in the software packages of Section 6.2. For example, point source inversions in 3D require 6 to 9 additional simulations for each earthquake obtained by perturbing the source parameters one-at-a-time [Liu *et al*. 2004]. The other three cases require calculation of seismic wavefields for unitary input sources, i.e. pre-calculated so-called Green's functions, forming the basis functions that are combined by the inversion procedures to get updated source solutions (e.g. [Dreger *et al.* 2005]). In this sense, as described in point 6 of RA, multiple simulations for the same earthquake should be easily linked based on metadata and provenance, in order to combine the input for the inversions. Among these four test cases for which SS is articulated, we consider the study of point-like sources with 3D wave-speed models a priority, especially because part of the workflow has already been developed in VERCE and we agreed on a main tool for inversion with a straightforward implementation (Section 6.2).

The format of input/output data for SS task is, as in the RA case, ASCII/XML/JSON for the summary files of the analysis software and ASCII/seed files for the seismograms. Analyses for the SS tasks also involve the code FLEXWIN/pyflex, described in Section 6.2, for the selection of waveform time windows suitable for inversion procedures. Their use is already managed by the VERCE platform. Finally, the storage and computational requirements described at points 9 and 10 for task RA above are also valid for the SS task.

The **ES** task has the scope of statistically characterising the ground motion parameters and their uncertainties, analysing ensembles of models constructed by the variability of the input parameters. Thus, it shares the requirements at points 1-7 described for RA, but in this case, we are more focused on exploring the variability of the source model parameters. Earthquake sources can be modelled as points or extended finite faults, and their impact on ground motion assessment, highlighting the strong connection of this test case with the other two proposed test cases. At step 2 of the RA test case, rather than requiring the selection of a single source model, it should be allowed to perform a grid search on ranges of values of the source parameters, implying that for each value in the range a new simulation should be carried out, while the other input parameters stay fixed. Thus, a major characteristic of this task is that a very large number of simulations (hundreds to thousands) or a library of pre-calculated basis functions (e.g. Green's functions) will be required. Their outputs should be managed automatically, also implementing tools to summarise them for comparisons with observations (requirement 7) and to quickly and easily link to these results in order to use them as input of the software for ensemble and uncertainty analyses described in Section 6.2 (requirement 8). Other specific requirements are:

1. Handling a storage demand that can reach a maximum of hundreds of TB per user for a complete ES experiment in the production phase.
2. Handling a computing demand that ranges from tens to hundreds of millions of CPU hours per user for a complete ES experiment in the production phase.

Analysing the interconnections and overlapping steps between the test cases described above, we have identified specific common requirements that will be the basis for the work of the architectural task force (WP2) and for the construction of the user stories (task T3.1 of WP3).

1. All the test cases require the combination of numerous outputs from multiple workflows, thus all the outputs should be described by their detailed metadata and provenance to allow their exploration, reuse and combination for complex analyses.
2. Since the proposed use cases have been designed with many overlapping steps, all the workflows that constitute their structure should be built with a high modularity in order to be as general as possible and to be applicable to different processing. This increases the platform flexibility and the possibility of adapting it to evolving approaches.
3. The three test cases also require that all the involved execution environments (HPC, Clouds and institutional resources) should be quickly and easily linked to each other in order to reduce and optimise the time required for analyses and transfers of data.
4. Another requirement is the possibility of handling different **data formats** for input and output products. Although this is again a general requirement, the three use cases have specific formats to be handled and details are reported in Figure 7. The figure lists the formats required for the involved input and output files for each test case, and the specific products managed by every case are highlighted with colours. As the main goal of the EPOS Use Case is studying the variation of ground motion parameters caused by earthquake source variability, it is evident that overlaps between the requirements of the different test cases exist and especially that ES in general combines the needs of RA and SS test cases.

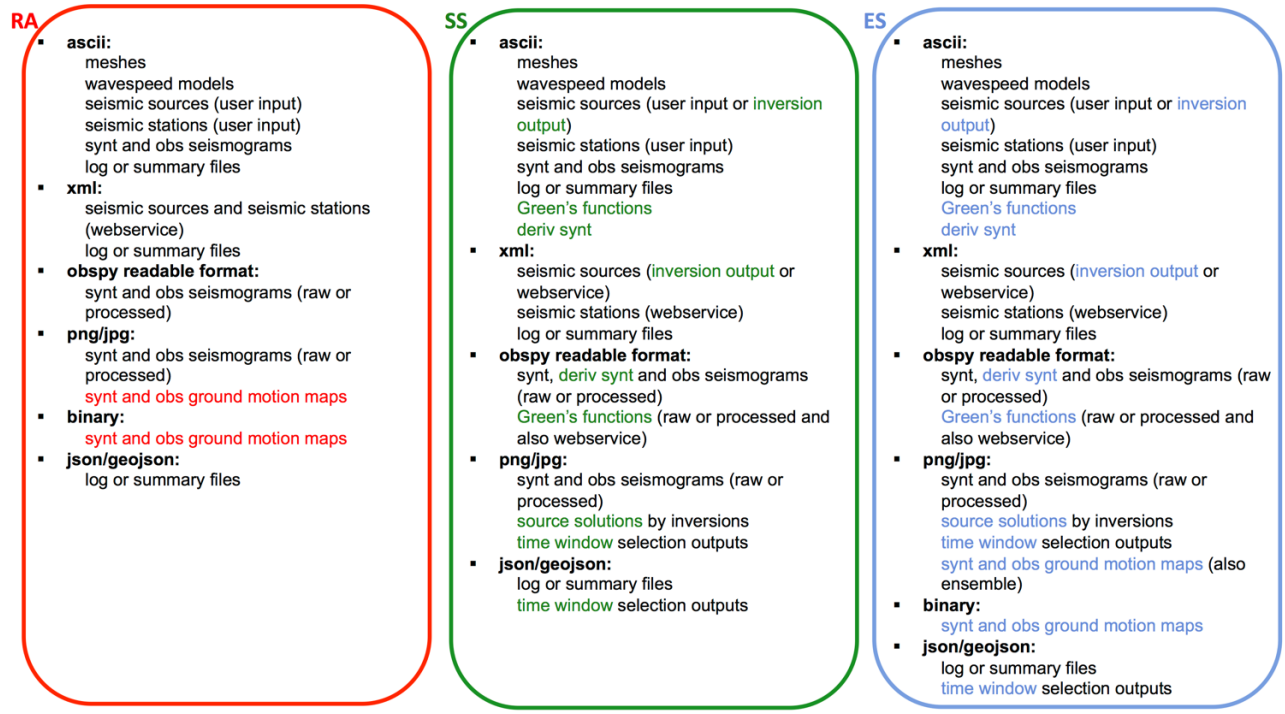

Figure 7: Data formats required for the input and output files of the three main test cases of the EPOS Use Case. Specific products managed by every case are highlighted with corresponding colours.

5. To gather the input products of the test cases the exploitation of multiple data sources is required. The specific requirements vary slightly between the three use cases, further details are provided in Figure 8. In the RA case, it is required to access public repositories of data for the source solutions, stations, waveforms and ground motion parameters. However, users should also be allowed to upload their own inputs for customised experiments. The same is valid for the SS test case that moreover requires to access public or private repositories of Green's functions and source geometry models. The ES test case again combines all the previous requirements.

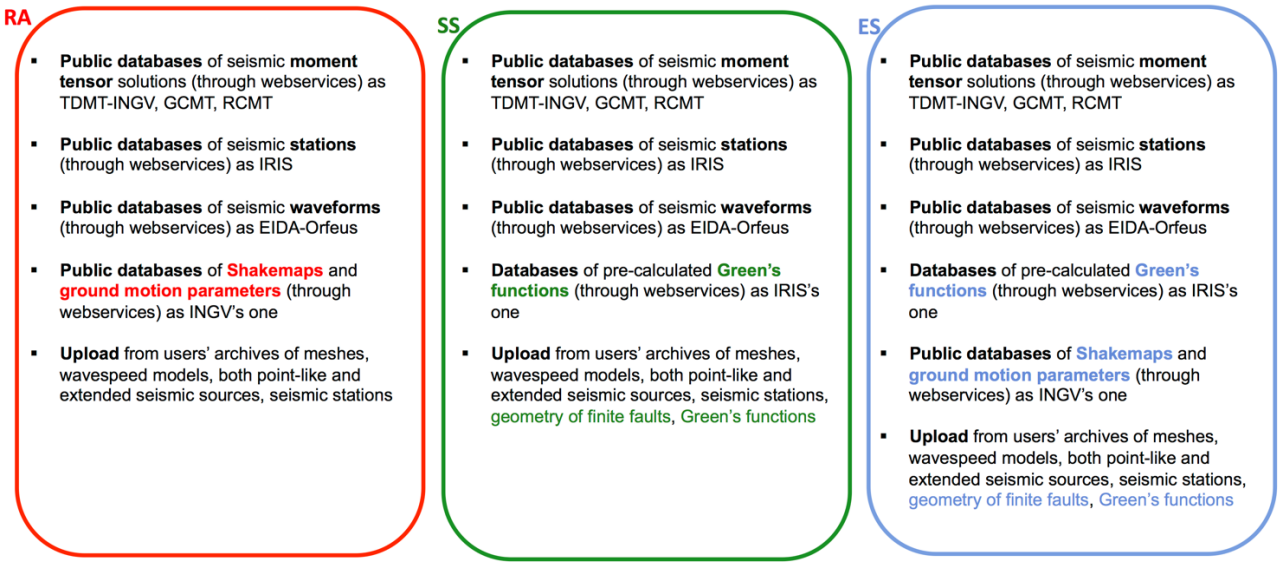

Figure 8: Data sources to be accessed in order to gather the input files of the three main test cases of the EPOS Use Case. Specific products managed by every case are highlighted with corresponding colours.

6. Following the strong requirement of carefully describing all the inputs and outputs of the test cases with detailed metadata and provenance in order to make them searchable and reusable, in Figure 9 we report a list of the main metadata that should be attributed and stored for the products of the use cases. Thus, RA specifically involves metadata for the ground motion maps, while SS involves metadata to describe the Green's functions and the inverted source models. As before, ES combines all these requirements.

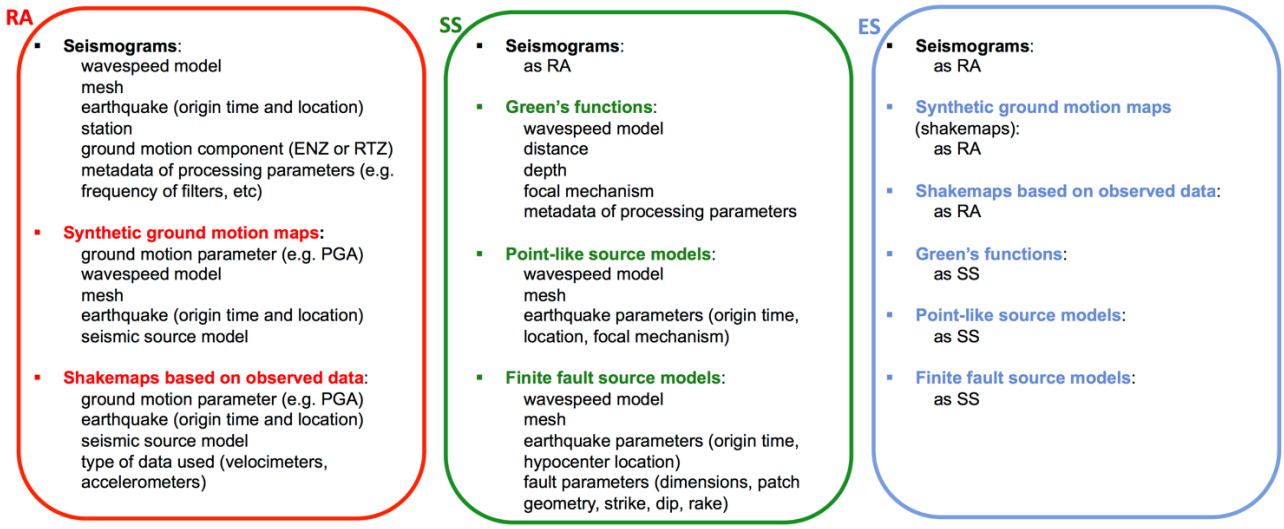

Figure 9: Metadata to be attributed and stored to the input and output files of the three main test cases of the EPOS Use Case. Colours highlight the specific products to be handled by every test case.

7. The last important requirement is the storage and computing demand. Figure 10 gives an upper limit of the resources that would be required for complete experiments in the production stage, i.e. when the DARE platform will be fully deployed and usable. However, there are possibilities to reduce these demands, for example by using pre-calculated basis functions that can be recombined instead of performing new simulations every time. Moreover, in the development phase both computing and storage requirements are drastically lower (~100s CPUhs and a few TBs per user).

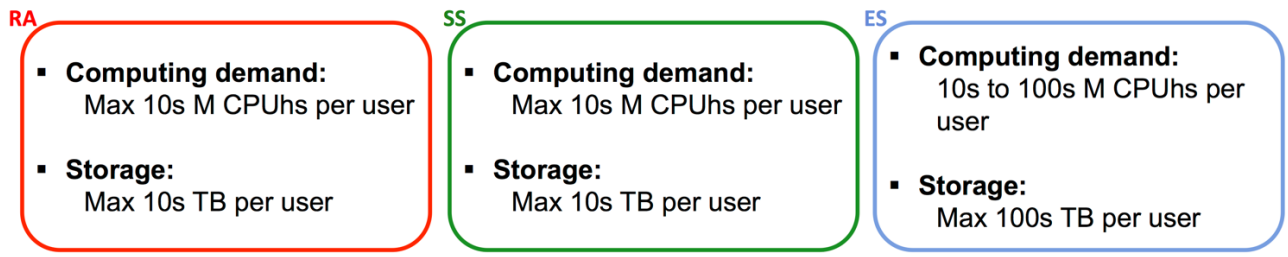

Figure 10: Computational and storage requirements for complete experiments in the context of the three main test cases of the EPOS Use Case, considering a production phase.

The requirements and workflows underlying the main test cases of the EPOS Use Case are also detailed in the tables[72] compiled to support WP3 in the definition of the user stories and of the

[72] https://docs.google.com/document/d/1Q9IbW1SZCskuKywX8Tz7D-PqZip7ykg4Vvn21Cq4FmE/edit

common requirements between the principal DARE Use Cases from WP6 and WP7. In general, we expect that the described requirements and workflows will be defined with more details during the next steps of the project.

7.2 Requirements for climate-impact modellers

The requirements of the climate-impact modelling community have been developed and analysed during the first nine months of the DARE project and are reported in the WP7 deliverable D7.1 [Pagé & Spinuso 2018].

The climate impact Use Case is quite generic, and it is built in such a way that it exposes the requirements needed for the most common climate impact Use Cases. The total data volume for this Use Case may seem small, but this is intentional. The prototype will be developed with smaller data volumes, knowing that for other Use Cases the input data volume can be significantly larger.

The goal of those Use Cases is to provide on-demand data analytics and processing for end users, as the front-end platform climate4impact.eu (C4I) needs access to data processing capabilities closer to the data storage, in terms of network bandwidth and computing power.

Before diving into the requirements, it is mandatory to have a description of this Use Case, and some technical information. The Scientific Workflow generates a multi-model, multi-scenario, multi-ensemble-member time series spatial average (over Western Europe) of surface temperature using CMIP5 data as input. The steps of the workflow are as follows:

1. For one given GHG scenario (RCP8.5), for each climate model, for each ensemble member, for the time period 1950-2100
    a. Extract the surface temperature fields over Western Europe (continent grid points only).
2. For each temperature field of each experiment, perform a spatial average of the land grid points over Western Europe. This step will provide one-time series per experiment.
3. Perform an average and calculate the standard deviation of all the time series.
4. Provide the results to the C4I platform as NetCDF files using the CF-convention.
5. The C4I platform can display the results as a plot of individual time series, overlaying the time series average. The standard deviation is used to draw a grey-shaded area.

The technical specifications are as follows:

- Input data files are stored on the ESGF data nodes, in NetCDF file format, using the CMOR and CF-conventions for file naming
    - Each file holds in general 5 to 10 years' worth of data
    - Daily datafiles will be used
    - Typical size is on the order of 1 GB per 5-year file
    - ESGF Authentication/Authorisation is done through C4I, then delegation must happen
    - Data Search will be done using the C4I Search API

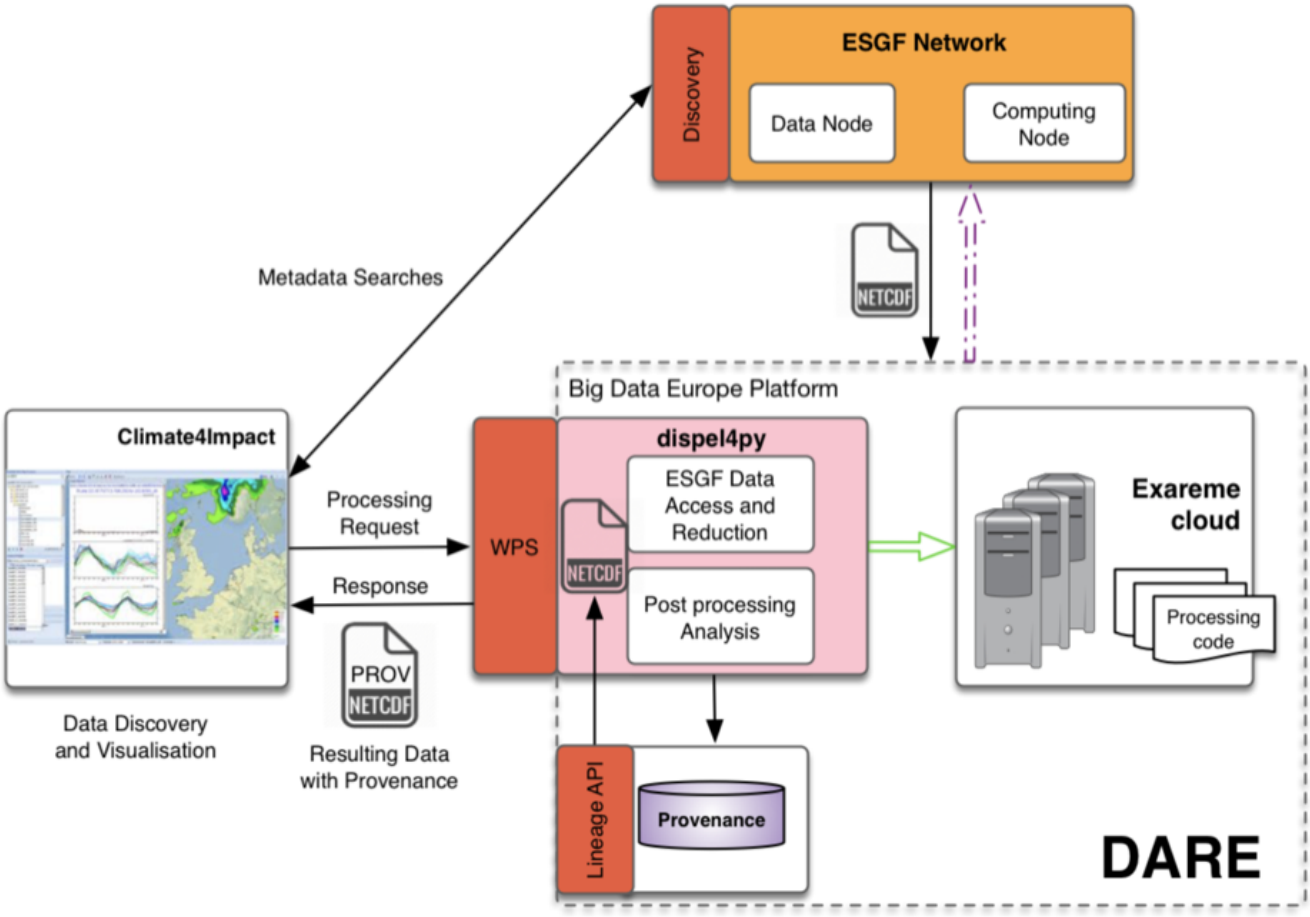

Figure 11: Overview of climate-impact modelling system.

7.2.1 Initial requirements

- dispel4py
  - Interfaces
    - Access ESGF Data Nodes
    - Access ESGF Computing Nodes (CWT API)
    - WPS: To exchange workflow information with C4I
  - Processing
    - Output: store to B2SHARE? and/or B2DROP? Results will need to be accessed by C4I
    - icclim as a post-processing tool
    - Delegate part of the calculations to the Computing Nodes when possible
    - Implementation of all analysis functions provided by the processing tool
  - Provenance
    - Specify a provenance model, to include relevant information about the processing workflow
- Climate4impact (C4I) Platform
  - Interfaces
    - Workflow system
    - Encapsulate with EUDAT GEF?
    - Receive results from DARE processing: WPS interface
  - UI
    - Enhance Access to Processing
    - UI Wizard

7.2.2 Interoperability / Open questions

- Identify which and where EUDAT Services will be used

    - GEF: Execution in a Docker Environment
        - Integration with the dispel4py architecture
    - B2SHARE: Storage of data output?
- EGI FedCloud
    - GEF can execute code on this platform
    - Use Exareme as an intermediate to run on EGI FedCloud?
    - EOSC & C3S: coordination needed
- AAAI: DARE EGI Check-in Service
    - B2ACCESS (EUDAT)
    - OpenID (ESGF/C4I)
    - ESGF CWT API Key

7.3 Requirements to support current and future provenance uses

There are many uses of provenance which DARE user-communities may require. We list a few examples:

1. To aid understanding of a method and how its results are derived, often through several stages, from the original inputs and controlling parameters. A user may do this to check that their own or a colleague or rival have used the method appropriately. A researcher new to the method, e.g., on the intellectual ramp provided by copying a Jupyter notebook and editing it (see Section 5), may review the provenance to understand how choices they make influence the derivation process.
2. To verify that things that ought to be done have been done. By visualising a derivation graph a user can quickly check that all of the intended inputs have been done, that all the user-controlled steps have been run and that the preservation and curation of significant evidence has been performed. Omissions spotted can be combined with direct activation of the necessary steps.
3. Recognition of contributions can be unequivocally established from the provenance records and properly acknowledged.
4. Defence against accusations that duties have not been done can be refuted from the provenance record.
5. Detection of issues technical and organisational is possible by analysing the accumulated provenance records.
6. Optimisation is supported in two ways. Many users and workflows repeat actions that have already been done. Examination of the provenance records can prevent redundant actions after verifying that the inputs, controls and context have not changed.
7. The second form of optimisation takes advantage of the fact that most research and community activities are very repetitive; e.g., the same method is used to derive the same reports for each event or for each day's observations. Mining provenance data from prior runs will provide accurate data to target the best platforms, with the deployments, co-locations and parallelisation that work best for those platforms. The completeness of the provenance record enables these optimisations to be automatically steered from reliable data as the target's availability and loads vary.

More examples, such as detection of intrusion or inappropriate behaviour will eventually become relevant. These potential uses lead to the provision of the framework for collecting and using provenance data. However, the user communities, through agreed decision making, need to decide precisely how to balance the costs of provenance collection[73], preservation, maintenance and access

[73] The costs of collecting, preserving and using provenance data include data movement and storage that may cause delays, the cost of the preservation system and typically small computational costs. These may impact response times or throughput. They will also have monetary and environmental-impact costs. In the context of DARE, as it addresses large-scale data and computation, these are proportionally minor. Potential savings from avoiding redundant computation and optimisation are expected to outweigh them. But this needs to be measured and shown.

against the benefits for them. As always, providing early benefits, a la Meyrs *et al.* [2015] encourages adoption. We review below the options in the DARE context and the inducements.

The role of provenance in DARE will help achieve a rapid and shared understanding of the methods developed and executed through the platform, through the *live* validation of user-controlled experiments and the *a posteriori* reproducibility of the results. Different phases of the evaluation of applications will require different provenance details and precision. Users should be assisted in extending the fundamental provenance statements with domain and application specific information through contextualisation controls. They may have an effect on concrete and catalogued data, as well as on volatile intermediate data. Information about the latter is key to the diagnosis of logical flaws and to communicate the effect of intermediate transformations on the route from input data to the generated results. The level of completeness of the provenance should be chosen to be both useful and manageable. Often it is shaped by common patterns. Patterns are useful from a high-level and end-user perspective, as well at system-level. They may be used to control and validate constraints (Why Not Provenance), which may affect the validity of exploitation of the platform within a target domain. Patterns could lead to an efficiency comparison between the behaviour of similar operators, which may lead for instance to the rapid identification and application of common optimisation strategies.

Patterns are reusable across operations and contexts, which suggests a combinable approach which allows users and developers to inject into provenance and lineage traces concepts and terms that are relevant to the current stage and progress of their investigation and implementation. Such phases are managed by the C4. References in the lineage entities to the C4 concepts will coexist with system level details, to support the broader coverage envisaged by P4.

Once provenance is contextualised and recorded, DARE should expose high-level methods for its interrogation. These should support typical end-user exploitation use cases such as monitoring, results' management for the discovery of experiment and data, and to navigate through the dependencies that led to a specific output. Dependencies could identify data and software, while discovery services should provide a user with hints and recommendations about suitable search metadata terms and value-ranges, where applicable. Here context awareness will be fundamental to identifying the terms and conceptual space in which a query should be framed and the results summarised and visualised.

Data derivations may reveal particular characteristics of the data dependencies, for instance, to distinguish among those data resources that have been acquired externally to DARE, exchanged between operators or reused across sessions, or computed and preserved as reusable intermediate results within the operator's state. The characterisation of the derivations will allow us to represent complex and heterogeneous data-integration activities, fostering the diagnostics of single operators, and the application of optimisation and failure recovery mechanisms. The latter could be realised with the automatic materialisation and reuse of the most recent snapshot of a process' state, whose information could be accessed from the provenance recordings, with the effect of reducing the need of expensive re-computations.

To summarise, a provenance system for DARE addressing the use cases just mentioned and those ones that will be refined during the project progress should consider the following aspects.

1. *Efficiency of provenance collection.* There are situations where a potential over-head may discourage researchers from pursuing provenance recording actions.
2. *Handling of provenance for streaming data.* Stream-based processing is a computational paradigm that is suitable for a variety of data-intensive use cases. DARE makes use of the dispel4py processing library which will accommodate mapping to various systems including streaming engines. Most of the data flowing into and transformed by such systems are consumed at a very high rate and

intermediate stages are often volatile[74], thus, making the injection of lineage recording procedures complex and expensive. Given the high demand for I/O in such digital ecosystems, even storing partial lineage data could be unmanageable and even uninformative, limiting the chances of its effective exploitation. Efficient approaches to collect and structure provenance for data-intensive scenarios are needed, in favour of a concise representation for its production, storage and access.

3. *Control by researchers of the provenance collected.* How can we involve the researchers in the archival process of the lineage from its early stages, returning immediate benefits at an acceptable overhead? This requires a guarantee of a certain level of freedom in letting users indicate the data-intensive engine which best fits their needs. We need to combine low-level details within high-level domain and application specific concepts, obtaining layered interpretations of the provenance data.

Automated use of provenance data. Considering provenance as actionable data may allow the automation of metadata-driven operations, such as transfers of data across infrastructures at runtime, or the allocation of dedicated resources for the post processing of intermediate results. For instance, intermediate raster graphical content could be immediately rendered by a dedicated system, which could be different from the architecture executing the computation. In these circumstances, the target system may consider contextual information associated with the domain of the received data and stored as part of the lineage, to enrich the presentation with complementary information.

8 The functions of the DARE platform

Here we present in abstract terms the operations that each part of the platform delivers. These are then drawn together as to operations delivered by the DARE platform API [Klampanos *et al.* 2018]. Section 8.4 then explains the steps the three technology pillars must take to ensure that they remain mutually consistent.

8.1 The functions of Workflows-as-a-Service (WaaS)

The role of a WaaS was introduced in Section 4.1. Here we further analyse the functionality and interrelationships required of a WaaS in an architecture capable of achieving DARE's goals.

Initially, the predominant form of 'workflow' WaaS should handle is a Python script. The **Configuration** step described above will establish the right context for all the dynamic binding that occurs during its execution. The mirroring of Concepts in the C4 with Python Classes (see Section 8.3.2) will enable such Python programs to directly use the conceptual space and populations assembled by C4 and to annotate, update, add and use those populations. This should mean that application experts and research developers can use their familiar Python tools without having to look beyond the DARE abstractions.

Running such a Python script that has used the *dispel4py* library will generate a dispel4py workflow graph that will then be subject to all of the steps introduced above [Filgueira *et al.* 2016]. The dispel4py workflow system, *d4p*, is tuned for fine-grained parallelism (handling streams of data units flowing between processing elements (**PE**s) and avoiding unnecessary data transport and storage-level boundary transition costs). However, it can emulate task-oriented workflows by carrying descriptions of tasks along streams, and having PEs run the tasks they receive.

For the moment, we assume Python and dispel4py suffice. The '*simple*' mapping of *d4p* onto a single Python interpreter will be needed for software development and for primitive action implementations. There will be one or more Deployed Configurations set up to support such *simple* uses. These will also run any other Python scripts.

[74] In DARE because we use data-streaming to avoid I/O overheads the intermediates may never be on a disk or in a persistent store. Even when the intermediates are passed via backing-storage or shared storage systems, their number and volume may be so large that they may be deleted automatically to release resources.

The *d4p* system is already integrated with P4, but this may need revision now that C4 is available. At present there are potential failure modes in d4p and some of these could be prevented, whilst others could be caught and recovery attempted. Incremental deployment of the dispel4py graphs would save wastage and clean up when failures occur and would provide additional information for optimisers.

"Workflow-as-a-Service" (**WaaS**) characterises complete and convenient support for workflows, whereby processes can be executed and managed through a unified interface or using a simple API. In the application of this idea to DARE, we envisage that the DARE platform's API (and hence tools using it) has an option for explicitly submitting workflows and requesting that they are run, with the usual ability to monitor progress through a returning stream of *active-provenance* records, as well as data outputs. Many uses of WaaS will be more indirect.

8.1.1 Action Enactment

Actions (see Section 8.3.2**Error! Reference source not found.**) stored and accessible through the c atalogue (**C4**) part of the API will often have workflows that implement them. This applies to *user-defined* actions and to *primitive* actions that are supplied as part of DARE's common-core foundation.

Alongside parameters and requirements set by the user that intends to enact the action, an analysis of the action's description in C4 will lead WaaS to identify defaults and inherent or obtained (from the analysis of previous runs available through P4) requirements for proceeding to the enactment (found either as part of the **Action** or in the **Concepts**[75] describing inputs and outputs(see Section 8.3.2**Error! Reference source not found.**).

Configuration and deployment recipes and target selection strategies will be stored in the description of an optimised workflow. The WaaS must use information from the C4 to plan and perform enactment, except in the case of **primitive Actions**[76], where it will be hard-coded – initially, all Actions may be treated in this way. When WaaS develops or revises enactment plans, it will update this information. The output from this phase will be a concrete workflow, except that in most cases it is not yet assigned to target computational and storage resources.

8.1.2 Computational Target Selection

Many **primitive** actions will be performed locally and directly, to avoid latency and coordination overheads, since they are handling small items of data. Other actions will demand significant resources because of the volume of data to be handled and the nature of the computation. Where this is input parameter or monitoring-mode dependent, data from the description will allow the characteristics of the enactment target to be computed[77]. The WaaS *searches* its available services considering their properties and data locations. If there is a suitable one it orchestrates the *assignment* of the work to the chosen target; this may use a queuing mechanism or a pilot that pulls in work. If there is more than one feasible target, it uses its optimisation system to *select* a target.

8.1.3 Computational Target Building

In the case that the target selection process discovers no suitable target, WaaS should *build* one on the available computation, storage, data and transport **Services**. It finds or builds from **Configuration** scripts, the required Docker containers. If it builds them it reports this to P4 and puts a **DockerContainer** entry (description and location(s) where it is stored) in C4. It then deploys

[75] Emboldened identifiers are ones that will be recognised and treated specially by the DARE core. Those commencing with an uppercase letter are concept names that help with the structuring of information in the C4.

[76] Primitive actions are those built into the framework that enables tailored versions of the DARE platform to be produced for different application communities; these will be marked **authoritative**. Additional small actions that are commonly required may be added to this list as an optimisation or to accelerate tailoring – if they are not **authoritative** WaaS should check they have not been overridden. The implementers of each version of WaaS should ensure that their system and the **authoritative** labels are consistent.

[77] In early versions of the DARE platform this will be statically defined.

the required configuration of containers on available services and makes this configuration available, describing it and recording it as a **Service** in C4. It reports this **Deployment** to P4, so that P4 has records of the underlying support for the configuration. The assignment to the new target then proceeds as above.

8.1.4 Data marshalling

If necessary, WaaS *marshals* any required data directly to the appropriate parts of the deployment. It then initiates the action, *notifying* P4 and other interested listeners, e.g., a user's client, that it has done so. Unless the action is primitive, it will also construct a **Run** (see Section 8.3.2**Error! Reference source not found.**) instance and record that in C4 with a link to the start of the provenance trace of that run in P4.

8.1.5 Failure Monitoring and Mitigation

As DARE is pushing scale limits, it is inevitable that there will be failures at various points of the enactment process. Thus, it is essential that all workflows are equipped with failure handling, progress monitoring, diagnostic options and instrumentation as appropriate. These need to be set up by WaaS and it needs to relay this information, at least to P4, for active-provenance applications. P4 uses provenance standards as a *lingua franca*, and if the information is not in the appropriate form WaaS is responsible for its translation.

Furthermore, WaaS has to incorporate functionality for not only detecting and reporting such failures, but also recover from them when possible, e.g., by investigating **Run** instances that haven't terminated as expected. If it can't recover and resume, then it should record the failure in P4 as well as issuing an informative notification that can be picked up by clients.

The failure handling should, as far as possible, be distributed and choreographed, so that local decisions can be made, to avoid coordination bottlenecks, c.f., Swift [Wilde *et al.* 2011, Armstrong *et al.* 2014] and DALiuGE [Wu *et al.* 2017].

8.1.6 Post-enactment Operations

Whether the enactment ends in failure or success, the WaaS must *clean up*; recovering all resources, eliminating and if required obscuring intermediate data, gathering all required outputs and provenance records. These should be transported to the specified destinations. For each output and intermediate that was retained, a **DataItem** (see Section 8.3.2) entry should be inserted into C4, with appropriate metadata, with the item's location and linkable with the provenance records.

When enactments complete successfully their provenance trail should contain sufficient information for the WaaS to analyse the collected provenance, configuration, mapping and context information to improve the parameters guiding optimisation. As many similar workflows are enacted, it should be possible to analyse sets of *Run* entities and their provenance trails, to get more reliable parameters and to discover performance dependencies. This may require bundling similar workflows, possibly using the approach developed by Gorijo [Gorijo *et al.* 2017] for recognising similar subgraphs.

8.1.7 Planning for diversity

The DARE platform already handles native Python scripts, dispel4py scripts and Exareme scripts. The research communities already use a wide range of notations for representing and authoring their scientific workflows – 13 popular scientific workflow systems are analysed in [Atkinson *et al.* 2017] – by no means all of the popular notations. This diversity is in part driven by chance concurrent developments and adoption choices and in part by the different workflow systems (WFS) being tuned for different uses. As a consequence, the established working practices in use in communities depend on a wide variety of WFS. As collaborations develop, or as researchers compare their methods with rival groups, pressure grows to run more WFS on the same platform. As collaborators combine methods based on different WFS, they require to run workflows based on several WFS in combination. Methods for doing this have been developed [Plankensteiner *et al.* 2013, Terstyanszky *et al.* 2014]. Whilst DARE cannot immediately accommodate such diversity

because of other targets, it must plan for this in the descriptions of *SoftwareItems* and the mechanisms for handling them (see Section 8.3.2).

8.2 The functions of Protected Pervasive Persistent Provenance (P4)

In this Section we list the categories of functions that will enable the sound acquisition and exploitation of provenance data. The **pervasive** nature of the information collected includes the platform's technical details, in the context of the user and the application domain. It captures users' behaviours and choices when they customise their conceptual and computational space, mark reproducible progress and share methods and results with peers within or external to DARE. For instance, when interfacing with the services and repositories offered by the EOSC.

8.2.1 Acquisition

Single-Run description. The execution of a method is described by its initial inputs, its components, their interdependencies and their implementation, and the computational resources used. The information should refer to entities stored in the DARE catalogues and enriched with contextual information that is specific to the run. The information collected should be compatible with the conceptual model described by S-PROV [Spinuso 2018].

Lineage ingestion. A single-run generates lineage information in the form of synthetic log documents that should be collected at runtime, thereby allowing for a rapid and efficient ingestion. These will be generated during the execution of a distributed deployment and may contain cross-references to concrete or volatile input and output products. Eventual consistency of these references should be verified, raising alerts otherwise. As above, the information collected should be compatible with the conceptual model described by S-PROV [ibid.].

Registration of Templates, their expansion and validation. This would especially affect the provenance information characterising interaction patterns between the users and the platform. (i.e. to capture provenance information related to steering enactments or progress developing and testing new or refined methods). Templates may capture references to the version of the selected primaty-data, (when available) in combination with data-staging timestamps. They can model dependencies when explicit requests are made by the user for an incremental customisation of the setup of a user computational environment, or when a milestone has been reached (snapshots of sessions). Provenance about the user-controlled customisation of the computational environment, for instance associated with the installation of new libraries, should differentiate between the user-selected libraries and those whose installation has been triggered by internal software dependencies. Moreover, templates can be used to validate the dependencies characterising the update by the user of new terms and concepts in C4. Such a service can be generic and connected to a dedicated store. Evaluation of the ENVRI+ Templating service [Magagna *et al.* 2018a] or those offered by openprovenance.org is needed before considering a home-brewed implementation.

8.2.2 Authorisation and Visibility

Who owns the rights to access the provenance information, and how that should be specifiable via the API. This aims at protecting early work of the researchers, such as the development phase of new methods or the production of preliminary results. Possible solutions should address levels of visibility, limiting the sets of properties, relationships and entities (from conceptual to technical) a certain user may be allowed to access. These rules will affect obviously all the functionalities described in the following sections and all entries in C4. Resource access regulations and FAIR principles may require eventual findability, accessibility, interworking and recoverability of all such information. Avoiding confirmation bias in the provenance records may put pressure on communities to also record things that did not work. DARE should support any choices an application community makes to balance and accommodate these issues, e.g., by enabling future obligations and future actions to be registered in C4 that change the accessibility status of data such as that preserved in P4.

8.2.3 Monitoring

The execution of a workflow can be monitored at different level of details. From views at conceptual level, considering the classification of the components introduced by the users, to more detailed information about the instances and their invocations, during a parallelised enactment of the component. Therefore, triggers can be associated with the occurrence of specific metadata values or value-ranges to automatically initiate a dependent action, for instance, to notify a user or deliver results [Spinuso 2018].

8.2.4 Lineage

It should be possible to explore the lineage associated with a specific data-product bidirectionally. That is, to identify the data from which the product was derived from and the data derived from. These discovered data items should be reachable by the users. They should be enabled to navigate the data derivation graph interactively by specifying how much depth should be returned at each step. Another method that combines graph traversals at a configurable depth with queries on metadata values-ranges could be used to filter a large collection of data products, by excluding those whose ancestors' properties do not match the query parameters [ibid].

8.2.5 Discovery

Users can search for workflow executions and data elements adopting concepts and metadata terms which refer to standard vocabularies, as well as experimental terms introduced by specific application's and researchers' requirements, possibly described in C4. The selection of the terms for the searching is assisted through hints. These should be suggested among the terms and concepts introduced by the user's runs that have been attributed to the S-PROV entities describing parameters, data and components. The same functionalities should be provided when users need to retrieve sessions' snapshots and their dependencies for reproducibility purposes [ibid].

8.2.6 Comparisons

Lineage traces of different executions of a workflow can be used to highlight differences in the results obtained by each step of the workflow. This is applicable when different data-sources are used, as well as different parametrisations or implementations of components that are associated with the same concept. Possibly, concepts should have been previously described in C4, when the users define the items within their conceptual space to be used during a specific phase of the implementation and execution of the method. Such a definition step, which also includes the description of new terms, should be captured by the provenance information system by proposing and validating a suitable template. Similar comparisons should be also provided to compare users' sessions snapshots according to the information captured by the associated templates.

8.2.7 Summaries

Important provenance functions are those providing summaries showing comprehensive information about the provenance relationships occurring within large computations or about the interaction between people and results obtained in different stages of a campaign. These are not necessarily associated with a shared scientific effort.

Concerning computational performances, summaries can highlight processing dynamics, such as data transfer between the different components. These can be grouped into high-level concepts or according to low-level technical properties. For instance, such as the location of the computations or the functions they implement.

Other types of summaries may reveal collaborative dynamics, such as data-reuse between people, workflows and within infrastructures. Summaries should be produced by focusing on a targeted group of properties and values, in order to reduce noise, especially when these are used in to create official reports on the exploitation of the platform and the methods' results, or to instantly highlight visual-analytics tools.

8.2.8 Export

Exporting provenance is intended to support its ingestion, management and processing within interoperable systems which are dedicated to the long-time preservation of **persistent** provenance information, as well as to linked-data engines. In these scenarios adopting interoperable digital formats, such as PROV Notation and RDF is required. The completeness of the entities should be guaranteed. Clients may ask to extract complete provenance data about a workflow's execution or lineage traces associated with a particular data product, which may go across runs. In the latter, the depth of the derivation graph could be offered as configurable parameter. Other export methods could instead allow requests to extract bulk provenance data associated with runs that have been executed within a configurable time-range or attributed to a specific user; e.g., to investigate a bottleneck and conduct directed performance tuning.

8.3 The functions of Common Conceptual Core Catalogue (C4)

An overview of the purpose and features of C4 was presented in Section 4.3. Several aspects of that subsystem are reconsidered here. Two are particularly important:

1. The structuring of the information space by using concepts to help communities share enough information to collaborate even when their diversity and scope expands and
2. The functions and properties of catalogues and their management [Trani *et al.* 2018] which are developed for the DARE platform in this Section.

Overall, C4 should provide a convenient way of naming, talking about and using information, representing anything that a community chooses, without requiring that all data are co-located or compliant with excessive constraints. C4 should help a community growing in diversity to collaborate effectively as they assimilate new data sources, create new methods and establish new practices. Similarly, C4 should enable a growing and evolving set of software subsystems, tools and encoded methods to share their information, so that they can evolve independently and adapt to new digital environments without being rigidly coupled. The rest of this section begins the analysis of the implications of those goals. Section 9.3 initiates the process of making design and implementation choices to achieve these functionalities. Design and implementation issues remain to be resolved by architectural, feasibility and usability experiments in the next period of DARE development.

The C4 functions and evolving persistent state that they provide access to and operations on must appear as a *logical work context* to a user as illustrated for Ann in Section 3.1. However, research developers, who may be the same individuals in a different role, should be able to see the parts of the underlying complex world that they specialise in. They can then develop ways of working with those parts, of combining information and of applying new or revised methods. When they have found ways of making their innovations consistent, complementary and reliable *for the practices they are trying to enable*[78], they may promote their innovations for adoption in the target logical context. We refer a work context that presents a stable, holistic and self-consistent set of abstractions, as a *production context*. We refer to the context where new things can be introduced, and existing things changed, as the *development context.* C4 must support both contexts and maintain the relationships between them. At the start of a *Session* a user may choose an appropriate context by the role they select. The AAAI system will verify they are authorised for that role or this authority may be a property of the *Person* instance recorded in C4, in which case only the authentication and identity verification is delegated.

Reality is more complex than this. Changes that are localised, e.g., affect only the researcher currently acting as a developer, often form a normal part of a day's work. Typically, such innovations are closely intertwined with individuals' expertise and skills. Normally, it is specific to an established practice and is not totally novel. Whereas, some development will be in the deepest parts of the DARE core that may affect all users in every community that installs the new release of

[78] Often the properties that make the innovation useful are limited to these anticipated practices. Errors ensue when use moves beyond this safe envelope.

the DARE core. Hence a range of authorisations and contexts matching those is ultimately needed. We use the *Conceptual* structuring of the shared information space to enable parts of the information space (those associated with a set of *Concepts*) to be visible in detail and able to be changed, i.e., are in the development context. The rest provide a coherent working context, that supports the work at this level, thereby mixing development and production in ways that can be tailored to suit a community's requirements. In particular, the *Concepts* defined by and delivering the core require the most-trusted authorisation as they affect everybody, *Concepts* and entries that are marked **primitive** can only be changed at this highest level. *Concepts* and entries may be defined as **authoritative** meaning they cannot be changed or hidden unless the development-authorisation level has reached the specified threshold[79].

By default, new entries and updates in a C4 are specific to the user associated with the current *Session*. This is achieved by inserting a unique prefix identifying that user, e.g., Ann may have "ann:" as her prefix. This automatically comes into play when a user starts a session, i.e., there is an implicit:

set prefix "ann:"

at the start of each of her sessions. She or others with the authority to see Ann's world, e.g., a research developer working with her, can use this setting to obtain her view at any time.

To enable a user to repeatedly use the same identifiers in a work pattern, e.g., when they launch a new session using a copy of a Jupyter notebook, they may also add a second prefix, **tag**, to identify a specific session context. This they must do explicitly, but they will be prompted when a **new** responds that the identifier already exists.

set tag "nov2918:"

The shared entries are prefixed with a text that identifies the *Group* that is sharing (see Section 8.3.2), e.g., "RAteam:", who are part of "CompSeis:", who are part of "EPOS:" and everybody is part of "all:" where the DARE core is defined. Attempts to set the prefix to any of these groups will warn the user and do nothing if they don't have the authority. By including in queries and searches that identifiers should begin with a specified prefix, individual and group views can be presented. When a search for an identifier fails, the next outer group prefix is tried, repeatedly until "all:" has been tried. This enables innovators to hide prevailing definitions by making a more local one with the same name. A few of the core definitions are marked so that they cannot be hidden because the integrity of the platform depends on their definition.

8.3.1 Functions supported by C4

These are presented in two groups: first, a set of operations to build and populate a C4 and second, functions that use information that is in C4. Both groups will be built incrementally. Readers are advised that this section and Sections 8.3.2 and 8.3.3 are intimately interconnected and that they may need to visit other sections to complete their comprehension. We have not included specific cross-references as this would clutter the text. The update operations that need to be supported are:

1. **new** creates a new empty catalogue so that the DARE team may build a new instance and developers supporting new communities may set up local facilities.
2. **prime** <*source*> populate the catalogue with a sufficient set of initial entries, containing the DARE core, that it is ready to be used; this will have to be done by a privileged script or workflow, as it will not be able to call on stored definitions of **Actions**. Note that *source* may be used to choose a new version of the platform. This will probably be encoded in dispel4py. It will be accomplished by direct developer actions during the DARE project's early years.

[79] For example, 0 implies read only, update and add: 1 implies own only, 2 implies group's, 5 implies communities & 9 implies core seen by all future users.

3. **tailor** <source>[<query>] on the assumption that **prime** has been completed, run a script or workflow to use the concepts, actions and facilities that **prime** delivers to adapt the catalogue to include all the contents required by a particular community. The source chooses versions and is specific to a target application community. This enables previously developed material for that community (from a developer context or a version of DARE based on an earlier release) to be installed. **prime** and **tailor** are allowed to install (new versions of) entries marked **prime** and **authoritative**. Where a query is specified, only the subset satisfying the query is installed. This enables communities to be selective about what they adopt.
4. **restore** <checkpoint-identity> <*transforms*> import into the catalogue all the entries saved from a previous checkpoint operation. Note this allows communities to incorporate new capabilities, without themselves having to worry about the necessary transformations. The DARE core and DARE tailoring teams would ensure that any transformations needed on reinstallation are identified in the *transforms* parameter. Transformations may depend on the *Concept* represented. They may be localised to C4 if other parts of DARE look things up in DARE to find the latest definition maintaining semantic intent.
5. **add** [<*entry*>] inserts a sequence of entries into the catalogue in the order supplied. Each entry encodes the name-value pairs to be recorded. If required name-value pairs are omitted a default, if specified in the *Concept* will be included. The representation of an entry, e.g., as a JSON document or an RDF graph may include the *Concept*; it defaults to *Thing* otherwise. Any checks needed for each entry may result in an error message and *none* of the entries being included. This means that the list members are either all inserted into the catalogue or none are, so that other parts of DARE and users can achieve a transactional addition of a list of entries. The result is either a sequence of data units carrying rejection error messages, or a sequence of identity tokens, accession-time, identity pairs. Both have an order corresponding to list, with acceptance indicators representing entries that did not cause a problem.
6. **update** [<*identity-token*>, <*delta*>] requests that the sequence of entries is updated according to the change specified by *delta* in each case. As before, this delivers a transactional group of updates. The result is a stream of error messages interspersed with acceptances or a sequence of new identity tokens. The *chaining of versions* is automatically handled by this action. A *delta* is a list of *name*, *value* pairs, which may add new optional fields to an entry or replace previous values. If the previous value should contribute part of the new value the user's software should find the previous value, perform the required contribution and submit the new value. The alternative is for such a composite value to be a collection represented by reference to that collection's state. Then a collection update can be used.
7. **annotate** [<*identity-token*>, [<*annotation*>]] for each identity-token and the list of annotations supplied without changing the accession date and without performing a version chaining action. The first annotate on an entry will insert the optional "annotations" field name and start a list of annotations with this as the first, subsequent annotations will prefix this list.
8. **remove** [*identity-token*] discard the sequence of identified entries after verifying that this is permitted and causes no predictable problems as a transaction (all or none). Return a corresponding stream of *'ok'* or error messages as a sequence of data units.

All of the above operations will notify or call P4 appropriately, so that it can maintain a complete consistent with the contents of C4. The read operations are typical of those supported by a catalogue:

1. **find** <query> obtains a stream of data units that match the query, e.g., one using SPARQL. It should be possible to specify a list of identifiers, to specify views, and to specify projections that include any of the name-value pairs associated with each retrieved entry. It should be possible specify value ranges, particularly for the accession time. It may

eventually be possible to ship user-defined functions (UDFs) (Python functions) to apply to each selected entry and to return their yielded result. This matches DARE's philosophy of shipping computation to data, but this may not be available in the triple-stores used or sufficiently safe. The find operation should return a (possibly empty) stream of data units corresponding to the found entries and their projection. Each data unit will start with the full identifier, accession-time pair and then contain the projection results.

2. **navigate** <query> <arcName><depth> for each of the entries identified by query return all of the entries reached by following *arcName* repeatedly up to the specified depth if possible. The result will be a stream of data units, each starting with the starter entry found as for **find** and then followed by a list of entries found by following the arcs.

See Section 8.3.3 for accesses and traversals of collections that may connect with these access operations. There are some management functions that will eventually needed.

1. **checkpoint** which **preserve**s the current contents of the catalogue for recovery and diagnostics purposes; returning as a result a <*source*> URI for **prime**, **tailor** and **restore**. This may use incremental mechanisms.
2. **recover** <*source*> where *source* was returned by a **checkpoint**.

8.3.2 Concepts: their contents and roles

As explained previously, *Concepts* support two roles:

1. They help humans recognise, talk about and make decisions about an abstraction over a population of *Things*, just as they use "Mammal" to denote all the species that are mammals, or even all the individual organisms that are mammals. They choose and define *Concepts* to discriminate from other *Concepts* when the differences are important for the work they are doing. The *Concept* ignores other differences, in the members because for that work these differences are unimportant. A *Concept* may be refined into *SubConcepts* that still have the properties of their *SuperConcept* but now need recognised differences. This leads to an **isa** hierarchy among *Concepts*, analogous with and possibly identical to those that appear in ontologies. The top of this hierarchy in DARE is *Thing,* so every entry in C4 **isa** *Thing*. The significant similarities and differences include properties and behaviour, i.e., the values and relationships that may be found in a *Concept* and the operations which it is sensible to apply to those instances. Debating the useful *Concepts* and their meaning is a vital aid to building mutual understanding necessary for successful collaboration [Trani *et al.* 2018].
2. Concepts help software systems and tools avoid multiple copies of information they need by providing a context for information that applies to all instances of a *Concept* or its *SubConcepts*. This is a recurrence of the object-oriented (O-O) models that underpin **Class**es in languages like Python and underpin inheritance polymorphism. In DARE this will cover the required and optional properties of their instances, default values to hide complexity and the specification of the semantics of the inputs and outputs of *Actions*. Where they are themselves the subject of an *Action* this is isomorphic with methods in an O-O language. We exploit this by having a simple relationship between DARE *Concepts* and Python **Class**es. C4 has a concept *Concept* in which all the common properties and available *Actions* are described that apply to every C4 entry that **isa** that concept. Localising information rather than scattering it through software elements significantly reduces adaptation costs as technology changes. It is therefore an important step towards improving sustainability.

We present an example of the DARE core Concepts in Figure 12 and then discuss some of them to illustrate their use and explain their properties. As we explain in Section 9.3, we draw on others experience using such conceptual structures, such as EPOS's use of DCAT[80]. When we import that work, the details shown here will change, but the form and purpose of DARE Concepts will remain aligned with this presentation. Note that we emphasise here what *Concept* instances would be communicating. They would hold data corresponding to the requirements of the *Thing Concept*

[80] EPOS-DCAT-AP https://github.com/epos-eu/EPOS-DCAT-AP

which every entry must comply with; e.g., they would have as their *name* property: “all:Thing”, “all:Concept”, “all:Person”, etc. They would all have a *primitive* property and an *authoritative* property of 9, to indicate they were part of the DARE core and could only be changed by people with the highest authority. Each *Concept* would also have properties by dint of being instances of *Concept*. For example, the *class* property would have values: “DC_Thing”, “DC_Concept”, “DC_Person”, etc.

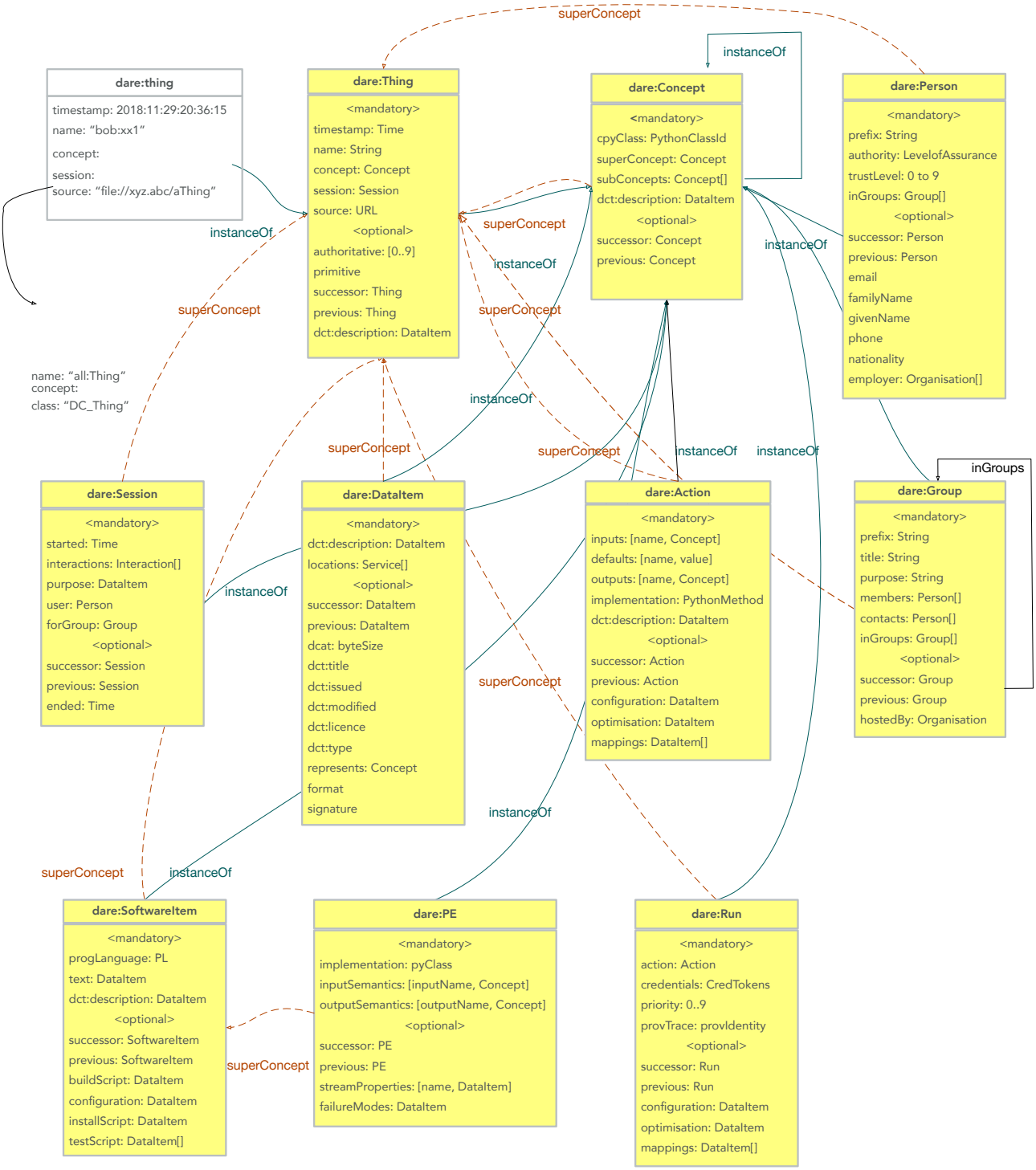

Figure 12: The DARE core Concepts. One instance of Thing is shown in white. Instances of Concept are shown in yellow. The arcs illustrate some relationships; not all are shown.

The interpretation of some of the properties is shown in Table 2. Note that every *Concept* has all of the *Thing* properties and *Actions*, and those of any intermediate *Concepts* on the *SuperConcept* path

to *Thing*. These are not repeated in the Figure or Tables[81]. A provisional set of *Actions* available on instances of each concept are shown in Table 3. The following example *Concepts* are shown in the Figure and the two Tables:

1. *Thing* is the base of a tree of DARE *Concepts*, all branches of the tree inherit from *Thing*; this includes core *Concepts*, tailoring *Concepts* and user-defined *Concepts*. Therefore, every entry in the catalogue has all of the mandatory properties of *Thing* and may have its optional properties. Similarly, all of *Thing*'s *Actions* may be applied to every entry. As a result, any entry may be used anywhere that a *Thing* is required.
2. *Concept* provides the opportunity to name a new *Concept* and to specify its required properties and its *SuperConcept* and the *Actions* which may be applied to instances of that *Concept*. The *description* is made mandatory to help humans interpret it correctly.
3. *Person* denotes any individual interacting with DARE or any individual a user wishes to represent in the default form.
4. *Group*, a set of *Person* instances that denotes an information-sharing and collaboration scope, identified in DARE by a *prefix*. These are properly nested to define an expanding search path out to the whole community, denoted by the prefix *all:*. Of course, these often reflect socio-economic, technical and discipline motivated alliances and teams in the external communities. We imagine, but do not present, the *Concept Organisation* in EPOS-DCAT-AP being a *SubConcept* of *Group*.
5. *Session* denotes a commitment by a *Person* or *Group* to accomplish a number of cognate tasks, performing established methods or creating new methods, in order to achieve one or more goals, described in the *purpose*. Connection with the *Session* may be terminated and then resumed. The period from the start or resumption to the next termination or *Session* end, we call an *Interaction*, possibly represented by another *Concept*, carrying data about the context from which the *Interaction* is being controlled and the devices being used[82].
6. *DataItem* denotes any unit of data that the DARE platform needs to refer to wherever it is stored. It may be stored locally in a storage *Service* managed or used by DARE. Very often it will be stored in some other location, not directly under DARE's control, such as in an archive *Service*. The scale and purpose of most *DataItems* are application determined, though some are used by the DARE core.
7. *Action* a specified sequence or graph of operations that the DARE platform is able to enact. It is normally denoted by a *name* and may have one or more parameters, as named input values, named data input streams or named *DataItems*. It may use other *Actions*. It will normally produce one or more data outputs as DataItems, data streams or progress indicators. It may produce errors. It should either complete or leave the state of the DARE platform unchanged[83]. It should deliver informative error messages and clear up after completion or failure. The user may be able to review progress and issue controls – at least emergency stop to save resources.
8. *Run* denotes an *Action* that has been started and eventually this entry and entity persists in C4 after that *Action* has completed or failed. This is the exception to the rule above that a failing *Action* should leave the state unchanged. This is necessary to provide access to diagnostic information. The *Run* links with the trail of information about the enactment

[81] The names of properties are chosen here for readability. In practice they would be based on widely adopted ontologies, e.g., "*name*" here would be "*skos:prefLabel*" (see Section 9). Similarly, extra detail developed in EPOS-DCAT-AP and found necessary for the DARE platform functions, users and developers will be needed eventually.

[82] Timescales for these *Concepts* that encapsulate purpose, permission and resources in reality involve a longer-running *Concept Campaign*, which we are not proposing to model explicitly at present. *Campaigns* are sustained until objectives are reached, e.g., 50 years for building a working gravitational wave detector or verifying the existence of the Higgs boson. The computational seismologists have identified three. *Sessions* are then human managed steps in such *Campaigns* lasting hours to days. *Interactions* may be based on a sequence of calls from another service or a user checking something they started in a *Session* is still working. Hence these are from seconds to a few hours.

[83] This transactional property may not be achievable during the DARE project. However, it is a requirement, otherwise users and developers have to get to grips with implementation details to organise recovery themselves. This destroys the abstractional simplification that DARE sets out to achieve.

gathered in P4; that should include diagnostic and performance information when they are required.

9. *SoftwareItem* is any unit of software the developers or domain specialists are working with. Very often it will be developed and managed in a development environment such as GitHub. The C4 entry then brings its existence and use into the DARE context.
10. *PE* is a specialisation of a SoftwareItem that can be used in dispel4py workflows as a Processing Element (PE). It takes input from zero or more streams of data units and emits streams of data units on zero or more output streams. Many PEs are specific about the semantics of there data streams, here described in terms of *Concepts*.

Table 2: Properties associated with several DARE core *Concepts*.

Property	Description
Thing <mandatory>	
timestamp	The time at which the entry was **add**ed or **update**d, but the time designated by external sources when their entities are represented. This should be a *Time* entity with about 1 millisecond resolution[84].
name	The user's name for the entity, prefixed by their prefix and optional tag or an externally minted identity, e.g., an identity given by an archival service that was the source [Hellström *et al.* 2017, 2018. 2019]. When no identity is supplied a non-repeating DARE internal identity is allocated, which is different from those issued by other instances of the DARE platform, e.g., each platform instance has its own prefix and it then a count or pseudorandom sequence.
concept	A link to the most specific *Concept* for which **isa** is true for this entry, i.e., this entry is an instance of that *Concept*.
session	A link to the *Session* instance during which this entry was **add**ed or **update**d. This leads indirectly to the *Person*, etc., causing this to happen.
source	Where the information in this entry came from. If user input, the current *Session*. If from a local or accessible file system, the file's URL. If from an external service a URL of that service, possibly extended by a query that obtains a specific item.
Thing <optional>	
authoritative	Integer value in [0,9] indicating authority required to change this entry. 0 implies read only, 1 implies everybody, 2 implies creator only, …, 9 implies DARE core maintainers only. If omitted the *authoritative* value is implicitly 1.
primitive	If present it indicates this instance is part of the DARE core, in which case it cannot be updated, deleted or hidden unless the *Session* has established a level 9 authority to update the core. This is only given to trusted experts. When it is absent the entry can be hidden by definitions in an inner scope and amendments are allowed if the specified authoritative level had been reached – specified in its *Concept* definition.
successor	When the entry has been **update**d then a corresponding entry replaces this one with the new *timestamp*. The *successor* property refers to the new entry. Its presence inhibits further **update**s; they must apply to the new entry.
previous	The new entry, resulting from the **update**, will use this to reference the *previous* entry's value – this enables explicit reversion.
description	A reference to a *DataItem* that describes an instance.
Concept <mandatory> (This includes the mandatory properties of *Thing*: *Concept*'s *SuperConcept*).	
pyClass	The Python class that provides the constructor for entries matching a *Concept* being defined in an instance of *Concept*. It normally has the name *DC_<conceptName>*. It also has methods that define the available *Actions* on instances of this *Concept*. These are named da_<actionName>. At present they compose DARE atomic actions to conduct all that is required by P4, C4 and WaaS. Later some of these may be factored out into WaaS.
SuperConcept	The *Concept* which is the immediate predecessor on the *SuperConcept* path to *Thing*.

[84] DARE should adopt a consistent notion of time for its internal administration, e.g., a *Time* entity from https://www.w3.org/TR/owl-time/. This does not stop users from using other notions and representations of time in their data. Where the *timestamp* is imported from an external source, a translation may be necessary.

subConcepts	The *Concepts* that have stated this *Concept* is their *SuperConcept*, which may be none. Instances of *subConcepts* satisfy the **isa** relationship with this *Concept*. This means they are included in the implicit collection – see Section 8.3.3.
description	An explanation of what the *Concept* means and how it should be used. Note this is held as a separate *DataItem*, so that it can be in any form a community requires and updated independently from the formal *Concept* entry.
Concept <optional> (as for *Thing* plus)	
defaults	A list of defaults for instances of this *Concept* as a list or dictionary of <*propertyName*, *Value*> pairs.
Person <mandatory> (This includes the mandatory properties of *Thing*: *Person*'s *SuperConcept*).	
prefix	The string of Unicode characters used to distinguish entries specific to this person in the C4 catalogue.
authority	The *authority*, based on accreditation processes, this *Person* has on *this* DARE-platform instance.
trustLevel	The recognition of the maximum level of authority this person may seek in any *Session*. Their current *Session* may be at this level or at a lower level.
inGroups	The sequence of DARE *Groups* this person is directly a member of. Their searches may be extended to include those *Groups'* prefixes as well as "all:".
Person <optional> (This includes the optional properties of *Thing* : *Person*'s *SuperConcept*).	
email	
familyName	
givenName	
phone	
nationality	
employer	An *Organisation* providing AAAI information and taking legal responsibility.
Group <mandatory> (This includes the mandatory properties of *Thing*: *Group's SuperConcept*).	
prefix	The string of Unicode characters used to distinguish entries specific to this *Group* in the C4 catalogue
title	Informative short name as a String
purpose	Precise but inclusive explanation of the goals of this *Group*. It may need to be represented in a *DataItem*.
members	The sequence of individuals as *Person* instances in this *Group*, with their joining and optionally leave dates.
contacts	The sequence of individuals, as *Person* instances, to contact when issues arise with this *Group*.
inGroup	This Group may be part of a larger group, until Group All is reached, which denotes the total community served by a DARE instance.
Group <optional> (This includes the optional properties of *Thing*: *Group's SuperConcept*).	
hostedBy	An *Organisation* (*subConcept* of *Group*) that is a legal entity that takes responsibility for a *Group.*
Session <mandatory> (This includes the mandatory properties of *Thing*: *Session's SuperConcept*).	
started	The Time at which this *Session* was started[84].
interactions	The sequence of *Interactions* occurred so far associated with this *Session*.
purpose	The reason this *Session* is being performed.
user	The *Person* who started and is responsible for this *Session*.
forGroup	The *Group* for which this *Session* is being conducted, this may determine resources and accounting and the *Person* may be in more than one *Group* or working for a *Group* for which they are not a member, e.g., to investigate and remedy a reported issue.
Session <optional> (This includes the optional properties of *Thing*: *Session's SuperConcept*).	
ended	While this is absent the *Session* may still be resumed or continue. This indicates the time at which the *Session* was completed or terminated, a decision made by the user.
DataItem <mandatory> (This includes the mandatory properties of *Thing*: *DataItem's SuperConcept*).	
description	Whatever a user or software needs to use this *DataItem* correctly. Note the promotion from optional. Eventually, this may be a dictionary of DataItems to accommodate different user personas, internationalisation, and multiple software handlers.

locations	A list of locations, e.g., as URLs, from where a copy of the DataItem is available. Perhaps in the order they should be tried or considered by optimisers, when there is more than one.
DataItem <optional> (This includes the optional properties of *Thing* : *DataItem's SuperConcept*).	
byteSize	The size of the *DataItem* a la DCAT
title	The title used when reporting or visualising this *DataItem* a la DCAT
issued	The *Time* that this *DataItem* was issued by its originating authority a la DCAT.
modified	The Time at which this *DataItem* was last updated.
licence	A *DataItem* or URL containing the licence details, or a recognised licence name.
type	A specification of this *DataItem* to have a known type from an agreed ontology.
concept	The DARE Concept this *DataItem* represents.
format	The format in which it is stored.
signature	A computed hash to detect when the referenced data has changed – used to detect accidental data loss or malevolent intrusions.
Action <mandatory> (This includes the mandatory properties of *Thing* : *Action's SuperConcept*).	
description	Whatever a user or software needs to use this *Action* correctly.
inputs	A dictionary of <*inputName*, *Concept*> pairs specifying constraints on each input. Unspecified inputs are free to use any *Thing*. Because of the assumption of auto-iteration over streams of data units an input may be a sequence of *ConceptX* when this says *ConceptX*[85].
defaults	A dictionary of <*inputName*, *Value*> pairs specifying a default value to use when an input is not supplied.
outputs	A dictionary of <out*putName*, *Concept*> pairs specifying the *Concept* delivered by each output of the *Actio*[85].
implementation	A Python method or Python script implementing the *Action*.
Action <optional> (This includes the optional properties of *Thing*: *Action's SuperConcept*).	
configuration	A *DataItem* specifying the required configurations for this *Action* if necessary, in a form required by the current implementation of WaaS. When multiple targets are envisaged with different configurations, this will be a dictionary from target to configuration.
optimisation	A *DataItem* specifying an optimisation recipe for this *Action* if necessary, in a form required by the current implementation of WaaS. When multiple targets are envisaged with different optimisations, this will be a dictionary from target to optimisation.
mappings	A *DataItem* specifying a mapping recipe for this *Action* if necessary, in a form required by the current implementation of WaaS. When multiple targets are envisaged with different mappings, this will be a dictionary from target to mapping.
Run <mandatory> (This includes the mandatory properties of *Thing*: *Run's SuperConcept*).	
action	The outermost *Action* that was launched (normally by a user action) and that may be calling inner *Actions*, which may or may not warrant a *Run* instance – to be decided.
credentials	The authorisation tokens and accounting credits used to gain permission for running this *Action* that may be presented and or consumed during this *Action*.
priority	An indicator [0:9] to WaaS of the desired trade-off between speed and costs (e.g., response time versus energy & GHG emissions)
provTrace	A link to the provenance trail data in P4 that may still be being accumulated.
Run <optional> (This includes the optional properties of *Thing*: *Run's SuperConcept*).	
configuration	As for *Action*, but this may be while the *configuration* is still being decided or has yet to be proved successful. In that case, it may be needed here, e.g., for diagnosis or to be copied to the *Action* on successful completion. A *DataItem* specifying the required configurations for this *Action* if necessary, in a form required by the current implementation of WaaS. When multiple targets are envisaged with different configurations, this will be a dictionary from target to configuration. It may include computational-contextualisation for P4.
optimisation	With a caveat similar to *configuration*. A *DataItem* specifying an optimisation recipe for this *Action* if necessary, in a form required by the current implementation of WaaS. When

[85] Some *Actions* may handle inputs of any *Concept* and yield outputs of the same *Concept*, e.g., **copy**. To capture this, input *Concepts* can be denoted by Greek letters capitalised, e.g., *Alpha*. Those same Greek letters associated with a result, then indicates that this result is an instance of the *Concept* arriving via parameters that had the same Greek letter.

	multiple targets are envisaged with different optimisations, this will be a dictionary from target to optimisation. Again, P4 may extract information from this.
mappings	With a caveat similar to *configuration*. A *DataItem* specifying a mapping recipe for this *Action* if necessary, in a form required by the current implementation of WaaS. When multiple targets are envisaged with different mappings, this will be a dictionary from target to mapping. Again, P4 may extract information from this.
SoftwareItem <mandatory> (This includes the mandatory properties of *Thing*: *SoftwareItem's SuperConcept*).	
progLanguage	A specification of the programming language the top level of the SoftwareItem is written in.
text	A *DataItem* stating where the *text* corresponding to the top level of the *SoftwareItem* s located or containing the *text* directly – to accommodate a common form of co-working (see Section 5).
description	A *DataItem* explaining what this *SoftwareItem* does and ho0w it should be used. Note the promotion from optional. Eventually, this may be a dictionary of DataItems for different user personas, internationalisation, and multiple software handlers.
SoftwareItem <optional> (This includes the mandatory properties of *Thing*: *SoftwareItem's SuperConcept*).	
buildScript	A *DataItem* containing a script to build a version of the *SoftwareItem*. Possibly with some *configuration* options specified.
configuration	A *DataItem* in the form used by the DARE platform to specify the required computational context, or to find/build such a digital context if necessary.
installScript	A *DataItem* containing a script to install a version of the *SoftwareItem*. Possibly with some *configuration* run-time options specified.
testScripts	
PE <mandatory> (This includes the mandatory properties of *SoftwareItem*: *PE's SuperConcept*).	
implementation	A Python Class that implements this PE.
inputSemantics	A dictionary from *inputName* to *Concept* specifying the *Concept* each incoming data unit or the series of data units should represent on the named input channel[86].
outputSemantics	A dictionary from *outputName* to *Concept* specifying the *Concept* each output stream carries.
PE <optional> (This includes the mandatory properties of *SoftwareItem*: *PE's SuperConcept*).	
streamProperties	A dictionary from *outputName* to *DataItem* that contains a description defining the expected properties of the output streams in a form used by the WaaS.
failureModes	A list of known failure modes and their implication.

Table 3: Actions provided initially with the initial core Concepts.

Actions	Description
on instances of **Thing** and therefore applicable to every catalogue entry	
add	Create a new instance according to the Python dictionary[87] supplied as an argument to the *DC_Thing* constructor, with the string names matching intended properties, and any default values already added. Then insert that instance into the C4 catalogue.
update	If the subject of the **update** already has a *successor* then raise an error. Otherwise, insert a new entry with the same *identifier* and a new *timestamp*, forward linking from the instance's entry by providing a *successor* link. Replace all of the properties named in the Python dictionary supplied with the associated new values. Other properties are unchanged.
read **read** *name***[]**	Return a value which is a Python dictionary with the top-level (i.e., not following additional linkage graphs) of the instance, with property names and a representation of their corresponding values. The user could then modify this dictionary and use it as an argument to an **add** to create and insert a new entry with controlled differences. When a

[86] As before, a Greek letter may be used to show that any kind of data unit may be supplied and that the same data unit flows out on similarly labelled output streams, e.g., *splitter* PEs take in a stream of data units representing instances of *Alpha* and emit on each of their outputs a partition of those input data units still denoting *Alpha*..

[87] An other means of denoting a set of <*name*,*value*> pairs, e.g., a JSON document may be used by implementers of the DARE platform, but the choice should be made consistently.

	list of the names of the instance's properties is supplied, just those properties are read and returned.
preserve	Make certain the information in this entry is not lost (according to criteria prevailing for this community). When this has already been done then this is a quick no-cost action. This may have a different interpretation for some Concepts, e.g., to include referenced data if it is stored under DARE's control.
remove	Take the entry out of the catalogue. As with many other *Actions* this will require sufficient authority.
visualise	Provide a visual (and interactive in many cases) presentation of the value of this instance. It will typically default to a standard LoD visualiser useful for research developers. Normally, this will only show the immediate entry but allow expansion of connected entries and drill-down into referenced data. The MVV developed by WP3 would be a suitable candidate – see Section 9.2.1.
export	Return a DataItem that represents this entry in a standard textual form for LoD adopted by DARE platforms. This is necessary data to be used for **promotion** beyond enclosing groups and for operations such as **prime**, **tailor** and **restore** on deployed DARE platforms – see Section 8.3.1.
on instances of **Concept** and therefore applicable to every *Concept* entry	
DC_Concept	Build and insert a new *Concept* according to the supplied dictionary
on instances of **Person** and therefore applicable to every *Person* entry	
DC_Person	Introduce a new *Person* according to the supplied dictionary
join g	Arrange that the *Person* instance is now a member of *Group g*.
leave g	Arrange that the *Person* instance is no longer a member of *Group g*.
on instances of **Group** and therefore applicable to every *Group* entry	
DC_Group	Introduce a new *Group* according to the supplied dictionary
join g	Arrange that the *Group* instance is now a member of *Group g*.
leave g	Arrange that the *Group* instance is no longer a member of *Group g*.
on instances of **Session** and therefore applicable to every *Session* entry	
start	Begin a *Session* creating it at the current time with the identity and credentials of the current user but with other values set by the supplied dictionary
leave	Leave this *Session*, i.e., terminate the current *Interaction*.
resume	Return to this Session possibly with some update of resettable properties.
terminate	Terminate this *Session* indicating a completion status.
analyse	Provide an analysis of the *Session* and its work to help developers (implement later)
on instances of **DataItem** and therefore applicable to every *DataItem* entry	
DC_DataItem	Create and enter a new DataItem according to the supplied dictionary.
ingest	Load a referenced file according to the supplied dictionary, storing it under DARE's control. Note that if the *Concept* it represents (*represent* property) it may be tested and transformed to a form required in this DARE community for that *Concept*. This may also derive extra property values based on metadata discovered from the file or its source.
use	Arrange that the externally managed file specified is presented as if it were within DARE's control with the interpretation specified in the *represent* property and the supplied or extracted other property values. There will be mechanisms for synchronising (**pull**ing) with external updates and for **push**ing updates made via the DARE platform. This is necessary for the co-working pattern of incremental adoption – see Section 5.
move	Transfer a copy of a *DataItem* to a specified destination if it is not already there according to a specified *priority* [0,9] with 0 = lazily when needed, 1 = at minimum cost to 9 = pre-empting lower priority resources as quickly as possible.
on instances of **Action** and therefore applicable to every *Action* entry	
start	Cause an *Action* to be performed subject on implicitly associated instances or as explicitly parameterised.
define	Set the Python script supplied in a *DataItem* as a parameter to be the definition of this *Action* if that is permitted, and deal with any consequential updates, such as revising configuration requirements. This could all be done with update leaving the responsibility for consistency to external code. It may be better to embed that consistency and to expect WaaS to ensure it each time the *Action* is *Run*. This is preferable for script authors, who

	will be around and remember the changes they have made, if the error is detected promptly. First use of the *Action* may occur much later.
monitor	*Run* the *Action* with full monitoring for diagnostic purposes.
analyse	Provide an analysis of the *Action* and its *Runs* to help developers (implement later)
on instances of **Run** and therefore applicable to every *Run* entry	
observe	For a completed *Run* this is equivalent to **read**. However, for a running enactment of an *Action* it may provide progress and monitoring information. This should avoid forcing users to look inside the level of abstraction they are working with.
control	If the enactment has not yet finished the user may supply updates to parameters. If the *Action* has been appropriately coded or configured, this may modify the behaviour of the rest of the *Action*. Running enactments may set up additional control points and name them. Their names may be found via **observe** if they are not already known. Users may supply relevant values by having in the dictionary input to control the relevant <*controlpointName*, *value*> pairs.
stop	Force the enactment to do an emergency halt and clean-up. Supply the provenance trail with a termination record that includes a user-supplied reason.
analyse	Provide an analysis of the *Run* and to help developers (implement later)
diagnose	Collect information, from the provenance trail and any running parts and perform an analysis of these to explain to the user or a software system request the current or terminal state of the enactment. This should avoid forcing users and developers having to look inside the level of abstraction they are working with.
on instances of **SoftwareItem** and therefore applicable to every *SoftwareItem* entry	
build	Build the specified software, e.g., by a compilation and binding process. This, and the next two *Actions* will call standard development systems in most cases.
Install	Install the software in the current or a specified destination, e.g., a container.
test	*Run* the list of tests on this *SoftwareItem* in the order listed, reporting any failures. It should be possible to specify whether these *Runs* should be recorded in P4.
on instances of **PE** and therefore applicable to every *PE* entry	

These initial ideas about core *Concepts* will be developed, revised and extended by exploring their power with DARE's user communities and research developers. The initial core set of *Concepts* and their properties and Actions will be defined during the next 12 months.

8.3.3 Collections in C4

It is necessary to empower both researchers and developers with methods that support repeated processes. This grows in importance as the scale of data increases and as evidence of sufficient quality, particularly of time-varying phenomena, demands repetition of processes. Iteration over *Collections*, predominantly sequences (aka lists) and sets (occasionally bags) is the traditional foundation for this power. Databases and files deliver ways of representing such *Collections*. Queries against databases and *Services* that handle queries in their API provide a means of logically synthesising further *Collections* without materialising and transporting whole *Collections*, in a way that can be efficiently coupled with processing, e.g., by using embedded user-defined functions (UDFs) in queries or by streaming results to processing elements supported by Exareme and dispel4py.

The construction of dictionaries, collections of <name, value> pairs provides another form of *Collection*. This *Collection* pattern is manifest in individual entities in DARE. But it is also represented by indexes, by catalogues and in graph data bases.

Users and software need to handle *Collections* efficiently and have relevant operations over them that are easily used. These are illustrated and introduced in Table 4. Users and software may explicitly create and populate *Collections* they manage explicitly. However, there are a number of implicit collections. For example, there are sequences implicitly defined associated with each *Concept*. They are named by replacing the *Concept*'s first letter with its lower-case counterpart and

adding an s at the end. For example, *things* is the list of all *Thing* instances in order of rising first insertion or *timestamp*. That identifier without the trailing *s*, i.e., *thing* in this case, denotes the current instance as an iteration over the members proceeds.

Table 4: The operations on DARE collections.

Operation	Description
add *Concept*[] **add** *Concept*{} **add** <Concept, Concept>	Create a new instance according to the *Collection* pattern indicated with an empty collection of that form and insert it into the DARE catalogue with the property values supplied as a Python dictionary.
update	Perform one of the Python updates on *Collection* values for the particular *Collection* pattern, including assigning a new value of the same pattern. Subject, as always to integrity and authorisation constraints.
where	An expression that yields a *Collection* with the same pattern as it is applied to but containing only those elements in the original that satisfy the expression after **where**, possibly supplied as a Python function yielding a *Boolean* function, but if supplied as a query, optimisers may speed evaluation. A special predicate **last**, runs to the end of the *successor* chain for each entry.
apply	Applies a supplied *Action* or Python function to every entry in the *Collection*.
map	Applies a supplied *Action* or Python function to every entry in the *Collection* and forms a new Collection with the same pattern with the results produced.
f(*Collection*)	Apply an *Action* or function to an entire collection. Many will be recognised, *Average*, and have special implementations.

The map-reduce model of deriving values, as well as efficient mappings onto storage, optimised representations, parallelisation, pipelining, and functional transformations underpin the collection handling in a completed version of the DARE platform. However, these are hidden from users and external APIs. Research developers may require to be aware of them on some occasions.

8.4 Requirements for architectural integrity

Mutual interdependencies and commitments must be maintained at all times within a DARE platform, in order that it retains consistency and does not circumvent the long-term integrity requirements with temporary short cuts. In brief, the catalogue function of **C4** must be the logical source for interpreting all actions, so that users' intentions are maintained as technology changes, and so that technology evolution can be accommodated economically by changes in one place. This use of **C4** is essential if DARE is to take responsibility for data location, data movement and computation deployment, all of which are necessary for coping with scale and with minimising costs. All actions, except the primitive (built-in) commands, must be handled via the **WaaS**. The **C4** system, the primitive commands, and the WaaS must all write provenance records to P4, so that the recorded history is pervasive. **C4** and **P4** must take responsibility for long-term state preservation. **WaaS** must take responsibility for enactment-target selection, monitoring and recovery. The interactions between the three pillars are summarised below.

The required relationships, denoted by a, b and c in Figure 3, are summarised here.

1. **C4** must provide (a) and preserve all the terms and their definitions the **WaaS** needs to create and execute workflows, i.e., it contains the mapping from users' intentions to practical implementations, that are used to transform from abstract to concrete workflows, to optimise those mappings, and to select and, if necessary, prepare and deploy enactment contexts. **C4** must provide and preserve (c) for the duration of provenance records all the provenance system (**P4**) references [Trani *et al.* 2018][88].

[88] Under normal circumstances there are many runs of workflows generating provenance traces compared with the number of updates to terms they use in the **C4**. Consequently, tracking versions of the **C4** entries so that references to it are reliable, involves much less work and storage, than copying snapshots into the provenance traces.

2. The **WaaS** must request information from the **C4** (a). This information must shape the interpretation of re-mappable *Actions*, especially scripts and workflows. **WaaS** must pick up current translations from users' intents to corresponding implementations. It must get its optimisation and target choice data from **C4** or generate it from analysis of **P4** histories and store it in **C4**. It must find in **C4** target descriptions, deployments and deployment construction recipes[89]. It may also run workflows to implement **C4** or **P4** actions (a), e.g., implementing **promote** to issue new versions, reconciling during a **pull C4**s that have progressed independently or **ingest**ing new material to **add** content. For **P4** it could analyse recent selected activity and present a steered visualisation to managers or users. The **WaaS** must supply provenance records for all its actions (b) in the required form, translating when necessary from underlying systems and software, e.g., from logs [Magagna *et al.* 2018a].
3. The **P4** must provide information to support replay, e.g., after a partial failure or when a user revises their method, with or without user modification (b). It should be possible to mine the histories of previous runs to support the optimisation of current workflow runs (b) and to enable improved platform management. The **C4** system should be able to analyse what changes have been made to a **C4** that has been used independently (c) in order to generate input to inbuilt workflows, such as **promote**.

When aspects of the system change or when failures are detected, it is necessary to coordinate the three pillars by means of a notification system such as Kafka.

9 Technology review

This is a limited review concerned with the technologies pertinent to the current issues, identified earlier in this document and to the immediate or near-term plans that have not already been analysed. Technology will be further investigated in task T2.4.

9.1 Technology for WaaS

The role of WaaS was introduced in Section 4.1 and analysed to develop specific architectural requirements in Section 8.1.

A review of work relevant to WaaS is presented here, as DARE builds on previous research.

- In [Filgueira *et al*. 2016b], it is presented the Data Intensive Workflow as a Service (DIaaS) model to provide scientists with a flexible and easy-to-use environment for running scientific applications within containers. In this model, all required software (workflow systems and execution engines) are packed into the docker containers, which significantly reduces the effort (and possible human errors) required by scientists or platform administrators to build such systems.
- In [Wang, Jianwu *et al*. 2014], the authors present a new Workflow as a Service (WFaaS) architecture with independent services. A core part of the architecture is how to efficiently respond to a continuous stream of workflow requests from users and schedule those workflow enactments Cloud platforms. Based on different targets, they propose four heuristic workflow-scheduling algorithms for the WFaaS architecture and analyse their differences and best usages of the algorithms in terms of performance, cost and price-performance ratio (PPR) via experimental studies. And they found that an algorithm with proper configuration could reduce both cost and PPR without impacting overall performance.
- In [Rodriguez *et al*. 2018], the authors propose a resource provisioning and scheduling strategy designed specifically for WaaS environments. The algorithm is scalable and dynamic to adapt to changes in the environment and workload. It leverages containers to address resource utilisation inefficiencies and aims to minimise the overall cost of leasing the infrastructure resources while meeting the deadline constraint of each individual

[89] For speed of enactment, it may cache mappings and target selections. However, when information that changes these is updated in **C4**, the platform must invalidate these mappings.

workflow. This is the first approach that explicitly addresses VM sharing in the context of WaaS by modelling the use of containers in the resource provisioning and scheduling heuristics.

- The work presented in [Rodriguez *et al.* 2017], identifies these challenges and studies existing algorithms from the perspective of the scheduling models they adopt as well as the resource and application model they consider. A detailed taxonomy that focuses on features particular to clouds and WaaS are presented, and the surveyed algorithms are classified according to it. In this way, the authors aim to provide a comprehensive review of the literature.
- In [Wang Dingxian *et al.* 2014], the authors illustrate how the current cloud workflow systems are mainly used by people who possess the knowledge about the business processes such as the process structures and functional/non-functional requirements because they need to create the executable workflow applications by themselves. However, most end-users do not acquire such knowledge. This limits the system usability and hinders the implementation of complete WaaS. To address this problem, their paper proposes the design of a novel online workflow recommendation system which would help end-users create their own executable workflow applications by only providing some keywords to describe their requirements without undertaking the specific workflow modelling process.

The development of the DRIP system for mapping complex workflows with multiple time constraints onto heterogeneous distributed cloud infrastructure may provide useful technology and insights [Wang *et al.* 2017]. It builds models of the workflows, and of the distributed infrastructure in its knowledge base and uses these to optimise workflow deployment. Its knowledge base must therefore contain representations and maintenance techniques that will prove close to key parts of DARE's WaaS technology.

In DARE the initial steps to build the relationship between WaaS and C4 will include:

1. Each **Concept** that is implemented in C4 has a corresponding Python Class *DC_<ConceptName>* – see Section 8.3.2. Many of these will transform an abstract workflow template to a parameterised WaaS request for enactment, initially formulated as dispel4py or Exareme graphs. For *Actions* used in dispel4py workflows a corresponding PE should also be developed. This poses interdependencies to be navigated.
2. For the built-in **primitive** *Actions*, some require a small workflow for their implementation; others are just coded in native Python. A selection of these are core to every other activity. Identifying that critical core (see Sections 4.3 and 8.3) and prototyping a bootstrap version of each is a priority.
3. The description of PEs and the construction of a useful population of such descriptions, would enable tools and checkers to be developed, accelerating dispel4py authoring and reducing dispel4py error rates – see below.

The range and encoding of data units in d4p's streams, should handle scale and diversity. At the very least, they must handle in some way all the different forms of data the user communities need (see Section 7.1 and 7.2).

PEs that match communities' requirements, should wrap their processing, analysing, modelling and visualisation algorithms, with inputs and outputs in the agreed form. These PEs should be represented by a *Concept PE* in the catalogue (A subconcept of *Action* and *SoftwareItem* perhaps) with descriptions useful to humans and descriptions needed by software. Whatever data marshalling is needed should be wrapped in a PE deployed to the sites where those data are required.

Helping users authoring correct workflows is a very open-ended issue that will need addressing incrementally as failure modes are detected, but some of the following should be done pre-emptively.

1. Describe the PEs so that the required input and the produced types for each data unit on a port and hence a stream are explicitly defined (semantics and format) and use this to guide authors and to verify stream connection consistency before mapping and enactment (this was prototyped in DISPEL [Martin & Yaikhom 2013]).

2. Describe the PEs so that the data-unit consumption patterns and output patterns are defined as well as the relationship between input flow rates and output flow rates. Use this as input to deadlock and live-lock prevention as well as for optimisation.
3. Modify the implementation of the output ports of PEs to start the sequence of data units with a start-of-stream (SoS) marker. This should include the source (*PEInstance*) identity, the definition of semantic type and representation of the data units, and, if known, an indication of the number of expected data units. This is a step towards localising the failure detection.
4. Modify PE input port implementations, so that:
    a. If the stream does not start with an SoS an error is raised.
    b. If the expected data units do not match local expectations (semantics or representation) then raise an error.
    c. Make a note of the source's identity for failure propagation and No-Longer Interested (NLI) signals.
    d. If the incoming sequence of data units is incorrect, raise an error, implicating the source.
    e. If the rate of arrival or total number of data items exceeds a threshold, raise an error.
    f. If data arrives after an NLI signal (see below) has been sent to the source, raise an error. It may be necessary to introduce a tolerance here to handle signalling latency.
    g. If expected data doesn't arrive before a default / set timeout, then raise an error.
5. Arrange that the PE abstract class catches unhandled exceptions and transforms these to error handling with diagnostic data captured (see below).
6. Arrange that the error handling system for a PE does the following:
    a. If there is a local error-recovery mechanism coded for this kind of error apply it. Otherwise, minimise the impact by promulgating warnings and by performing local clean up.
    b. Send a No-Longer Interested (NLI) message to each source for each input port using the saved identities. It propagates the shutdown upstream until a recovery boundary is reached.
    c. From each output port send an End-of-Stream (EoS) marker with the error encoded. This also flushes all output streams. It propagates the shutdown downstream until a recovery boundary is reached.
    d. Using the platform's signalling system monitored by other subsystems, send an Error report. New issue: what do these contain and how are they encoded?
    e. Release any temporarily reserved local resources.
    f. Shut down the local interpreter.
7. Modify the PE abstract class to send an EoS with a success marker and data-unit count on successful completion from each output port. The input ports will check their count to detect lost data units. They will notify their processing algorithm that their input stream is complete. When all inputs have ended, and the algorithm has run to completion the PE performs a successful-completion shut down.
8. The successful-shutdown flushes all output stream buffers and appends to them an EoS with a success marker and data-unit count. This propagates the shut-down downstream along the graph. Finally, the local resources associated with this *PEInstance* are released.
9. A controlled *StopStartFlow* PE will be provided which takes any input stream of data units and emits an identical stream on its output port. Instances of this PE may be set up with the flow open or closed initially. Signals may be sent to it, from a control app or from orchestration software to retrieve the data-unit count and setting, and to set the flow setting (with an optional specification about whether to send NLI and EoS markers). This can be used to switch on and off iteration loops, to switch on and off process visualisation and to perform algorithmic flow control. This allows users to steer dispel4py enactments.
10. The buffering within input ports and in output ports should be as elastic as possible with settable upper bounds. This should accommodate some failure modes, e.g., a PE performing an ordered merge swamped on one input with nothing on the other. An elastic buffer handles acceptable timing variations. A bounded buffer eventually detects an error. It is possible that

using Kafka for data transport will be a good strategy for implementing this. However, it may result in non-scalable data-traffic focused on Kafka servers.

11. Setting a time-out on input ports will pick up further failure modes.
12. Some workflows may be long-running, or indefinitely continuing, e.g., as a continuous signal is processed. They may have branches in their graph only activated when a rare condition is detected, when a steering user switches on a monitoring, diagnostic or visualisation flow. There are two advantages to dynamically deploying the graph:
    a. Resources are not consumed deploying the unused parts of the graph (many when failures occur) nor while waiting for input to arrive after deployment, and
    b. When a subgraph deployment is dynamic, all the information discovered from the earlier part of the graph is available to inform deployment, e.g., the size and content of the data units, the length or rate of steams, etc., and the current state of the platform is known – this should result in better optimised deployments.

    The subgraphs necessary to gather diagnostics and translate them into a comprehensible form are necessary but potentially complex. Managing them by dynamic binding is particularly worthwhile, as they can then avoid the partial failures that have led to a problem. As greater scale is attempted, algorithms in the early part of a workflow can calculate optimal parameters for subsequent sub-graph deployments.
13. The fragment of workflow graph needed can be encoded and an output port that is sending an SoS can call on an interpreter of that encoding to grow the next step in the graph and return the connection targets it should use.
14. Scalability can depend on bespoke parallelisation. The replication of subgraphs is already supported in *d4p*. However, the splitting and merging algorithms are currently limited to those in the target platform. Encoding these as PEs, with standard names for those built-in in some mappings, opens up algorithmic scaling that depends on properties of the data being processed. This requires vectors of output and input ports respectively.

The above extensive list needs to be selectively investigated. Prioritising a few selected items from that list is an issue. There is already an experiment underway investigating the first two items. Error detection and error handling should be a high priority, so that other DARE activities do not encounter inexplicable failures during dispel4py enactments.

9.2 Technology for P4

DARE wants to accommodate the dialog between different users' roles and expertise. This is supported by the concepts expressed in **C4** in combination with the granularity and details captured by the **P4** system. Provenance should describe the causal-effect relationships between data and methods involved in the running system, as well as the interactions of the users with and through the platform. Both contribute to the evolution of the DARE instance within a specific application domain. We distinguish in this section between the different aspects of creating, managing and exploiting provenance data, describing the current technical framework and the envisaged extensions.

9.2.1 Capturing provenance information

The actors that will trigger the generation of most of the provenance information will be the research-developers. We aim at supporting them in tuning and disambiguating the lineage produced by their computational methods with ultimate objective of guaranteeing usability of the records. This is achieved by balancing between full automation and ad-hoc adjustments of the provenance extraction mechanisms, with the application of provenance patterns that can be further contextualised and re-used across operators. The result is to increase the precision and descriptiveness of the lineage associated with a specific campaign, fostering classification and analysis of the traces produced by many interconnected and evolving experiments.

As previously mentioned, dispel4py is technology offering a workflow language to the DARE platform. This will be further extended to support more functionalities, which already include a technical framework to capture and tune lineage information. This is based on an approach that considers a workflow component described by a Python class. This class defines the behaviour of its

instances as their type, which specifies what an instance will do in terms of a set of methods. We introduce in this typical setting, the concept of *Provenance Type*, that augments the basic behaviour by extending the class native type, so that a subset of those methods performs the additional actions needed to deliver provenance data. Some of these are being used by some of the pre-existing methods, and characterise the behaviour of the specific provenance type, some others can be used by the developer to easily control precision and granularity.

We distinguish between provenance **Pattern Types** and **Contextualisation Types**, to be used in combination. The former capture provenance patterns by applying rules associated with the events triggered by the ingestion and production of data-streams within components. The latter are used to extract domain metadata from the newly produced Data. Domain contextualisation is achieved by adopting standardised metadata formats, specific ontologies or vocabularies, the generation of unique identifiers in compliance with a well-defined schema.

In order to enable the user of a computational application to configure the attribution of types, selectivity controls and activation of advanced exploitation mechanisms, we introduce also the concept of **provenance configuration and profiling**. In Figure 13 we illustrate the different phases envisaged by the framework, to prepare the provenance-aware execution of a workflow. In that respect, we propose a configuration profile, where users can specify several properties, such as attribution, provenance types, semantic clusters, sensors and selectivity rules. Selectivity will be used to enable provenance recordings only for data with specific properties and value-ranges, while sensors analyse the provenance traces for runtime profiling purposes, establishing feedback loops into the workflow to trigger steering actions.

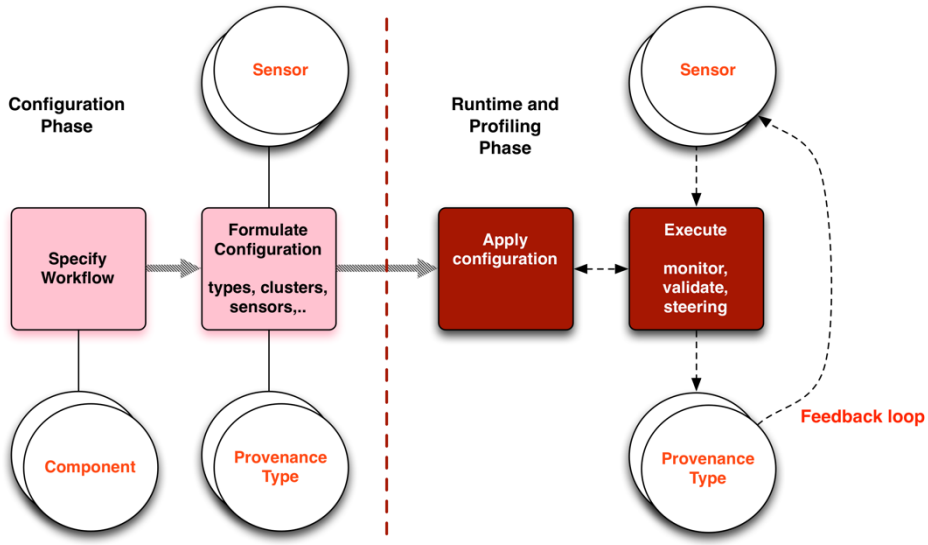

Figure 13: Provenance Configuration and Profiling. Phases involving the preparation of a workflow and its provenance-aware execution. Once the workflow is specified, the user formulates a configuration that will be applied when the workflow is initialised. She indicates the provenance types of the components, to capture patterns and metadata describing the use of the data flowing through the system. The configuration profile includes the specification of clusters of components and may include provenance sensors associated with these clusters. Sensors read and analyse the provenance traces for profiling purposes, such as monitoring and validation and establish feedback loops into the workflow to trigger steering actions. These could include re-parametrisation or the re-implementation of the component internal functions, or the dynamic re-assignment of a provenance type.

We consider that a chosen configuration may be influenced by personal and community preferences, as well as by rules introduced by institutional policies. For instance, a Research Infrastructure (RI) may indicate best practices to reproduce and establish requirements on the actions performed by the users exploiting its facilities, or even impose requirements which may turn into quality assessment metrics. DARE will extend this capturing mechanism by including new patterns, such as **Interaction Patterns**, describing how users' approach the usage of the platform. Such patters may suggest to users that they specify access control rules for a group or co-workers, as well as linking to a specific vocabulary of domain concepts. Thereby, interaction patterns may set the constraints and the templates representing the activities, the entities and their provenance relationships involved in performing operations on *Things* and *Concepts***,** as introduced in Section 8.3.2. Developers may be asked to indicate the level of maturity of the methods and datasets produced, including the metadata terms adopted for the experimental description and validation of the results. Tracing such events will highlight the dependencies and derivation contributing to the progress of the research across several stages (e.g. from development to production) and milestones. Each of these phases may use of an extended set of descriptive metadata. The description of new terms should be recorded in the **C4** within the appropriate context, which is characterised by the level maturity (innovation, production) and application domain. These could be automatically extracted by the offline analysis of the lineage and prompted to the developer, who is responsible to describe their meaning and to specify the level of acceptance by a community or a group of peers (e.g. experimental, qualified).

Currently the provenance captured by the S-ProvFlow system from the execution of a method in dispel4py, is mapped to a conceptual model of provenance (MoP) that makes use of and further specialises general schemas, such as PROV and ProvOne[90]. We call such model S-PROV. This will be our starting point to address the mapping between logical representation and concrete implementation of a computational method (or workflow) and its enactment onto a set of computational resources. The model captures aspects associated with the distribution of the computation, volatile and materialised data-flow and the management of the internal state of each concrete process. Moreover, it captures changes occurring to the workflow at runtime, especially concerning dynamic steering. S-PROV is currently available as an ontology[91]. In Figure 14 we show a schematic representation of its constructs.

By identifying a general set of **Interaction Patterns**, we will further extend the model to include more concepts and capabilities of **P4**, beyond the execution of a method. We aim at recording the interactions of the research-developers and the domain scientists with the platform and the data (e.g. archival and sharing of new methods and metadata terms, annotations of new data products, etc.), in coordination with **C4**.

[90] ProveOne https://purl.dataone.org/provone-v1-dev

[91] S-PROV ontology https://github.com/aspinuso/s-provenance/blob/master/resources/s-prov-o.owl

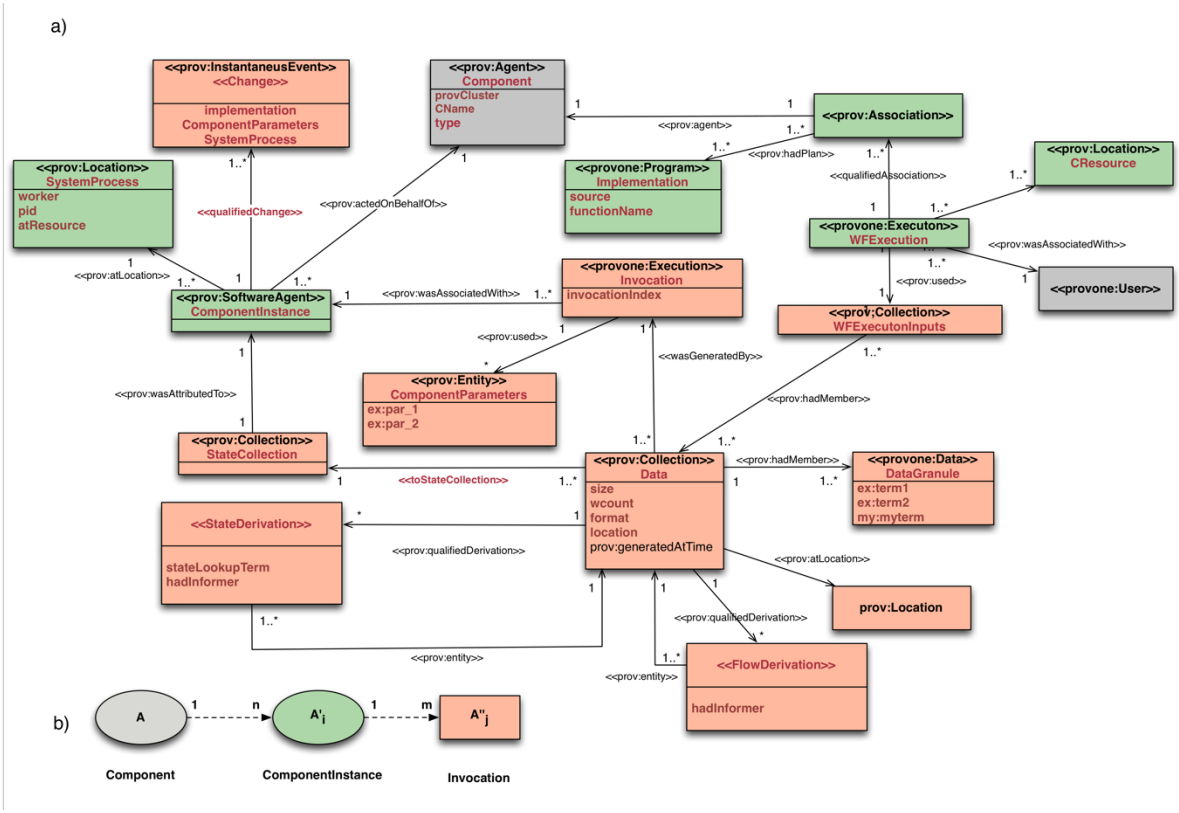

Figure 14: (a) S-PROV provenance model. Colour coding indicates the following: (grey) elements for abstract and prospective provenance; (green) concrete workflow elements and state; (red) execution elements and fine grain dependencies. Extensions of PROV and ProvONE are indicated for each class. (b) After the abstract workflow is deployed to the underlying resources, each Component is mapped into several *ComponentInstances* that perform many Invocations to execute the workflow task on the incoming data. Components and *ComponentInstances* are *prov:Agents*. They are represented as ovals throughout the chapter.

9.2.2 Storage and Access

Making provenance information representable by the combination of different, and preferably standard, ontologies and metadata schema is fundamental to support interoperability across contexts, assuring a shared understanding of the relationships among different players accessing and producing a scientific result. The use of standards in the representation extends the period of reliable interpretation. For this reason, the chosen technology and implementation need to allow a semantic-web representation for ingestion by third party services and semantic reasoning. This makes it possible to use tools and quality assurance methods that have been developed for those standards. However, practical aspects of the implementation of different access modes and use cases, with special look at retrieval performance for visualisation and rapid runtime analysis (e.g., by machine-learning (ML) algorithms), require a hybrid storage architecture and a class of services which abstract the implementation details. Figure 15 shows the current architecture of the S-ProvFlow system.

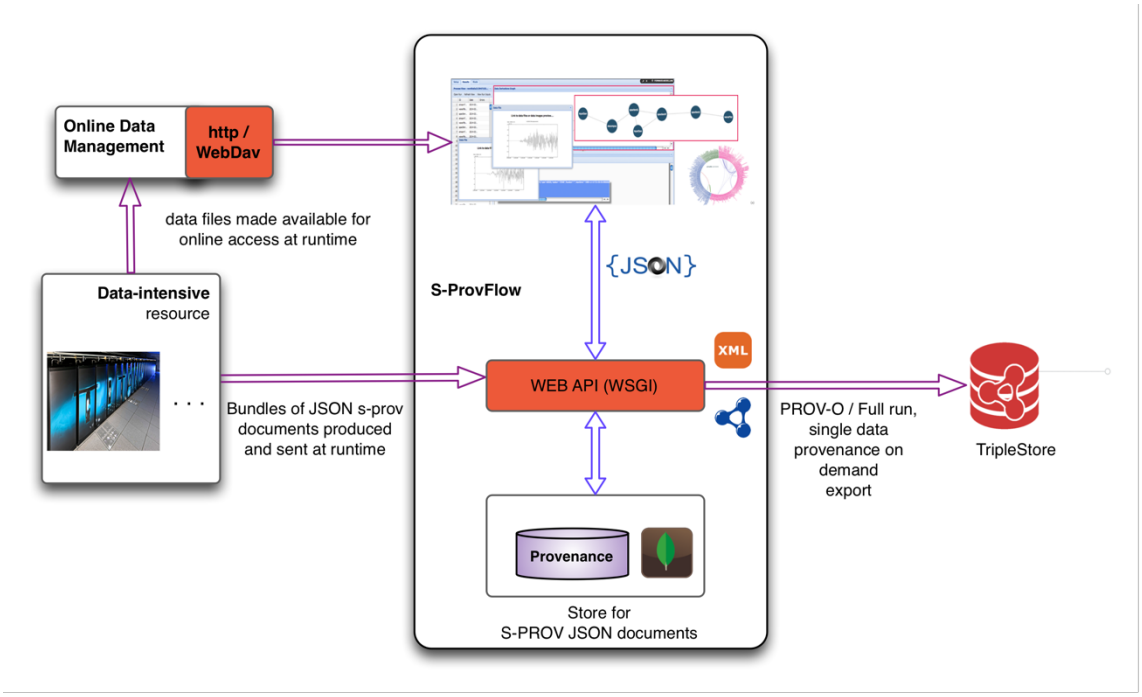

Figure 15: Schematic architecture exploiting the S-ProvFlow system for acquisition, visualisation, data access and provenance export services.

It includes a database, a webservice layer and two complementary interactive tools. The whole system is delivered as a composition of different Docker containers for an "easy" and decoupled installation of its components. The data is captured in a document-oriented database based on JSON representation (MongoDB[92]), we explored ways to benefit from the simplicity of implementation and the flexibility of integration for which such technologies are gaining momentum. On the practical side, as many systems exchange metadata over a network, the JSON format has advantages due to the wide availability of software libraries delivering efficient manipulation and visualisation. It facilitates the adoption of the same format across all layers of a system reducing the necessity for mappings and complex format conversions. Finally, thanks to a shared representation of the information, it allows the rapid implementation and validation of new features, fostering the evolution of the software involved into the different components.

To facilitate the realisation of interactive tools that exploit provenance data, the S-ProvFlow system exposes a collection of methods through a web API. Its endpoint is exposed and described according to the OpenAPI 2.0 standard. It addresses use cases such as runtime monitoring, detailed dependency exploration, visual summarisation and integrated data-discovery. The API abstracts the underlying storage technology returning data in different representation. Currently JSON is represents all results returned by the query methods. We foresee this to be further improved 1) with better support for JSON-LD and 2) by extending the current export functionalities with additional parameters for the bulk extraction of a large portion of traces from other DARE components. For instance, as depicted in Figure 15, traces can be extracted in RDF and transferred to a triple store. Extraction may be regulated by *runId*, *depth* of a derivation trace or, as further development, according to their time-stamp given a certain time-range. Therefore, it is important that all provenance concepts, including the new ones developed in DARE should be compliant and mappable to the PROV standard. This will be crucial for the communication with the actors external to DARE and to guarantee the exchange of information between the internal, yet independent DARE catalogues and control elements. Third-

[92] https://www.mongodb.com/

party access will require also a set of authorisation and visibility levels. The technology responsible for their delivery needs to be selected and tailored.

The technical implementation of the API methods may benefit from the adoption of graph databases such as those handled by Neo4j, aiming at combining flexible indexing possibilities of MongoDB with the graph traversals offered by Neo4j[93]. Moreover, use cases that require full-text searching are not currently part of the API specification and are not well supported by the underlying technology. Such queries may be useful to extract information based on less precise knowledge about the provenance properties and content. They could be implemented by combining the current document-store with other technologies, such as ElasticSearch[94], which better serve such functionalities.

The current visualisation interfaces are built on top of the API. One of the tools, the Monitoring and Validation Visualiser (MVV), assists the users in the fine-grain interpretation of the provenance records to understand dependencies; it allows them to select and configure viewpoints by specifiable searches over domain metadata value-ranges, previews, navigation of data dependency graph, within and across runs, data download and staging. It offers detailed runtime diagnostics, also differentiating between stateless and stateful operations. A graphical tool, the Bulk Dependency Visualiser (BDV) offers broader perspectives on computational characteristics as well as collaborative behaviour via customisable radial diagrams. It adopts hierarchical edge-bundle techniques and configurable grouping. It allows its users to dynamically adjust viewing and clustering controls to uncover aspects of the distribution of the processing for large single runs and data-reuse between workflow executions and users.

In DARE these tools will be further developed in parallel to the extension of the underlying provenance model and API, that will offer views on those categories of provenance relationships characterised by the aforementioned **Interaction Patterns**. Thus, enabling users and developers to re-evaluate, trace and compare their choices and decisions related to the refinement of their context and the exploitation of the platform, in single-user or collaborative settings, throughout the different sessions and phases of their research. The implementation of the model representing the interaction patterns could be implemented as a new collection in the MongoDB storage or directly mapped to and RDF/PROV representation, for instance through the adoption of the PROV-Templates[95]. A technology that could provide a starting point for the realisation of this task is the prototypical PROV-Template registration, expansion and storage service, delivered by the ENVRI+ project [Magagna *et al.* 2018a, Doron Goldfarb *et al. 2018*]. Details of this implementation need to be evaluated.

9.3 Technology for C4

An introduction to C4 as a means of developing and organising the conceptual framework for sharing information appeared as Section 4.3. This was further developed by exposing technical requirements in Section 8.3. Here we present initial thoughts on the options for developing an efficient and sustainable implementation of C4.

Logically users and software need to be able to name anything and extend anything that is not constrained by integrity or authorisation controls. RDF (aka Linked-Open Data (LOD)) has the ability to represent any potential concept or form of information as it is designed to be open, i.e., extensible in unconstrained ways. Being able to name any RDF subgraph makes the scope and extensibility feasible.

To help users cope with the complexity arising from scale and diversity, the conceptualisation structure is introduced as a knowledge organisation technology. While in principle LoD is free form, in practice, any substantial body of information needs regularity to make it constructible, understandable and usable. The *Conceptual* framework encodes that regularity without inhibiting

[93] https://neo4j.com/developer/mongodb/

[94] https://www.compose.com/articles/mongoosastic-the-power-of-mongodb-and-elasticsearch-together/

[95] https://provenance.ecs.soton.ac.uk/prov-template/

occasional divergences. We specify a substructure that must be consistently implemented to enable the C4 functionality to be developed, but extensions beyond this can emerge. Every entry has an *annotate* property, which can refer to arbitrarily complex graphs that users or software wish to use it for. This does not inhibit the introduction of other properties. When such *ad hoc* developments prove to be useful, they can be **promote**d into the proscribed framework.

A particular representation of RDF needs to be chosen, and a minimum set of arc names needs to be selected, principally from established ontologies. The integrity maintenance can depend on SHACL descriptions and mechanisms supplied by the triple store technology adopted. The knowledge base underpinning DRIP may provide useful input for these decisions [Wang *et al.* 2017].

As described in Section 4.3, C4 has the role of setting the *Conceptual* foundation that will be used by the other key components of the DARE architecture: workflow management (WaaS) and detailed provenance (P4); their relationships are outlined in Section 8.4. C4 has to clearly define "what is what" and "where is what" that is:

i) standardise a set of *Concepts* that are generic and of use for any (data) science gateway;
ii) provide a set of pointers to the definitions of such *Concepts* (and related properties and *Actions*) that will stand the test of time.

These are then used to build a re-usable common core. That is in turn further developed. Typically, it has a period of rapid inflation, during the initial tailoring extending the knowledge organisation to meet the agreed needs of a target community. These will often bring in bundles of *Concepts* with associated representations. Each discipline in a multi-disciplinary collaboration may have its established culture. When these disciplines need to collaborate the points of overlap and the gaps between pose potential communication problems. Facilitating the necessary discussion leading to agreements on what is required and what it means is essential. It is necessary to engage the relevant domain experts in this process [Trani *et al.* 2018] – domain experts for conceptual formulation, informaticians for representation and population harvesting decisions.

Existing agreed vocabularies, and ontologies are an important source to build on. This can be seen from the current use of ontologies in the EPOS-DCAT-AP as shown in Listing 1. Vocabularies in computer science have an analogous function to their use in linguistics, that is they provide definitions of concepts. Having precise definition of terms and concepts is essential to be able to state what is being under investigation by a scientist/research engineer in the context of DARE. In addition, it is also essential that every element participating in a computational workflow can be unambiguously identified (with a definition characterising it) in the provenance traces. The discussion and these agreements take a lot of high-level effort. However, they are invaluable in reducing misunderstanding and in supporting collaboration across discipline boundaries. Achieving that gain while saving on effort by importing definitions is a profitable tactic. There are many other sources. One pertinent to the current application domains is OIL-E, an ontology for systems supporting the data lifecycle in environmental research infrastructures[96]. The agreements reached by professional standardisation bodies for each domain are a major input to this process.

Listing 1: Some of the prefix to ontology mappings in use in the EPOS-DCAT-AP

```
@prefix adms: <http://www.w3.org/ns/adms#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix epos: <https://www.epos-eu.org/epos-dcat-ap#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix hydra: <http://www.w3.org/ns/hydra/core#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema: <http://schema.org/> .
```

[96]Open Information Linking for Environmental science research infrastructures (OIL-E) http://oil-e.net/ontology/

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix http: <http://www.w3.org/2006/http#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix gsp: <http://www.opengis.net/ont/geosparql#> .
```

DCAT

C4 should depend on the work of DCAT. The Data Catalogue Vocabulary (DCAT) is an ongoing standardisation effort by the W3C to improve the description in catalogues of datasets and related concepts. According to W3C:

"DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web... By using DCAT to describe datasets in data catalogs, publishers increase discoverability and enable applications easily to consume metadata from multiple catalogs. It further enables decentralized publishing of catalogs and facilitates federated dataset search across sites. Aggregated DCAT metadata can serve as a manifest file to facilitate digital preservation"

[Maali & Erickson 2014]. DCAT has been proposed for use in scientific environment. This is one of the main reasons for its use in DARE [Perego *et.al* 2016].

In the context of DARE, DCAT might not be just used as a vocabulary to facilitate the interoperability, but as the means whereby each concept (technical and non-technical) of DARE is explicitly defined or linked to a definition of the concept in another dictionary or thesaurus / ontology. This would require an extension for the concepts needed that are not already covered by DCAT or its extensions – known as Application Profiles (**AP**).

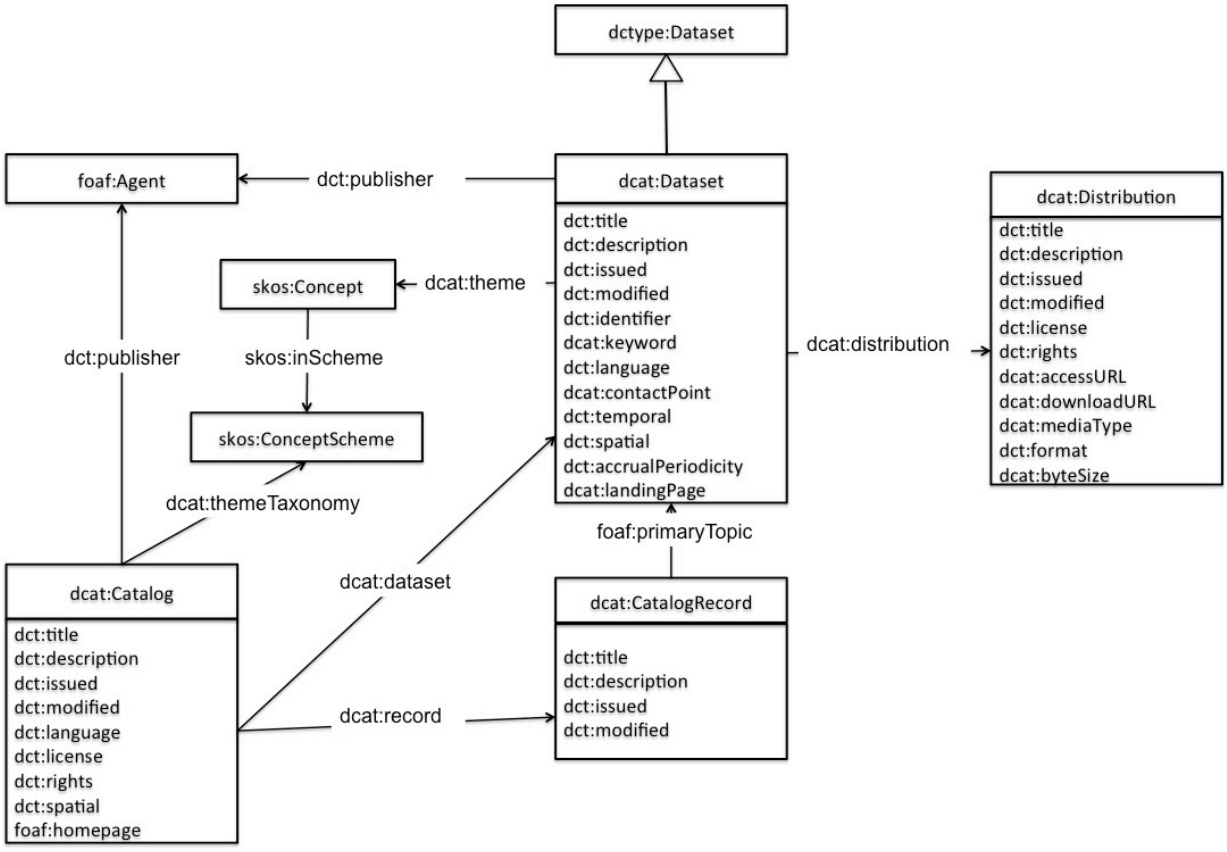

Figure 16: RDF representation for concepts in DCAT.

DCAT extensions

The European Commission has devoted effort to extend DCAT to boost the interoperability in public sector datasets in Europe via the DCAT-AP. The basic aspect to achieve the interoperability is the ability to have cross-data portal search, so that the data are better searchable across borders and sectors. This is achieved by the exchange of descriptions of datasets among data portals.

Extensions following the application profile have been realised. They match DARE's requirements and application domains:

- GeoDCAT-AP is an extension for the description of datasets that have a geospatial component which is a common type of dataset in DARE's use cases (https://joinup.ec.europa.eu/release/geodcat-ap/v101).
- StatDCAT-AP is an extension for the description of datasets that have a statistical domain such as time series which are a very common type of dataset in the DARE use cases (https://joinup.ec.europa.eu/solution/statdcat-application-profile-data-portals-europe).
- EPOS-DCAT-AP is an extension to deal with the requirements of the EPOS use case, thus very much aligned with the requirements of DARE seismological use case (https://github.com/epos-eu/EPOS-DCAT-AP).

The RDF schema of EPOS-DCAT-AP, which imports parts of GeoDCAT-AP and StatDCAT-AP is available[97].

Mapping DARE resources to catalogues and dictionaries

Given the requirements elicited in DARE deliverables D6.1 and D7.1 [Rietbrock *et al.* 2018, Pagé & Spinuso 2018] and the content in Section 7, we have identified the following mapping between the categories of concepts described above and a set of additional concepts that are essential to further specify the macro concept. For each concept the catalogue technology and vocabularies (containing the definitions) are provided.

Users (management): concepts such as name, surname, organisation, address, group, research discipline are needed for the definition of the user executing computation on DARE. EPOS-DCAT-AP has extensions for Person, Organisation, Project. The schema provided is based on the definitions in the FOAF vocabulary (http://xmlns.com/foaf/spec/). See the *Person Concept* in Section 8.3.2 for a proposal. This can be specialised to deliver variations required by communities, so it is deliberately minimal.

Research Group is another concept to keep track of and the FOAF contains a definition for that. See the *Group Concept* in Section 8.3.2 for an initial version. This can be specialised to deliver variations required by communities, so it is deliberately minimal.

Data: DCAT and DCAT-AP are all focused on this concept. The extensions of DCAT such as GeoDCAT-AP and StatDCAT-AP are definitely technologies to be used since they will make data better characterised. See the *DataItem Concept* in Section 8.3.2 for a basic version. This can be specialised to deliver variations required by parts of the DARE platform and communities, so it is deliberately minimal.

Model data are another type of data that DARE has to use. For this type of input a representation via DCAT-AP resources is required. This will be a specialisation of *DataItem* when the required properties and *Actions* are clarified.

Computation: for this essential concept in C4 a catalogue defining which parameters are needed can be developed for the purpose. A set of attributes would be required based on the information needed by the WaaS and the P4 elements. Currently we foresee the following attributes: identifier, has version, location, requires, publisher, rights, description. These terms can all be found, with the related definition, in the Dublin Core Metadata Initiative Metadata Terms [Weibel & Koch 2000]. The definition of *Concept Thing*, from which all other *Concepts* inherit, is intended to introduce the

[97] EPOS-DCAT-AP https://github.com/epos-eu/EPOS-DCAT-AP/tree/EPOS-DCAT-AP-shapes

minimum standard of such attributes for DARE platforms – see Section 8.3.2. It will be adjusted in the light of experience and the impact will propagate to all *Concepts* and therefore to all entries, i.e., instances of *Concepts*. The *Action* and *Run Concepts* currently deal with major aspects of Computation. However, a *SoftwareItem Concept* and its specialisations will also be needed. These will be developed, starting from the outline in Section 8.3.2.

The choice of storage system or database system to support the C4 content and operations needs to accommodate all of the operations set out in Section 8.3.1. This clearly needs to be a suitable triple store, with transactional properties to allow correlated updates to be made consistently and with good indexing, querying and bulk operation support to deliver required performance for anticipated intensive and concurrent loads. It does not handle the bulk data that DARE enables practitioners to work with. However, it is key to the construction, tailoring and use of DARE platforms and must not become a bottle neck nor point of failure. Experiments investigating how candidate triple stores handle this requirement will be conducted during the next year. It is anticipated that substantial tuning will be needed to ensure C4 supports the DARE platforms well.

Triple store resources

The C4 catalogue will grow large, as it accommodates more complexity, more workers, more *Sessions* and is used more intensively by the rest of the DARE platform and by external systems. Research campaigns, communities and groups all have significant longevity. If C4 is serving their research, their development and their collaboration it will hold very large numbers, maybe millions, of entries with a know knowledge structuring based on hundreds to thousands of *Concepts*. Whilst we are unlikely to encounter these scales in the DARE project itself, we must plan for them. Otherwise, we leave our application domain colleagues with an impending disaster; as they get more committed and use the DARE platform more intensively, it will reach its capacity limits. Existing trip[le stores may have the capacity, but will they have the operational performance needed as well. That is, the rates at which they can simultaneously serve many queries, while supporting high update rates? DARE will try virtuoso[98] initially. We will investigate this and compare it with alternatives. We are aware that there is significant recent work on graph databases and on automated parallelisation of graph algorithms, e.g., Grape [Fan *et al.* 2017a, b, 2016, 2015]. Grape or a similar system could provide an escape route if triple-store performance starts to limit DARE platform performance – this will depend on the nature of the loads, being explored through two independent and significantly different application areas, as well as the mapping and organisation of the information required. Both will need investigation and characterisation during the coming 12 months. We note that many loads on triple stores are dominated by highly parallel query requests often coupled directly with users and hence limited by their rate of working. The updates may be more managed by curational processes. However, in DARE the update rates will be driven by WaaS enacting workflows concurrently, each of which may need thousands of accesses and updates to the triple store during their execution.

9.4 Contemporary data-intensive architectures

Specific related R&D that DARE should be influenced by has been identified in the preceding Sections. We present here broader spectrum contemporary work. In the long-run the DARE platforms will need to build on or draw in relevant selections from such work.

9.4.1 Architectures for Big Data processing

The panorama of architectures for processing big data is broad and in recent years substantial effort from business and scientific communities has been invested in this topic.

As in any modern architecture for distributed data processing that provides interaction with remote users, the main corner stone to base the computational framework is the use of web services. This will be the mode of interaction between DARE and other systems [Austin *et al*. 2004, Kendall *et al*.

[98] The virtuoso triple store backed by an RDBMS https://github.com/big-data-europe/README/wiki/Virtuoso

1994]. The microservice paradigm should influence DARE [Dragoni *et al*. 2017]. Employing small atomic functionalities specialised for well-defined tasks yields resilience and sustainability. However, the architecture must present their integration as a coherent whole to users of a production environment. In contrast, research developers need to exploit the potential of new ways of composing microservices.

For built-in and user-defined actions WaaS will need to allocate work to resources and services dynamically. This will require it to find suitable services or deploy them on hosting Clouds and institutional facilities. WaaS takes responsibility for this deployment and then the orchestration of the enactment. Instances of the DARE platform will also perform its actions as parts of workflows run by other systems. To achieve these functions, the DARE architecture will build on the OASIS consortium standard (Topology and Orchestration Specification for Cloud Applications) to establish the topological properties and service orchestration deployed across cloud services [TOSCA WG 2017]. DARE will draw on the EU MiCADO project TOSCA-handling tools [Kiss *et al.* 2017] to help developers, managers and even users develop the descriptions in TOSCA that are needed for WaaS to operate and for DARE to be used via TOSCA deployment and orchestration.

Business, particularly entertainment, games and on-line/business-to-business transactions, dominate the data-handling and processing digital ecosystem. Consequently, design patterns for scientific applications and the architectures that support them must take account of the resultant technological and business trends. Therefore, articles presenting big-data-processing architectures for business and science should be considered. For example, Pääkkönen *et al.* [2015] provide a good summary paper focused on the comparison between architectures. They give special attention to companies that have big data processing as a core of their operations e.g., Facebook, LinkedIn and Twitter. They provide a reference architecture synthesised from the works surveyed. The DARE should review the requirements and goals in the light of this paper. The paper shows how architectures were mapped by the implementations in the businesses. Nadal *et al.* [2017] present a set of architectures to address the 5V (Volume, Velocity, Variety, Variability and Veracity) challenges and the integration of semantic aware layer. DARE should look at the semantic considerations of the paper and compare to the approach taken by using C4. The Lambda architecture [Marz & Warren 2015] is used in industry for addressing big data computations; however, it falls short with respect to variability and veracity. The solution proposed in [Marz & Warren 2015] is based on the combination of Lambda and semantic web architectures and it is called Bolster. Bolster[99] provides a framework for software evolution and adaptation for data-intensive applications. DARE might reuse of some parts (e.g., the speed, batch or semantic layer) of Bolster. Demchenko *et al*. [2014] define a Big Data Architecture Framework (BDAF) addressing all aspects of the big data ecosystem. They define key components: Big Data Infrastructure, Big Data Analytics, Data structures and models, Big Data Lifecycle Management and Big Data Security. Similar components are required and delivered in the DARE framework by BDE (see Section 3.2) [Auer *et al.* 2017]. Demchenko *et al*. claim that in the future the whole data lifecycle should be taken into account, as it is in the ENVRI reference model[100]. Agents have also been proposed to be key components in architecture to process big data [Twardowski *et al*. 2014]. We believe that agents are effective technologies to handle interactions between components, however we do not envision DARE to have a specific agent-

[99] https://gessi.upc.edu/en/projects/supersede-h2020

[100] https://wiki.envri.eu/display/EC/ENVRI+Reference+Model

based approach given the amount of data DARE aspires to handle, since agents generally work well in low-volume data-exchange environments to allow for interactions (e.g., FIPA-ACL[101]).

9.4.2 Standardisation for data and metadata

In addition to processing large volumes of data, DARE has the challenges of dealing with complex geospatial data. Gathering and processing the spatial data for a platform that is open to the researchers worldwide has either to support the data types of every user or it has to rely on data and metadata standardisation. This is developed by relevant standardisation bodies in the (geo)scientific world. These standards are intended to make it easier for researchers and their software to interoperate. That benefit depends on wide adoption of the same standards. For this purpose, the W3C spatial working group has defined some best practices [W3C-SP WG 2017]. It is in close contact with the Open Geospatial Consortium (OGC) [Lupp 2008]; the EU is also encouraging the standardisation process of spatial information via the INSPIRE Directive. However, many decisions pertinent to governmental data may not carry over, e.g., have appropriate detail and precision, to scientific applications.

Generally, the processing of data requires metadata accompanying the data to let the system or scientists understand the characteristics (e.g., units of measurement) or conventions (e.g., coordinate system, fill values) applied to the data. Although not all the business sectors are aligned with providing good and complete metadata and/or data dictionaries, the scientific communities that have the need to share information and knowledge have adopted standards. The use case communities in DARE, i.e., the Climate and Seismology communities have well-defined and widely adopted standards. For example, the Climate community rely on the WMO Core Profile of the ISO 19115: Geographic Information – metadata; the seismologists draw on international standards, such as those developed by the FDSN and ORFEUS.

In the case of extension of the DARE platform to other domains, a standardisation effort for data and metadata would be needed. The W3C efforts around Data Catalogue Vocabulary (see Section 9.3 for more details) and Research Data Alliance Metadata Discovery Directory [RDA-MS WG 2016] provide contexts to facilitate this.

9.4.3 Non-traditional user interaction

The work of a scientist has been traditionally behind a workstation or interfacing with an HPC system via a console. Nowadays, these ways of interacting with systems and computation are being augmented via smart mobile devices, processors in instruments and IoT integration facilities. DARE should not neglect these trends and look for technologies and ways to build its platform so that it can seamlessly support different interaction modalities. Frameworks that allow easy cross-platform development have been proposed by standardisation bodies [W3C-TAG WG 2006] and in the literature [Raj & Tolety 2012].

10 Guidelines for DARE development phase

Here we identify the scope and interdependencies of work to be undertaken by work packages during the next 12 months. It is crucial that those undertaking DARE R&D find the architecture helpful rather than over-constraining. Here we provide notes of an initial analysis of the implications of the architecture for each work packages in consultation with work package leaders (WPLs).

[101] http://www.fipa.org/repository/aclspecs.html

10.1 WP1: Project management

WP1 has arranged that the propose architecture and each stage of Task 2.1 4-monthly deliverables has been reviewed at the plenary meetings. This will continue during the next 12 months plenaries. WP1, through its technical director, will ensure that architectural recommendations are considered and that issues encountered are fed back into the next evolution of the architecture. In particular, it will decide on the balance between exploring the architectural potential and delivering operational platforms.

10.2 WP2: Architecture Specification and Innovation

WP2 will ensure that the architecture is effectively communicated. There are two mechanisms:

1. The monthly ArchTF meetings[102]. These consider and refine ideas developed in each iteration of the architecture taking all viewpoints and stakeholders into account.
2. Communication to the project and beyond, by presentations at face-to-face DARE meetings, by (internal) deliverables, and within the next 12 months, papers and presentations at relevant events (see Section 10.8).

WP2 will develop deeper understanding and greater clarity about the interaction between the architecture and today's digital technology and methods. This is exemplified by the recent two-day meeting at KNMI[103]. It will be continued with sufficient intensity through half-day telcos and a face-to-face meeting at INGV in January 2019 to influence the next phase of ATF sprints, preparing material for our user communities. This in turn will expose issues with the architecture.

It will continue to evolve with a 4-monthly refinement cycle for the next year. Its interaction with aspects of DARE, such as the operational development platforms and the developing platform API will be tightened.

Liaison with ENVRI and relevant working groups in RDA, W3C and the global science gateway and scientific workflows R&D communities will help keep WP2 well informed and provide opportunities for DARE outreach.

10.3 WP3: Large-scale Lineage and Process Management

WP3 is already very actively engaged in key issues through Task 3.1. This revealed and clarified the precise needs of the two user communities and the impact of those needs on the other WPs. This will be further refined in the coming 12 months. The work started with the identification of the community stories described in ID 3.1, according to Agile guidelines. It will proceed together with WP2 and WP4 to mapping prioritised user stories to **Feature** and **Capabilities** that characterise the DARE platform, in line with its architectural principles – Figure 17 shows an example of this mapping. This approach should facilitate the identification of the **Technical Stories** contributing to the implementation of DARE and will make sure we communicate our achievements to WP6 and WP7, by directly addressing the targets revealed by their user-stories.

[102] Records of WP2 Meetings are collected here https://drive.google.com/open?id=1B143S-jcVGGuqBKqDnhscoqG-bgK8zOM there were 13 in the first 12 months.

[103] Architecture meets Technology meeting 26&27 Nov. 18 https://drive.google.com/open?id=1tdtpIU7rzMUOBTWe3UvZAnOjrzu6MM7o

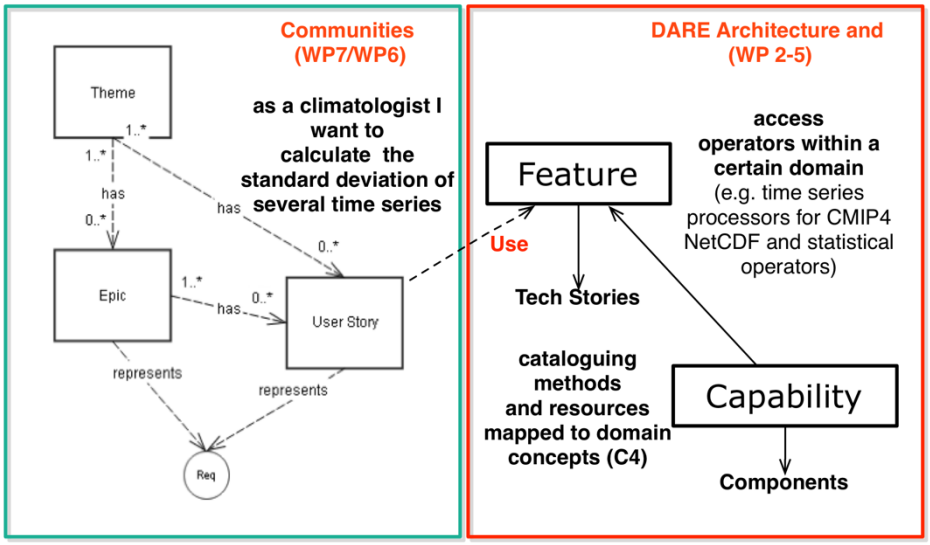

Figure 17: Mapping from Community User Stories to DARE Features and Capabilities. The user-story requires to access the right implementation of a component (Feature), which may be implemented through resolving services (Tech Story). Cataloguing is a capability of DARE, which will affect different conceptual and technical components constituting C4.

Given this framework, WP3 will then focus on contributing to the following Capabilities:

- **Provenance Acquisition and Characterisation (P4):** collection of and access of lineage produced by the methods at runtime. This should be integrated, in cooperation with WP2, with provenance information concerning the interactions characterising their evolving working contexts (C4). This will lead to *pervasive* provenance information. Issues inhibiting provenance becoming *pervasive* should be discussed with WPs 1, 8 and 9. Limits to its *persistent* nature need to be discussed with WPs 5 and 2.
- **Process Control (WaaS):** investigation (in collaboration with WP4) of the preferred notations and API for describing actions to WaaS that allow domain experts to accurately define and refine their intent in ways which are independent of platform technologies, so that the interpretation of their requests for action, monitoring and control of workspaces and workflows remains consistent as platforms evolve and as different targets are chosen.
- **Provenance Exploitation (P4):** Development and demonstration of provenance-driven tools and API to facilitate domain-oriented innovation and production. This will clarify provenance-trace requirements in order to support the tools and their use by domain practitioners. It will begin to demonstrate the power of a provenance-driven approach to researchers and innovators and thereby expose issues to be addressed in the hyper-platform's design and structure.
- **Application Development:** Exploration (in collaboration with WP4) of the required libraries needed for the user communities, for large-scale data processing and for integration and analysis of multi-source data.

Participation in ATFs that develop cross-cutting solutions will be helpful in cutting through technological and organisational boundaries, thereby demonstrating the strength of the current approach and revealing weaknesses.

10.4 WP4: Big Data Processing and Analytics

The main goal of the work package is to successfully provide deployments of the DARE platform that meet the requirements and use cases defined by WPs 6 and 7, and have the potential to extend in scale and performance as those communities' ambitions increase.

Having a carefully selected an initial set of technologies at its foundation, WP4 will adopt a bottom-up approach, ensuring that these technologies are integrated at the technical level and readily available for experimentation.

The main axes of activity are the following:

- Calibration of the infrastructural and deployment requirements of the platform, in collaboration with WP5. This includes the establishment of access mechanisms for specialised infrastructures and HPC resources that are potentially used during use case execution;
- Reification of the schemas and external semantic resources that are desirable/required to be accessible and used by C4, in collaboration with user partners;
- Refinements on the notations for describing DARE concepts and particularly actions and their links to programmatic implementations;
- Modifications on the existing technologies to adapt to the conceptual changes described above;
- Establishment of measurement mechanisms for the KPIs related to each use case, that will assess the benefits of using the DARE platform.

The purpose of the aforementioned activities at this stage is to lead to a concrete and as complete as possible demonstrative execution of use cases and tasks as defined in WP3. It is envisioned that relatively short, agile cycles of user feedback and technology updates will take place. Prioritisation will depend on technology readiness and the criticality of the update towards covering the entirety of the test cases.

10.5 WP5: Platform Operation and Maintenance

This WP will be developing an operational platform in which the proposed components of DARE can be developed and tested. It will be investigating how those components are best configured and distributed so that they combine to deliver an integrated experience for users of the DARE hyper-platform. WP5 will also be exploring how it can contribute from available platforms and systems elements of the DARE architecture.

This leads to the following activities:

- Clarification (in dialogue with WP4) of the platform properties they depend on and the extent to which they can accommodate variation in these properties. This will include access to specialised architectures and HPC resources, and how this relates to current and anticipated cloud provision.
- Preliminary integrations of services to provide an initial context for DARE development.
- Investigation of notations for describing such integrations and automating their deployment, e.g., high-level front ends to TOSCA.
- Investigation as to how existing catalogue services and their contents / operational procedures can help deliver DARE's requirement for C4.
- Investigation of strategies to make the existing WaaS services, e.g., dispel4py as used in the VERCE portal, more scalable and with better recovery from partial failures.
- Investigation of how to deliver security and preservation where DARE services require it, e.g., for P4 data.

The actual work will deal with meeting ATF and other development requirements first, so as not to introduce delays. However, some consideration to the more strategic platform issues should be given when possible.

10.6 WP6: EPOS Use Case

The primary goal of WP6 is to ensure that the other WPs fully understand their current working practices and technologies, the new capabilities they require and their priorities (the order in which they would lie additional capabilities if possible).

This leads to the following activities:

- Working mainly through T3.1 ensure that their requirements and constraints are explicitly recorded, understood and appropriately decomposed.
- Working in an ATF with other WPs, build demonstrators that test and expand that understanding.
- In conjunction with WPs 3 and 4, begin to develop a framework for testing solutions and measuring their scalability.

10.7 WP7: IS-ENES/Climate4Impact Use Case

WP7 will work with both bottom-up and top-bottom approaches. Requirements and constraints are well known, from experiences within previous European projects and initiatives (EUDAT&EUDAT2, IS-ENES&IS-ENES2, CLIPC, CIRCLE2), National projects and International Assessments. From these, one can also define components, platforms, standards and services that could be useful.

One of the primary goals of WP7 is to ensure that the other WPs fully understand their current working practices and technologies, the new capabilities they require and their priorities (the order in which they would lie additional capabilities if possible).

This leads to the following activities:

- Working through T3.1 ensure that their requirements and constraints are explicitly recorded, understood and appropriately decomposed. This impacts on several aspects of the Use Cases, but the DARE API can be put on top of the list, as it will be important that it can support not only the IS-ENES RI (Climate4Impact Platform), but also direct access like Jupyter notebooks for example, for ease of integration with the current scientific workflows.
- Working in an ATF with other WPs, build demonstrators that test and expand that understanding.
- Investigating the use of EUDAT technologies as top or middle layer in the DARE platform. It should consider the GEF prototype service, B2DROP/B2SHARE, B2ACCESS, and other B2 Services. Adding such a layer would promote an interoperable generic layer with easier integration with the future EOSC.
- Interfacing the DARE Platform with the Climate Community UI of the Climate4Impact Platform.
- Developing the Use Case backend by integrating calculation tools, e.g. icclim, within the dispel4py environment.
- Leveraging current services, platforms and tools to be fully “provenance-aware”, by providing required information to the DARE provenance components.

In conjunction with WPs 3 and 4, begin to develop a framework for testing offered solutions and measuring their scalability.

10.8 WP8: Innovation Management and Dissemination

WP8 will conduct outreach regarding the DARE architecture and the professional research collaboration models that motivate it. This has already begun with presentations and the

organisation of a session at the American Geophysics Union (AGU)[104]. It will continue at the European Geophysical Union (EGU) and at workshops and conferences organised by our two collaborating disciplines. The goal is to disseminate progress and a vision of the eventual DARE platform and the modes of working it will support well. Comments and suggestions will be gathered using questionnaires or a structured interview. The intention is to include a broad spectrum of practitioners, potential users, domain experts and research developers. Analysis of this feedback will inform further R&D.

Traditional dissemination through publications will stimulate further review of our approach and alert the wider community of research infrastructure architects to its potential. That dissemination and professional feedback will also be pursued through our active participation in RDA and W3C specialist groups.

The newsletter and web site will draw attention to these and deliver further insights. Internal training and feedback gathering relating to the DARE architecture, its application and use will be included in DARE face-to-face events to expand the high-level dissemination and review undertaken via the ArchTF into more practical and detailed forms.

10.9 WP9: Ethics requirements

The collection of provenance records and other information on the DARE platform, e.g., *Person* instances – see Section 8.3.2 – ineluctably requires personal data. During early R&D investigations, this can be dummy information or information pertaining to the members of ATFs who have given fully informed consent. In such R&D contexts proper protection of personal data may be infeasible but it only close colleagues who may access it. In due course, DARE should engage with others, to clarify the legal recognition of this R&D context and the scope of the data and activity covered. To some extent, this will overlap with clarifications on the arrangements for authentication and authorisation data during log-in processes.

The quality of science and the progress of careers depends on proper recognition of contributions. The handling of collaborative processes depends on identifying others who have contributed in some way to the information and methods being used. The maintenance of standards requires attribution of culpability. All of this is supposed to be well supported by DARE's provenance record keeping – see Section 7.3. DARE needs to develop an understanding, in conjunction with the rest of the research community. As to how these research requirements should be handled in production systems, as part of preparing the DARE platform for such uses. It may impose technical and procedural requirements on what we do.

10.10 Integrative effort

Due to resource limitations, each WP will only be able to partially fulfil these envisaged activities. The WP leaders in conjunction with management from WP1, will steer their available resources to address their most critical issues. WP1 will influence these choices so parts fit well together. This integration will be demonstrated in one or more operational exemplar systems. The architectural team (WP2) will prioritise resolving issues arising from the integration of separate parts.

11 Summary and Conclusions

During its first year the DARE project has developed significant advances in understanding and shaping the required architecture for ambitious platforms that support the new wave of data-powered research on the emerging higher capacity e-Infrastructures. We have set the direction for addressing all three of DARE's challenges:

1. Handling ever more demanding (5 Vs) bodies of data as research and decision making enjoys our expanding wealth of accessible digital information.

[104] AGU 2018 session https://agu.confex.com/agu/fm18/prelim.cgi/Session/50145

2. Planning for extremes of computation that may require partitioning across specialised parts of the e-Infrastructure.
3. Providing knowledge-organisation aids to help individual experts and multi-disciplinary collaborating teams cope well with the growing complexity of their information space.

This will work well for application domain professionals as they are presented with a coherent abstraction that will be mapped in semantic-preserving ways to new technologies as the digital ecosystem evolves. It will also work well for research developers as it provides a framework to help them. It is designed to incrementally deliver the benefits and power of the DARE platform to diverse application communities.

The challenges of delivering such an advanced platform in a sustainable way are addressed by partitioning the platform into three interdependent but tractable parts:

1. The *Workflows-as-a-Service* (WaaS) partition deals with allowing developers and application domain professionals express their requirements in familiar ways but with a greater abstraction from technical detail than has been attempted previously. It organises their work by mapping to the latest forms of data-intensive technology.
2. The *Protected Persistent Pervasive Provenance* (P4) subsystem delivers a sophisticated and controllable provenance collection and interpretation service enabling provenance-powered tools and methods.
3. The *Common Conceptual Core Catalogue* (C4) acts as an information exchange between humans and the platform, helping users develop and share a common understanding through which they express their intent, and helping the other two partitions interpret their intent in semantic-preserving but efficient ways. It encourages communication and control based on intuitive names and agreed vocabularies.

During the next 12 months DARE will explore these ideas through prototypes that investigate critical aspects of the architecture: its feasibility, understandability, maintainability and performance. The two user communities offer significantly different patterns of use and a variety of pinch-points that should help us determine the extent to which a common core and framework can be tailored to be convenient and attractive to diverse communities. This is a crucial architectural issue.

Modern data-powered science and decision making will become dependent on hyper-platforms such those delivered by DARE, to achieve and support their professional practices, to meet new and more demanding goals and to collaborate by sharing information, methods and resources to harness diverse viewpoints and many skills. When tackling society's pressing challenges, they will also be geographically dispersed with many different legal and organisational contexts. Resources and skills brought to them by research projects such as DARE cannot be afforded indefinitely. This is not just a matter of cost. There simply aren't enough digital-system, software engineering and data-science experts to meet the existing demand, and *that demand is growing*. Consequently, solutions and their support have to be shared as far as possible. This is already happening at some levels, e.g., in the provision of the cloud services. But sharing maintenance, support and R&D for the levels above that is uncharted waters. The DARE platform is an exemplar of the trade-off between a common core with adaptation for specific application-communities. Those application-specific adaptations in the longer term have to be developed and maintained by the communities they serve. However, the common core that has widely amortised support should raise the level so that the remaining *specialist maintenance and support is affordable*.

References

[Ackerman *et al.* 2013] M.S. Ackerman, J. Dachtera, V. Pipek and V. Wulf, *Sharing knowledge and expertise: The CSCW view of knowledge management*, Computer Supported Co-operative Work, 22, 531-573, 2013.

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

[Armstrong *et al.* 2014] T. G. Armstrong, J. M. Wozniak, M. Wilde, I. T. Foster, *Compiler techniques for massively scalable implicit task parallelism*, in: International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2014, New Orleans, LA, USA, November 16-21, 2014, 2014, pp. 299-310.

[Atkinson *et al.* 2013] M.P. Atkinson, R. Baxter, P. Brezany, O. Corcho, M. Galea, M. Parsons, D. Snelling and J. van Hemert, *The DATA Bonanza: Improving Knowledge Discovery in Science, Engineering and Business,* Wiley 2013.

[Atkinson *et al.* 2015] M.P. Atkinson, M. Carpené, E. Casarotti, S. Claus, R. Filgueira, A. Frank, M. Galea, T. Garth, A. Gemünd, H. Igel, I. Klampanos, A. Krause, L. Krischer, S. H. Leong, F. Magnoni, J. Matser, A. Michelini, A. Rietbrock, H. Schwichtenberg, A. Spinuso, and J.-P. Vilotte, *VERCE delivers a productive e-Science environment for seismology research*, in Proc. IEEE eScience 2015.

[Atkinson *et al.* 2016] Atkinson, M.P.; Hardisty, A.; Filgueira, R.; Alexandru, C.; Vermeulen, A.; Jeffery, K.; Loubrieu, T., Candela, L., Magagna, B., Martin, P., Chen, Y., Hellström, M. *A consistent characterisation of existing and planned RIs*. ENVRIplus deliverable D5.1, 2016. http://www.envriplus.eu/wp-content/uploads/2016/06/A-consistent-characterisation-of-RIs.pdf

[Atkinson *et al.* 2017] M.P. Atkinson, S. Gesing, J. Montagnat and I. Taylor, *Scientific Workflows: Past, Present and Future*, Future Generation Computer Systems, 75, 216–227, 2017.

[Atkinson *et al.* 2018] M.P. Atkinson, R. Filgueira, A. Spinuso, L. Trani *et al.* Download considered harmful – provokes – a flexible federation framework, DARE technical Report, 2018.

[Atkinson *et al.* 2018a] M.P. Atkinson, E. Casarotti, M. Ewering, R. Filgueira, A. Gemünd, I. Klampanos, A. Krause, F. Magnoni, A. Pagani, C. Pagé, A. Rietbrock and A. Spinuso, *DARE Architecture and Technical positioning*, DARE ID2.1-M4, April 2018, URL https://drive.google.com/open?id=1By8uy2Mdxj7S4OKxLuMNCllvhY7GoH0R

[Atkinson *et al.* 2018b] M.P. Atkinson, E. Casarotti, M. Ewering, R. Filgueira, A. Gemünd, I. Klampanos, A. Krause, F. Magnoni, A. Pagani, C. Pagé, A. Rietbrock, A. Spinuso and C. Wood, *DARE Architecture and Technical positioning*, DARE ID2.1-M8, October 2018, URL https://drive.google.com/open?id=1kqsVl-x3ZbgT68ifl1G9zK0XLJXjjkcO

[Auer *et al.* 2017] Sören Auer, Simon Scerri, Aad Versteden, Erika Pauwels, Angelos Charalambidis, Stasinos Konstantopoulos, Jens Lehmann, Hajira Jabeen, Ivan Ermilov, Gezim Sejdiu, Andreas Ikonomopoulos, Spyros Andronopoulos, Mandy Vlachogiannis, Charalambos Pappas, Athanasios Davettas, Iraklis A. Klampanos, Efstathios Grigoropoulos, Vangelis Karkaletsis, Victor de Boer, Ronald Siebes, Mohamed Nadjib Mami, Sergio Albani, Michele Lazzarini, Paulo Nunes, Emanuele Angiuli, Nikiforos Pittaras, George Giannakopoulos, Giorgos Argyriou, George Stamoulis, George Papadakis, Manolis Koubarakis, Pythagoras Karampiperis, Axel-Cyrille Ngonga Ngomo, Maria-Esther Vidal, *The BigDataEurope Platform – Supporting the Variety Dimension of Big Data*, in Proceedings of 17th International Conference on Web Engineering (ICWE 2017), LNCS, DOI 10.1007/978-3-319-60131-1_3. https://www.researchgate.net/publication/312597089_The_BigDataEurope_Platform_-_Supporting_the_Variety_Dimension_of_Big_Data [accessed Apr 19 2018].

[Austin et al. 2004] D. Austin, A. Barbir, C. Ferris, S. Garg, Web Services Architecture Requirements, W3C Working Group Note, 11 February 2004

[Bell-Burnell 2017] Dame Jocelyn Bell-Burnell, *50 years observing Pulsars* Royal Society of Edinburgh president's lecture https://www.rse.org.uk/event/fifty-years-of-pulsars-pulasting-radio-stars/ 2017.

[CF] CF (Climate and Forecast), http://cfconventions.org/

[Charbidis *et al.* 2015] Angelos Charalambidis, Antonis Troumpoukis and Stasinos Konstantopoulos, *SemaGrow: optimizing federated SPARQL queries*, in proceedings SEMANTiCS'15, ACM 2015. DOI: http://dx.doi.org/10.1145/2814864.2814886

[Chronis *et al.* 2016] Yannis Chronis, Yannis Foufoulas, Vaggelis Nikolopoulos, Alexandros Papadopoulos, Lefteris Stamatogiannakis, Christoforos Svingos and Yannis Ioannidis, *A Relational Approach to Complex Dataflows*, in Proceedings of the EDBT/ICDT 2016 Joint Conference.

[Constantin et al. 2018a] Constantin, A., Nieva, A., Hardisty, A., Atkinson, M. 2018. Using Persona as Lenses for a Reference Model. Submitted to BCS HCI'18

[Constantin et al. 2018b] Constantin, A., Nieva, A., Hardisty, A., Atkinson, M. 2018. Improving Knowledge and Expertise Sharing for Research Infrastructure Communities. Submitted to CSCW'18

[Deelman *et al.* 2017] Ewa Deelman *et al*., *Pegasus' role in gravitational wave detection* https://pegasus.isi.edu/tag/ligo/ 2017.

[Demchenko *et al.* 2014] Y. Demchenko, C. de Laat and P. Membrey, "Defining architecture components of the Big Data Ecosystem," *2014 International Conference on Collaboration Technologies and Systems (CTS)*, Minneapolis, MN, 2014, pp. 104-112. doi: 10.1109/CTS.2014.6867550

[Dragoni *et al.* 2017] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In Present and Ulterior Software Engineering (pp. 195-216). Springer.

[Dreger *et al.* 2005] D. S. Dreger, L. Gee, P. Lombard, M. H. Murray, & B. Romanowicz, 2005. Rapid finite-source analysis and near-fault strong ground motions: Application to the 2003 Mw 6.5 San Simeon and 2004 Mw 6.0 Parkfield earthquakes. Seismol. Res. Lett., 76(1), 40–48.

[Fan *et al.* 2017a] W. Fan, J. Xu, Y. Wu, W. Yu, J. Jiang, Z. Zheng, B. Zhang, Y. Cao, C. Tian. *Parallelizing sequential graph computations*, SIGMOD 2017 best paper

[Fan *et al.* 2017b] W. Fan, C. Hu, C. Tian. *Bounded incremental graph computations: Undoable and doable*, SIGMOD 2017.

[Fan *et al.* 2016] W. Fan, X. Wang, Y. Wu, and J. Xu. *Adding counting quantifiers to graph patterns. SIGMOD 2016.*

[Fan *et al.* 2015] W. Fan, X. Wang, Y. Wu, and J. Xu. *Association rules with graph patterns*, VLDB 2015.

[Filgueira *et al.* 2015] R. Filgueira, A. Krause, M. Atkinson, I. Klampanos, A. Spinuso and S. Sanchez-Exposito, *dispel4py: An Agile Framework for Data-Intensive eScience*, 2015 IEEE 11th International Conference on e-Science, Munich, 2015, pp. 454-464. doi: 10.1109/eScience.2015.40

[Filgueira *et al.* 2016a] Filgueira Vicente, R., Krause, A., Atkinson, M., Klampanos, I. & Moreno, A., *dispel4py: A Python Framework for Data-Intensive Scientific Computing,* International Journal of High Performance Computing Applications. 31, 4, p. 316-334, 2016.

[Filgueira *et al.* 2016b] *F*ilgueira, Rosa, Rafael Ferreira da Silva, Amrey Krause, Ewa Deelman and Malcolm P. Atkinson. *"Asterism: Pegasus and Dispel4py Hybrid Workflows for Data-Intensive Science."* In Proc. 2016 Seventh International Workshop on Data-Intensive Computing in the Clouds (DataCloud) (2016): 1-8.

[Gorijo *et al.* 2017] D. Garijo, Y. Gil and O. Corcho, *Abstract, Link, Publish, Exploit: An end-to-end framework for workflow sharing*, Future Gener. Comput. Syst. **75**, 271-283, 2017.

[Hellström *et al.* 2017] M. Hellström, M. Lassi, A. Vermeulen, R. Huber, M. Stocker, F. Toussaint, M. Atkinson and M. Fiebig: *A system design for data identifier and citation services for environmental RIs projects to prepare an ENVRIPLUS strategy to negotiate with external organisations*. ENVRIplus Deliverable D6.1, submitted on January 31, 2017. Available at http://www.envriplus.eu/wp-content/uploads/2015/08/D6.1-A-system-design-for-data-identifier-and-citation-services-for-environmental-RIs.pdf

[Hellström *et al.* 2018] M. Hellström, M. Johnsson, F. Toussaint, S. Kindermann, D. Lear, R. Huber, M. Stocker, I. Häggström and C.-F. Enell: *A report on negotiations with publishers, providers of existing data citation systems and other scientific organisations on implementing a global data citation system*. ENVRIplus Deliverable D6.2, submitted on April 30, 2018. Available at http://www.envriplus.eu/wp-content/uploads/2015/08/D6.2-A-report-on-negotiations-with-publishers-providers-of-existing-data-citation-systems-and-other-scientific-organisations-on-implementing-a-global-data-citation-system.pdf

[Hellström *et al.* 2019] M. Hellström, M. Johnsson, D. Lear, M. Fiebig *et al*.: Report on identification and citation service case studies. ENVRIplus Deliverable D6.3, work in progress. (Planned submission date February 28, 2019.)

[Kendall *et al.* 1994] S. C. Kendall, J. Waldo, A. Wollrath, G. Wyant, A Note on Distributed Computing, November 1994.

[Kharlamov *et al.* 2016] Kharlamov, E., Brandt, S., Giese, M., Jiménez-Ruiz, E., Kotidis, Y., Lamparter, S., Mailis, T., Neuenstadt, C., Özçep, Ö., Pinkel, C., Soylu, A., Svingos, C., Zheleznyakov, D., Horrocks, I., Ioannidis, Y., Möller, R. and Waaler, A., *Enabling Semantic Access to Static and Streaming Distributed Data with Optique: Demo*, in Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, 350-353, 2016.

[Kilapi *et al.* 2011] Herald Kllapi, Eva Sitaridi, Manolis M. Tsangaris and Yannis Ioannidis, *Schedule optimization for data processing flows on the cloud*, Proc. ACM SIGMOD 289-300, 2011.

[Kiss *et al.* 2017] Kiss, T., Kacsuk, P., Kovacs, J., Rakocsi, B., Hajnal, A., Farkas, A., Gesmier, G. and Terstyanszky, G., *MiCADO–Microservice-based Cloud Application-level Dynamic Orchestrator,* Future Generations of Computer Systems 2017. https://doi.org/10.1016/j.future.2017.09.050

[Klampanos *et al.* 2018] Iraklis Klampanos, Alessandro Spinuso, Antonia Tsili, Federica Magnoni, Amrey Krause, Rosa Filgueira, Xavier Pivan, Christian Páge and Antonis Koukourikos, *D3.5 DARE API*, DARE deliverable, 2018.

[Konstantopoulos *et al.* 2016] Konstantopoulos, Stasinos; Charalambidis, Angelos; Mouchakis, Giannis; Troumpoukis, Antonis; Jakobitsch, Jürgen; Karkaletsis, Vangelis, *Semantic Web Technologies and Big Data Infrastructures: SPARQL Federated Querying of Heterogeneous Big Data Stores*, in proceedings of 15th International Semantic Web Conference (ISWC), Kobe, Japan, 19-21 October 2016.

[Kristeková et *al. 2*006] M. Kristeková, J. Kristek, P. Moczo, & S. M. Day 2006. Misfit Criteria for Quantitative Comparison of Seismograms. Bulletin of the Seismological Society of America, 96(5), 1836–1850. http://doi.org/10.1785/012006

[Kristeková et *al. 2*009] M. Kristeková, J. Kristek & P. Moczo, 2009. Time-frequency misfit and goodness-of-fit criteria for quantitative comparison of time signals. Geophysical Journal International, 178(2), 813–825. http://doi.org/10.1111/j.1365- 246X.2009.04177.x

[Lupp 2008] Lupp M. (2008) Open Geospatial Consortium. In: Encyclopedia of GIS. Springer, Boston, MA

[Maali & Erickson 2014] F. Maali and J. Erickson 2014. Data Catalog Vocabulary (DCAT). W3C Recommendation. W3C. URL: https://www.w3.org/TR/vocab-dcat

[Magagna *et al.* 2018] Barbara Magagna, Malcolm Atkinson and Markus Stocker, *Using data for system-level science: A provenance perspective*, poster at EGU, 2018.

[Magagna *et al.* 2018a] Barbara Magagna, Paul Martin, Doron Goldfarb, Frank Toussaint, Stephan Kindermann, Margareta Hellström, Sébastien Denvil, Marco Rorro, Keith Jeffery, Christian Pichot, André Chanzy, Malcolm Atkinson, *Data provenance and tracing for environmental sciences: system design*, Deliverable D8.5 ENVRIplus project.

[Maggi *et al.* 2009] A. Maggi, C. Tape, M. Chen, D. Chao, & J. Tromp, 2009. An automated time window selection algorithm for seismic tomography, Geophys. J. Int., 178, 257–281.

[Martin *et al.* 2018] Paul Martin, Laurent Remy, Maria Theodoridou, Keith Jeffery and Zhiming Zhao. Mapping metadata from different research infrastructures into a unified framework for use in a virtual research environment, to appear in proceedings of International Workshop on Science Gateways, 2018.

[Martin & Yaikhom 2013] Martin, P. and Yaikhom, G. *Definition of the DISPEL language*, pp203-236, in [Atkinson *et al.* 2013].

[Marz & Warren 2015] Marz, N. and Warren, J., 2015. *Big Data: Principles and best practices of scalable real-time data systems*. New York; Manning Publications Co.

[Michelini *et al.* 2008] A. Michelini, L. Faenza, V. Lauciani & L. Malagnini 2008. ShakeMap implementation in Italy. Seismological Research Letters, 79(5), 688–697.

[Myers et al. 2015] J. Myers, M. Hedstrom, D. Akmon, and S. Payette. Towards Sustainable Curation and Preservation: The SEAD Project's Data Services Approach. 2015

[Nadal *et al.* 2017] Sergi Nadal, Victor Herrero, Oscar Romero, Alberto Abelló, Xavier Franch, Stijn Vansummeren, Danilo Valerio, A software reference architecture for semantic-aware Big Data systems, Information and Software Technology, Volume 90, 2017, Pages 75-92, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2017.06.001.

[ObsPy Team 2017] The ObsPy Development Team. (2017, October 27). ObsPy 1.1.0 (Version 1.1.0). Zenodo. http://doi.org/10.5281/zenodo.165135

[Pääkkönen *et al.* 2015] Pekka Pääkkönen, Daniel Pakkala, Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems, Big Data Research, Volume 2, Issue 4, 2015, Pages 166-186, ISSN 2214-5796, https://doi.org/10.1016/j.bdr.2015.01.001.

[Pagé & Spinuso 2018] Christian Pagé and Alessandro Spinuso, *D7.1 Requirements and Test Cases I*, DARE D7.1, July 2018. URL https://drive.google.com/open?id=1CmwSd0YU73JA80ADW9xLtSwuv9NbQsjJ.

[Perego *et.al* 2016] A. Perego, A. Friis-Christensen, L. Vaccari, and C. Tsinaraki. 2016. Using DCAT-AP for Research Data. In Workshop on Smart Descriptions & Smarter Vocabularies (SDSVoc). URL: https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_27

[Peter *et al.* 2011] D. Peter, D. Komatitsch, Y. Luo, R. Martin, N. Le Goff, E. Casarotti, P. Le Loher, F. Magnoni, Q. Liu, C. Blitz, T. Nissen-Meyer, P. Basini & J. Tromp, 2011. Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes, Geophys. J. Int., 186, 721–739.

[Pierce *et al.* 2018] Marlon Pierce, Mark Miller, Emre Brookes, Mona Wong, Yan Liu, Enis Afgan, Sandra Gesing, Maytal Dahan, Suresh Marru and Tony Walker, **T***owards a Science Gateway Reference Architecture,* in Proc. International Workshop on Science Gateways, 2018.

[Plankensteiner *et al.* 2013] K. Plankensteiner, R. Prodan, M. Janetschek, T. Fahringer, J. Montagnat, D. Rogers, I. Harvey, I. Taylor, Á. Balaskó, P. Kacsuk, *Fine-Grain Interoperability of Scientific Workflows in Distributed Computing Infrastructures*, Journal of Grid Computing **11** (3) 429-456, 2013.

[Quimbert *et al.* 2018] Erwann Quimbert, Keith G Jeffery, Claudia Martens, Christian Pichot, Damien Boulanger, Johannes Peterseil, Christoph Wohner, Harry Lankreijer, Maggie Hellström, Zhiming Zhao, Thierry Carval, *Interoperable cataloguing and metadata harmonisation for environmental RIs*, ENVRIplus deliverable D8.4, October 2018.

[Raj & Tolety 2012] C. P. Rahul Raj and Seshu Babu Tolety, "A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach," 2012 Annual IEEE India Conference (INDICON), Kochi, 2012, pp. 625-629.

[RDA-MS WG 2016] Research Data Alliance, Metadata Standards Directory Working Group: Final Report, 22 January 2016.

[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.

[Rodriguez et al. 2018] Rodriguez, Maria Alejandra and Rajkumar Buyya. "*Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms*." Future Generation Comp. Syst. 79 (2018): 739-750.

[Rodriguez et al. 2017] Rodriguez, Maria Alejandra and Rajkumar Buyya. "*A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments.*" Concurrency and Computation: Practice and Experience 29 (2017)

[Spinuso et al. 2013] Alessandro Spinuso, James Cheney and Malcolm Atkinson, Provenance for seismological processing pipelines in a distributed streaming workflow, in proceedings E DBT '13 Proceedings of the Joint EDBT/ICDT 2013 Workshops, 307-312, 2013.

[Spinuso 2018] Alessandro Spinuso, *Active Provenance for Data Intensive Research*, PhD thesis, University of Edinburgh 2018. URL https://www.era.lib.ed.ac.uk/handle/1842/33181

[Spinuso & Filgueira 2018] Alessandro Spinuso and Rosa Filgueira (eds), *Decomposed User Stories and Tooling*, DARE ID3.1-M6, https://docs.google.com/document/d/1El2syzHIK87usHg7nG98OdeS7VWF3Ft4A0nKyIlCU2c/edit

[Spinuso & Klampanos 2018] Alessandro Spinuso and Iraklis Klampanos, *Integrated Monitoring and Management Tools*, D3.7 deliverable DARE project, December 2018.

[Schubert & Jeffery 2015] Lutz Schubert and Keith G. Jeffery, *New Software Engineering requirements in Clouds and large-scale systems*, IEEE Cloud Computing, 2(1) pp 48-58, 2015.

[Svingos *et al.* 2016] C. Svingos, T. Mailis, H. Kllapi, L. Stamatogiannakis, Y. Kotidis and Y. Ioannidis, "Real time processing of streaming and static information," 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, 2016, pp. 410-415. doi: 10.1109/BigData.2016.7840631.

[Terstyanszky *et al.* 2014] G. Terstyanszky, T. Kukla, T. Kiss, P. Kacsuk, A. Balasko, Z. Farkas, *Enabling scientific workflow sharing through coarse-grained interoperability*, Future Gener. Comput. Syst. (0) 46-59, 2014. doi:http://dx.doi.org/10.1016/j.future.2014.02.016.

[TOSCA WG 2017] OASIS Topology and Orchestration Specification for Cloud Applications Version 1.0, [online] Available from: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html

[Trani *et al.* 2018] Luca Trani, Malcolm Atkinson, Daniele Bailo, Rossana Paciello and Rosa Filgueira, *Establishing Core Concepts for Information-Powered Collaborations*, FGCS vol. 89, 421-437, 2018.

[Twardowski & Ryzko 2014] Twardowski, B., & Ryzko, D. (2014, August). Multi-agent architecture for real-time big data processing. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on* (Vol. 3, pp. 333-337). IEEE.

[W3C-SP WG 2017] W3C-SP, Spatial Data on the Web Best Practices, W3C Working Group Note, 28 September 2017

[W3C-TAG WG 2006] W3C-TAG, On Linking Alternative Representations To Enable Discovery And Publishing, W3C TAG Finding, 1 November 2006.

[Wang Dingxian *et al.* 2014]Wang, Dingxian and Liu, Xiao and He, Zheng and Fan, Xiaoliang. "T*he Design of a Workflow Recommendation System for Workflow as a Service in the Cloud* ". Business Process Management Workshop. Lohmann, Niels and Song, Minseok and Wohed, Petia, 2014.

[Wang, Jianwu *et al.* 2014] Wang, Jianwu, Prakashan Korambath, Ilkay Altintas, Jim Davis and Daniel Crawl. "*Workflow as a Service in the Cloud: Architecture and Scheduling Algorithms.*" Procedia computer science 29 (2014): 546-556.

[Wang *et al.* 2017] Junchao Wang, Arie Taal, Paul Martin, Yang Hua, Huan Zhou, Jianmin Pang, Cees de Laat a, Zhiming Zhao, *Planning virtual infrastructures for time critical applications with multiple deadline constraints*, FGCS **75**, 365-375, 2017.

[Weibel & Koch 2000] Weibel, S. L., and Koch, T. (2000). The Dublin core metadata initiative. *D-lib magazine*, *6*(12), 1082-9873.

[Wilde *et al.* 2011] M. Wilde, M. Hategan, J. M. Wozniak, B. Cli_ord, D. S. Katz, I. T. Foster, *Swift: A language for distributed parallel scripting*, Parallel Computing 37 (9) (2011) 633-652.

[WMO 2013] WMO-WIS, http://www.wmo.int/pages/prog/www/WIS/metadata_en.html

[Wu *et al.* 2017] C. Wu, R. Tobar, K. Vinsen, A. Wicenec, D. Pallot, B. Lao, R. Wang, T. An, M. Boulton, I. Cooper, R. Dodson, M. Dolensky, Y. Mei, F. Wang, *DALiuGE: A graph execution framework for harnessing the astronomical data deluge*, CoRR abs/1702.07617.

[Yin *et al.* 2018] **Yi Yin**, Anneke Zuiderwijk, Marijn Janssen, Xander de Ronde and Keith Jeffery, *Multidisciplinary Collaboration Through Online Virtual Research Environments (VREs): what do their users need?* in Proc. International Workshop on Science Gateways, 2018.