D-JRA2.1 — Initial architecture: first selected components

22/05/2012

Project acronym:	VERCE
Project n°:	283543
Funding Scheme:	Combination of CP & CSA
Call Identifier:	FP7-INFRASTRUCTURES-2011-2
WP:	WP9/JRA2, VERCE Architecture and Tools for Data Analysis and Data Modelling Applications
Filename:	D-JRA2.1.pdf
Author(s):	Paul Martin, Malcolm Atkinson, Michelle Galea, Iraklis Klampanos, Amrey Krause, Rosa Filgueira, Chee Sun Liew, Alessandro Spinuso, Luca Trani, Siew Hoon Leong and Xiao Wang
Location:	http://www.verce-project.eu/Repository/Deliverables/RP1/
Type of document:	Deliverable RP1
Dissemination level:	Public
Status:	Final
Due date of delivery:	15/05/12
Reviewers:	H. Schwichtenberg (content), Geneviève Moguilny (format)
Keywords :	JRA2, data-intensive, compute-intensive, distributed-systems, architecture, e-Infrastructure, tools.

Ver.	*Author*		*Date*	*Comments*
1	P. Martin	(UEDIN)	12/3/2012	Initial draft, basic layout.
2	P. Martin		28/3/2012	Annotations of content for delegation.
3	A. Krause	(UEDIN)	30/3/2012	Added information about enactment.
4	X. Wang	(ULIV)	30/3/2012	Added information about SDX.
5	R. Filgueira	(UEDIN)	04/4/2012	Added information about job submission and data transport.
6	A. Spinuso	(KNMI)	09/4/2012	Added information about test-case, provenance and code injection.
7	S. H. Leong	(BADW-LRZ)	09/4/2012	Added information about HPC interaction.
8	P. Martin		10/4/2012	Added information about various subjects.
9	M. Galea	(UEDIN)	14/4/2012	Added information about workflows.
10	P. Martin		15/4/2012	**1st internal review draft.**
11	I. Klampanos	(UEDIN)	23/4/2012	Minor edits based on comments by M. Atkinson.
12	M. Atkinson	(UEDIN)	25/4/2012	Added section on architect's role and vision; minor additions.
13	I. Klampanos	(UEDIN)	26/4/2012	**2nd internal review draft.**
14	M. Atkinson and I. Klampanos	(UEDIN)	1/5/2012	Integrated comments from internal review.
15	M. Atkinson	(UEDIN)	2/5/2012	Further edits responding to H. Schwichtenberg internal-review comments.
16	P. Martin	(UEDIN)	3/5/2012	Added precise architecture list after discussions in IPGP.
17	M. Atkinson	(UEDIN)	3/5/2012	Further edits responding to H. Schwichtenberg internal-review comments continued and responses to discussions at IPGP.
18	M. Atkinson	(UEDIN)	6/5/2012	Revised executive summary.
19	M. Atkinson	(UEDIN)	7/5/2012	Added several definitions in section 1.3 and produced candidate final version.
20	I. Klampanos	(UEDIN)	9/5/2012	Corrections and updates to glossary.
21	M. Atkinson	(UEDIN)	14/5/2012	Corrections to keywords and executive summary.
22	M. Atkinson	(UEDIN)	15/5/2012	Compliance with formatting suggested by G. Moguilny.

Copyright notice

Contents

List of Figures

List of Tables

Executive summary

One of the principal goals of the VERCE project is the development of tools and services for data analysis and computational modelling applications for use by seismologists and other Earth scientists. In order to achieve this goal, an integrating framework for heterogeneous task submission and execution is being developed, which can be placed between the user-oriented world of scientific gateways and data-analysis development, and the system-oriented world of high-performance computing (HPC), grids and data archives. By providing a sufficiently flexible technical hub for the future VERCE Platform, it should become far easier to integrate a range of tools and services via well-understood interfaces. Thus the main purpose of this initial iteration of the VERCE platform is to produce a suitable prototype system, which permits the submission of computational and data-intensive workflows. This prototype system then delegates execution of those workflows onto a distributed set of heterogeneous resources. The system then gathers monitoring information and results in order to relay them to the researchers who initiated each workflow.

The driving requirements of seismology, which will couple data-intensive processes with HPC model runs, demand new e-Infrastructure[1]. VERCE will pioneer this e-Infrastructure for seismology with the intention of making the knowledge gained and prototyped software available to other disciplines. VERCE will work closely with other e-Infrastructure organisations to ensure that, as far as possible, the e-Infrastructure's foundations are widely shared and therefore sustainable and affordable.

The architectural role of JRA2 includes helping the VERCE project maintain a balance between long-term sustainability considerations and the scientific imperative to implement use cases quickly in order to explore and profit from the new possibilities that the VERCE project opens up. To that end, this report goes beyond listing the components in the first prototype; it also sets out the current context and initial views on how to address that need for balance. This is inevitable more complex than the initial prototype, which was mainly shaped by prior work, as the priority seismology use cases have only recently been identified by colleagues in work packages NA2 and JRA1. The understanding required to properly address this balance is in its early stages of development. Clarifying and formulating this understanding will continue to be a major concern for JRA2 throughout the project.

The most significant components in the initial iteration of the VERCE architecture are listed below.

Gateway Based on technology developed during the ADMIRE project[2], the gateway is the hub of the VERCE platform, delegating the enactment of user workflows to available distributed resources. Currently, only workflows written in the Dispel workflow language are accepted for deployment and execution on OGSA-DAI services[3]. The language Dispel is chosen for the present at least for three reasons: (i) it is data-flow based for multi-site enactment, (ii) it has functions to describe work patterns, and (iii) it is designed for human communication and to avoid detail that inhibits automated mapping and optimisation.

Grid integration Grid computing factors heavily into the VERCE project, especially as regards data movement. Greater support for GridFTP for reliable file movement has been integrated into OGSA-DAI in accordance with the specifications of the Globus project[4].

Obspy integration In order to promote uptake of any platform, it is important to make it as simple as possible for researchers to continue to use the languages, tools and libraries that they already use

[1] An e-Infrastructure is a distributed collection of data, storage and computational resources, interconnected by digital communications and organised to serve a common purpose. It includes the hardware, software, middleware, staff, operational procedures and policies needed to make it operate for that purpose, and requires maintenance to function in the evolving digital environment and to meet the changing needs of its user communities.

[2] http://admire-project.eu

[3] http://www.ogsadai.org.uk

[4] http://www.globus.org

fluently. A generic mechanism for implanting arbitrary Python (but specifically Obspy[5]) scripts into OGSA-DAI activities has been prototyped to this end.

Cross-correlation test case In anticipation of more substantive use-cases from the NA2 and JRA1 work packages, a simple test case for the cross-correlation of ambient noise from seismograms was put together in order to test the prototype platform and identify early requirements *vis-à-vis* data handling, process execution and distribution. The test-case was tested on EDIM1 [Martin et al., 2011], the Edinburgh Data Intensive Machine, a data-brick compute cluster [Szalay et al., 2010] operated by the University of Edinburgh.

SDX The Seismic Data eXplorer, developed at Liverpool University, initially as part of the RapidSeis project[6], is a tool for seismic waveform analysis and has been further refined within VERCE in work package SA3. It is an example of the kind of tool we wish to integrate into the architecture and JRA2 is investigating the requirements this raises.

Aside from the specification of prototype itself, this report also presents a general overview of the VERCE architecture, as well as a survey of *some* of the areas from which the VERCE platform must draw tools and services.

The current prototype is closely linked with data-intensive methods and with the R&D in SA3 to develop an initial scientific gateway. The next steps will incorporate more requirements from SA3, particularly support for registries and catalogues. During the first six months of the project the practical work concerning HPC in JRA1 has been separate from the JRA2 prototype. In the next six months these will be brought together. This report concludes with a section identifying the role of the architect, JRA2, and of the vision that gives the architectural decisions coherence.

[5] http://obspy.org
[6] http://research.nesc.ac.uk/node/423

1 Introduction

VERCE aims to deliver an e-science environment for data-intensive research geared towards the needs of the earthquake and seismological community; it aims to enable that community to easily exploit the increasingly large and available volumes of seismological data, data mining and modelling applications, and distributed Grid and HPC computing resources, in order to significantly advance the science by exploring new avenues of research. The purpose of this deliverable is to describe how investigations on the VERCE architecture have proceeded, the initial technologies that have been selected for prototyping purposes, and plans for the next stage of development of the architecture.

1.1 Role of JRA2

JRA2 (VERCE architecture and tools for data-analysis and data-modelling applications) leads the development of the VERCE architecture, and prototypes new distributed, high-level integration services and tools to enable new methods for seismology research. Its objectives are to:

1. Define the VERCE architecture and prototype critical components and services;
2. Identify and adapt existing seismology data resources and analysis tools for integration with the architecture;
3. Select and adapt a toolset for the development, parameterisation, extension and optimisation of scientific data-intensive workflows;
4. Select and adapt technologies for the VERCE scientific gateway to facilitate uptake by the user community.

The interaction of JRA2 with the other VERCE work packages is illustrated in Figure 1, where core information flows are drawn in blue.

Based on the prioritised scientific requirements captured in NA2 (pilot applications and use cases), JRA1 (harnessing data-intensive applications) identifies the resulting technical requirements and high-level services and tools to be integrated within the architecture being defined by JRA2. The two JRA work packages coordinate with each other to find a realistic balance between using existing facilities and re-engineering. The former depends on developing wrapping and descriptive technology. The latter selects application code and services and identifies how they should be changed.

JRA2 defines the prototype services and components of the architecture, but these are largely implemented by SA2 (integration and evaluation of the platform services) and SA3 (scientific gateway, user interfaces and knowledge and method sharing). In particular, JRA2 will clarify and refine the architecture definition as appropriate so that other work packages are better able to adopt it, *e.g.* describe its components in the terms necessary for the tools to facilitate their use in workflows and applications.

To a lesser degree, and when appropriate, JRA2 also provides input to SA1 (management and operation of the research platform), NA3 (training and user documentation), and NA4 (dissemination and public outreach). There are two benefits for JRA2 from engaging actively in these two work packages:

1. observing training and response to outreach quickly exposes deficiencies in the current design that JRA2 can take into account at the next iteration of the architectural prototype; and
2. long-term sustainability depends on attracting enough users that the cost of running and maintaining the e-Infrastructure can be seen to be warranted.

1.2 Architecture development strategy

Development of the VERCE architecture is based on a six-monthly review and refinement cycle. At the end of each six-month development period an enhanced VERCE architecture is released to the VERCE

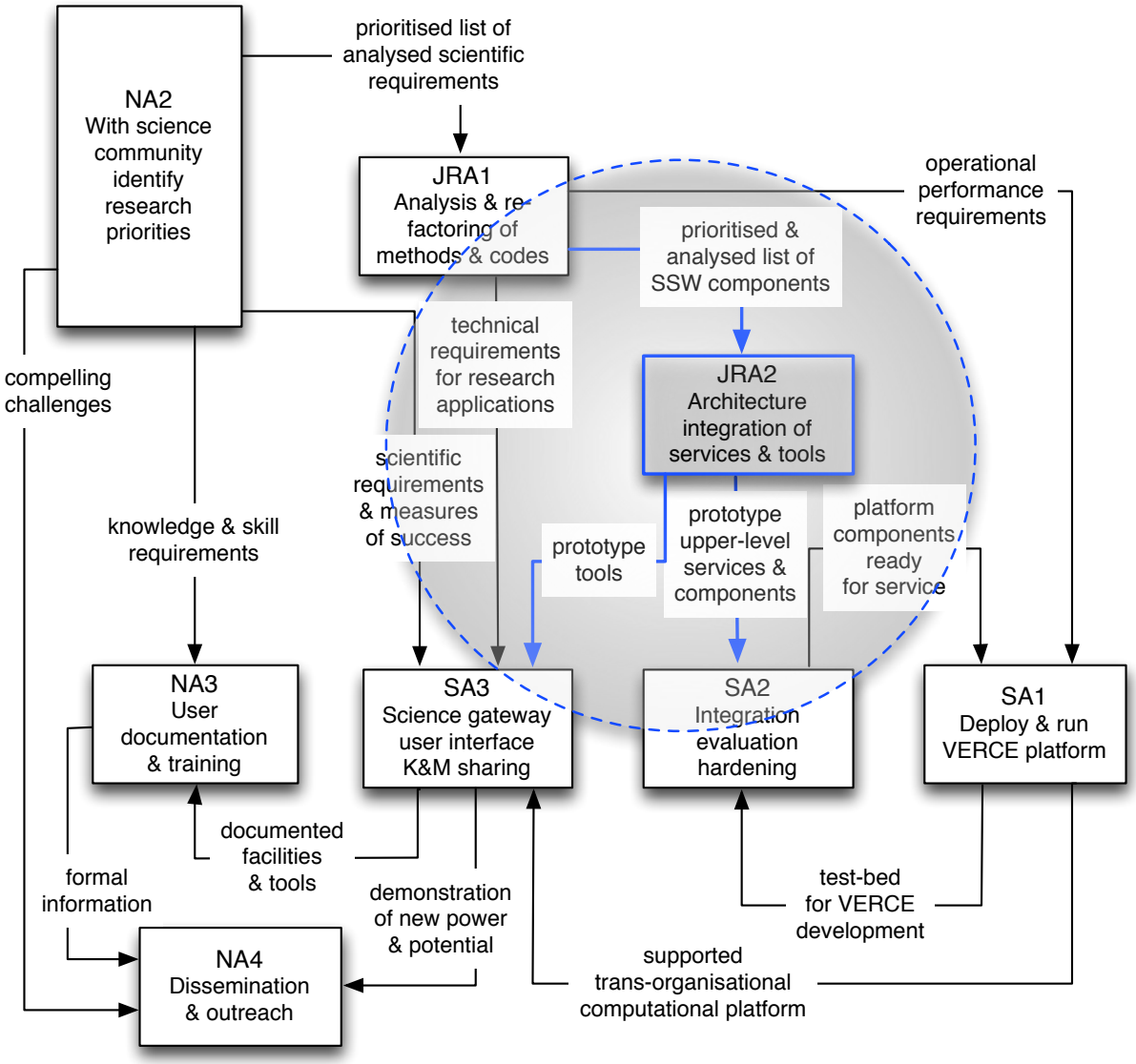


Figure 1: JRA2 interaction with VERCE work packages.

consortium by JRA2, subject to ratification by SA2. This approach ensures that a working architecture is provided to the VERCE consortium and interested parties as early as possible. Adoption of the architecture will be facilitated by portals and tools, presented by SA3, and the training and dissemination efforts of NA3 and NA4.

During each development iteration the VERCE researchers and research-application developers will be involved in the process, both to gather a better understanding of existing methods and the potential for new methods in seismology, and to harness their effort in bringing in data resources, tools and services. This will also provide regular feedback as to the adequacy of the emerging architecture and its components, and help identify priority areas for new or further development in the later development cycles.

The initial VERCE architecture will draw heavily on the data-intensive architecture, developed and evaluated within the ADMIRE project. The VERCE community requirements for its e-science data-intensive environment will be driven primarily by the use cases prioritised in NA2, and these in turn will determine how best to adapt and introduce the architecture into the HPC (CPU-intensive) and data-centre (archival, data-intensive) contexts of the VERCE community.

To avoid duplicated effort and to facilitate interdisciplinary research and sharing, JRA2 will also make

best use of existing and developing technologies of the VERCE consortium, collaborate with and draw on the experience of other EU projects, such as IGE [7], EUDAT[8] and MAPPER[9], and will negotiate with and use existing and developing EU e-Infrastructures such as EGI[10] and PRACE[11]. Members of JRA2 are also involved in the design work package of the ENVRI project[12], which is identifying and designing the common e-Infrastructure components for the ten environmental science ESFRI projects, including the EPOS Research Infrastructure[13]. We anticipate that this will facilitate the adoption of such common components later in the VERCE project. This in turn should facilitate sustainability by sharing costs and enhance the opportunities for interdisciplinary research using the VERCE e-Infrastructure.

[7] http://www.ige-project.eu/
[8] http://www.eudat.eu
[9] http://www.mapper-project.eu
[10] http://www.egi.eu
[11] http://www.prace-project.eu
[12] http://envri.eu
[13] http://www.epos-eu.org

1.3 Definition of terms

Definition of terms

Term	Explanation
e-Infrastructure	The ICT element of a research infrastructure, i.e. a distributed collection of data, storage and compute resources, interconnected by digital communications and organised to serve a common research purpose. It includes the hardware, software, middleware, staff, operational procedures and policies needed to make it operate for that purpose, and requires maintenance to function in the evolving digital environment and to meet the changing needs of its user communities.
Gateway	A software subsystem, typically at the middleware level, that accepts requests for computational and data-handling tasks. It vets those requests to establish whether they are valid, e.g. are syntactically and semantically consistent, and are authorised. Requests that aren't validated are rejected. Requests that are accepted are passed to other software systems, at the same or other locations, for execution. The requests may be partitioned and translated to combine heterogeneous services.
Research Infrastructure	The collection of equipment, resources, organisations, policies and community support that enables a particular discipline to conduct research. Normally, this refers to the advanced facilities that enable frontier research, such as the research infrastructures endorsed by ESFRI[14].
Research Object	A research item which some researcher wishes to identify. It may be a collection of primary or derived data, code, a workflow, a service, an ontology, a set of metadata, etc. It may be a paper or a talk. Often it is a composition of such elements.
Science gateway	A consistently presented set of facilities designed to be a convenient working environment for researchers in a particular domain, in this case seismology. It should bring together access to all of the capabilities and resources such a researcher needs: including catalogues of available data and tools, established methods and arrangements for applying them with specified parameters to specified data. Underpinning it will be the e-Infrastructure, which will perform the processes implementing the methods, including data handling, computation and scientific record keeping. The VERCE scientific gateway will also provide facilities for establishing identity and authority. Researchers will be able to associate a profile with their identity, which may include persistent storage of any research object corresponding to work in progress. Normally, a scientific gateway will encourage sharing and will provide mechanisms for controlled release of research objects and will facilitate proper attribution. It will also provide mechanisms for adding and updating the presentation and the capabilities on offer for those with the authority to manage its content. SA3 has the primary responsibility for the VERCE science gateway. The term 'science gateway' originates in the USA, and is effectively synonymous with the term 'virtual research environment (VRE)', which originated in the UK and Australia, partially to include disciplines that needed data and computation but which did not call themselves sciences. There is a nuance that a VRE includes and shapes the underpinning e-Infrastructure, whereas a science gateway enables access to a pre-designed e-Infrastructure.
VERCE architecture	A high-level and coherent design for the VERCE e-Infrastructure; it evolves as the seismological goals and digital environment evolve and become better understood. It should guide the development of successive VERCE platforms.

VERCE e-Infrastructure An envisaged result of VERCE, as an integrated computational and data environment that presents a coherent virtual research environment in which to conduct seismology research and eventually research in other Earth sciences.

VERCE Platform The current realisation of the VERCE e-Infrastructure at any time in the VERCE project. Initially this is not fully integrated and may only constitute a partial implementation. Nevertheless, it is sufficient both to pursue research identified as priority seismology use cases and to develop and test the design of the VERCE e-Infrastructure. The VERCE platform is an approximation to the VERCE e-Infrastructure. These approximations should converge on the VERCE e-Infrastructure by the end of the VERCE project.

Virtual Research Environment A presentation of (ideally all of) the resources a researcher may need in a consistent and easily used form. These resources include catalogues, data, metadata, libraries, tools, workflows, programs, services, visualisation systems and research methods. Its coherence is intended to accelerate the uptake of its facilities and to expedite the research processes by making sure everything is to hand. It arranges that it is easy to create and share effective methods and that unnecessary technological detail is hidden. Normally, a VRE also attempts to develop a distributed community by providing collaboration and communication aids. These combine to encourage faster education of its new users. For most researchers, most of the underpinning computational, communication and data-handling will be hidden, but there may be some who want access to that technological level.

Workflow A process of composed data-handling tasks, computational tasks and human interactions intended to implement a research method or established working practice. Initially, these are often carried out by humans '*orchestrating* each step of the process via command-line and web-interface interactions. In that form they are shared and improved by cultural practices and education; for example, key workflows may be explained to new researchers on joining a laboratory, and recorded in laboratory notebooks. As part of the adoption of e-Infrastructure, these workflows are progressively automated, by encoding sequences of tasks in scripting or workflow languages. This may make them more tangible as research objects and IPR. Automated workflows should relieve researchers of tedious processes so they can focus on the remaining essential creative and judgemental tasks. This automation should decrease errors, accelerate research and improve the quality of the scientific record. Making a workflow a tangible research object allows it to be the subject of improvements and facilitates controlled workflow sharing.

1.4 Deliverable structure

This deliverable is divided into six parts, excluding this introduction. These parts are as follows:

Architectural overview VERCE seeks to integrate a wide range of resources into an e-Infrastructure that presents a coherent virtual research environment for seismologists. This section discusses the VERCE architecture both as a concept and as an artefact with which a user may interact.

Prototype specification An initial selection of tools and services have been made so as to elicit feedback on JRA2's current understanding of requirements. This section describes this initial selection and discusses the reasoning behind it.

Cross-correlation test case In anticipation of more advanced use-cases being generated by the NA2 and JRA1 working groups, the VERCE prototype platform has been used for a simple noise-correlation test case.

Additional technology survey A significant part of JRA2 is the identification of existing technologies which might be integrated into the VERCE platform. This section surveys a subset of the areas in which such technologies might be found, both in competition with and to complement the technologies already selected for the first prototype.

Future development Plans for future development of the VERCE platform must be explored, with acknowledgement of critical areas not yet adequately addressed. A long-term concern is sustainability. When VERCE delivers an effective virtual research environment the research of the seismologists using it will prosper. Hence the ability to keep the VERCE e-Infrastructure running will assume importance that is not so evident while it is under construction. The JRA2 work package takes responsibility for considering this issue. For example, by limiting complexity and by sharing common solutions with other e-Infrastructure organisations.

Architect's Role and Vision The role of the JRA2 work package as architect is reviewed to show how architects have to balance a number of conflicting pressures while giving priority to developing the brief present by the clients, NA2 and JRA1 in this case. To explore how this balance should take into account long-term issues, a vision of a virtual research environment (VRE) for Earth Scientists in the early post-VERCE era is presented. This invites clarification of the conceptual aspects of the VERCE brief.

2 Architectural overview

The VERCE architecture is founded on a principle of accessibility at multiple levels – whether by the definition of computing tasks via suitable scripting interfaces, or by arrangement and scheduling of logical workflows via web portals. The architecture must be extensible, and support a variety of low-level technologies, but must nonetheless adhere to a consistent model which supports the tracking of provenance and thus the replication of experiments. By building a new distributed computing platform targeted and driven by seismology around a core of well-founded use-cases, we can motivate timely adoption by the scientific community. By using existing technologies, we hope to exploit the accumulated expertise of prior (and contemporary) projects in order to achieve more, to prevent further fragmentation of a technical community (already swamped with partially-implemented solutions) and to enhance the potential for sustaining the VERCE e-Infrastructure after the completion of the VERCE project.

To achieve the goals of the architecture a conceptual model must be chosen which supports the vision (Section 2.1). In spite of being still in the early stages of the project, thought must also be given now

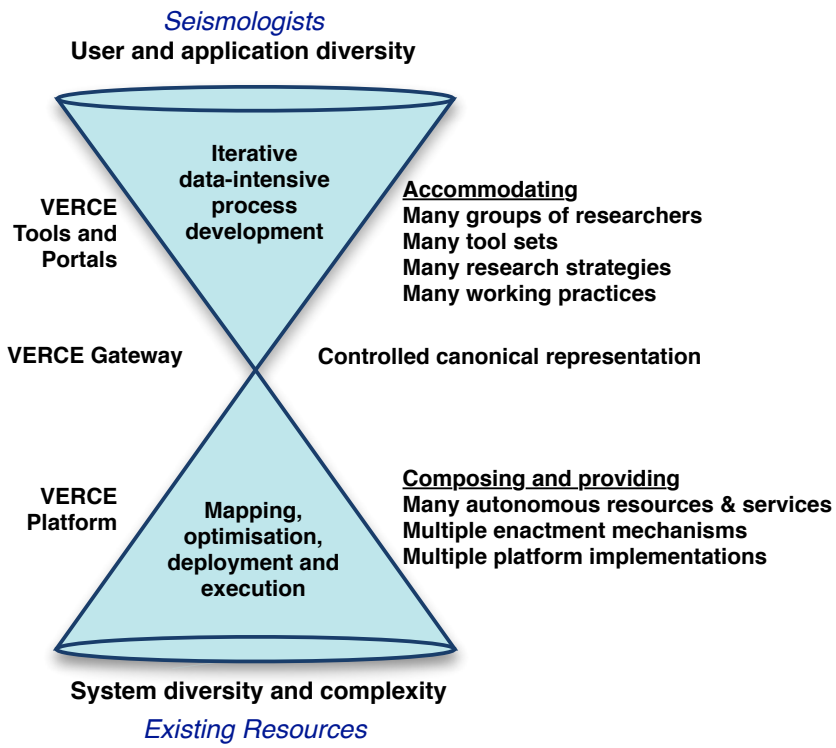


Figure 2: The hourglass model of distributed computation for VERCE.

as to how users should be expected to interact with the architecture (Section 2.2), and to how tasks can be choreographed and enacted across a distributed and heterogeneous set of computational, storage, grid and data-centre facilities (Section 2.3).

2.1 Architectural Model

In order to partition the complexity, it is helpful to consider three broad classes of research participants: domain experts, who are knowledgeable about the solid-Earth science that they wish to pursue, but who are not necessarily accustomed to using complex ICT tools and services[15]; data-analysis experts, who understand the algorithms and methods needed by the domain experts and the tools to implement those algorithms and methods, but who are not au fait with the engineering requirements of distributed computation and data manipulation; and data-intensive engineers, who often know little about the science which they are being asked to provide a platform for.

The VERCE architecture is being designed with the requirements of data-intensive computational science in mind. Specifically, it is being built using the hourglass model (Figure 2) first proposed by the ADMIRE project [Atkinson et al., 2011]. The idea is that we insulate the domain expert from the underlying execution platform by providing a common gateway through which various tools and services can request computational resources upon which to execute workflows for various experimental tasks. These tools and services used by domain experts concern themselves only with the logical specification of workflows whilst the gateway concerns itself with finding a way to implement such specifications. It is the role of data-analysis experts to provide the codes which implement different workflow elements, based on their knowledge of the research domain and its computational requirements. These codes will be made discoverable via a *registry*, keeping track or relevant logical entities, and whose services will

[15]Information and Communication Technology (ICT) is an all encompassing term for software, computational resources and digital communications that enable an e-Infrastructure to be built and to be presented as an easy-to-use virtual research environment.

be accessible by the VERCE submission gateways. Such a registry will provide information about about workflow elements so that each gateway uses the same information. The registry will play an important role in the sustainability and extensibility of the VERCE architecture. A first version of the registry will be developed during the next phase of the architecture specification.

Responsibility for executing these codes is then delegated to any available resources – whether individual computers, Grid services, data archives or high-performance compute clusters. Optimising selection of such resources, as well as providing the basic service infrastructure to coordinate execution between disparate computation environments, is the job of the data-intensive engineers, who are themselves distanced from the task of translating domain experts' requirements into computations.

ADMIRE implemented this hourglass model by providing a gateway service which would accept any valid workflow described using the Dispel workflow language and would produce a workflow graph which could be used to map logical workflow elements (processing element – PEs) onto activities written in OGSA-DAI, a streaming-data distributed workflow execution platform. The ADMIRE gateway was written so that alternative execution platforms could be used (potentially in combination). We plan to exploit this potential for interaction with multiple execution environments in VERCE. Dispel is a potential *lingua franca* for communication between top-level services and intermediary gateways.

By using an adapted ADMIRE gateway as the central hub for the VERCE architecture and by progressively modifying these gateways we can build interfaces with a variety of different top- and bottom-level technologies (such as those referenced in Figure 3) quickly and effectively. This will allow us to support a range of use-cases, and provide acceptable points of ingress into a wider range of HPC, Grid and data-archive resources, despite an underlying lack of uniformity of technology and access. But we also remain open to building on and adapting other integration technologies, either in combination with these gateways or as a replacement, should the alternatives be well supported and obviously beneficial. If this redirection of effort were to prove desirable later in the project, the work based on the existing gateways would remain a considerable advantage. It would have clarified the seismology requirements, workflows and workloads, and it would have established interfacing strategies with a number of the heterogeneous existing technologies that need to be integrated.

2.2 User Interaction

Key to promoting use of any architecture is the provision of a variety of means to interact with the architecture directed at different classes of user with different ranges of expertise. It is currently believed that the majority of interaction with the VERCE architecture will be through web portals. It is therefore necessary that there exist both dedicated web portals for each VERCE use-case (addressed to the typical domain expert for the use-case), and more flexible generic portals for more technical or unorthodox users.

The kind of approach envisaged involves the swift deployment of simple portlets for specific use-cases which permit the adjustment of a few core parameters before submission of tasks to the platform. An example of such a portlet creation technology is Rapid[16], which was adapted as part of the RapidSeis project to contribute seismology-focused portlets for the NERIES project[17]. However whilst we envisage the majority of users as being concerned only with their interactions with one of a number of portals tailored towards executing one of a number of pre-configured scientific workflows, it is expected that a few users will be willing to learn how to specify workflows manually in order to acquire greater power and control over their intended experiments, as well as to prototype the implementation of new workflows for various tasks not catered for by the infrastructure already. To facilitate this, the VERCE infrastructure requires a common language for specifying workflows, one which can be translated into the specific constructs used by the various underlying workflow execution technologies used by different parts of the infrastructure.

[16] http://research.nesc.ac.uk/rapid
[17] http://www.neries-eu.org

Even should the workflow language be an abstract one, the common use of such a language entails the adoption of certain assumptions about workflows and how they are executed – assumptions which must be common throughout the infrastructure. For example, distinction must be made between *data-flow* oriented workflows and *control-flow* oriented ones [Deelman et al., 2009]:

- *Data-flow* oriented workflows concern themselves only with the flow of data units from one processing element to another. Decoupling the logical view of data dependency between workflow elements from any specific notion of control passing permits the execution environment to which a workflow is delegated to make its own decisions regarding parallelisation of data processing and the rate of data streaming between different points of the workflow. The flip-side is that greater intelligence must be embedded in the workflow executor, such that it can adequately buffer data *in situ* and avoid dead-lock situations.
- *Control-flow* oriented workflows concern themselves with the passing of control between computational components, such that each sub-task is executed in accordance with a well-defined scheme. A certain amount of data-flow may be implied, but the associations between workflow elements is one of passing of execution privilege, not piping of data. Control-flow oriented workflows can be a simple way to choreograph how distributed resources interact for complex tasks, but by that very virtue produce very rigid specifications of resource topology and behaviour.

The workflows currently presented to JRA2 in VERCE are data-flow oriented. This permits the separation of the logical view of the workflow from the physical topology of execution environment. It makes no assumptions about how data is transported between resources and how control is distributed between technologies built upon those resources. This allows independent environments to configure themselves differently for the same abstract workflow. For this reason, Dispel was retained as the language for specifying workflows, pending further requirements analysis.

Data-analysis experts will have established ways of working, e.g. with MatLab or Python, and well-honed libraries which are key assets in their work. Consequently the VERCE e-Infrastructure must enable these experts to bring their skills and assets to bear on the Earth-science challenges as easily as possible. In the long run, this requires computational bridges with their services and stand-alone tools, harnesses and interactive development environments (IDEs)[18] that facilitate the creation of new and revised data-analysis components and registries which encourage their discovery and reuse. To facilitate the engagement of the data-analysis experts, we need to provide intellectual ramps, which allow these experts to engage with the extended facilities incrementally as and when they want to; this is well illustrated by Microsoft Research's Data Scope that provides data-analysis tools running on the Azure Cloud[19] as a data-exploration and analysis band in Excel [Barga et al., 2010]. Building adaptors for the data-analysis expert's favourite tools is complex and labour intensive, and there is normally diversity in their work patterns and preferences. Hence this cannot be completed within the VERCE project alone. Initially, we will provide facilities based on the Eclipse system and adapt these as user requirements become better understood.

All gateway services, which lie between front-end portal services (and tools) and back-end resources, should have standard interfaces through which to direct submission of workflows and other useful directives, allowing expert users to use their own favoured tools for interacting with the VERCE platform[20]. The current VERCE gateway is RESTful (i.e. complying with the specification for REpresentational

[18] A harness in this case, is an automatically generated program text, that is shaped for the task the data-analysis expert wants to undertake, and is in the expert's preferred language. It takes care of the house-keeping, internal interfaces and safety mechanisms, so that the expert only has to write the key algorithm in the preferred language. An Interactive Development Environment (IDE) is an interactive environment presenting a software engineer with the tools needed to undertake software tasks – Eclipse is a popular academic example (http://www.eclipse.org), RationalRose a commercial strength alternative (http://www.ibm.com/software/rational).

[19] Microsoft's cloud computing platform. http://www.windowsazure.com

[20] For testing and diagnosis during development and for those who prefer a command-line environment, this can include a set of unix commands to perform interactions with gateways.

State Transfer [Fielding, 2000]), permitting directives to be embedded in HTTP instructions (see Table 1 in Section 3.1).

Even given a robust workflow specification language, there will remain cases where users will wish to access resources directly via a remote command-line. In this scenario, a standard means of user authentication would be useful, which then permits the user to invoke commands as befits their authority. The specific instructions available to such a user will remain particular to the technologies chosen as part of the VERCE platform. Conceivably, the creation of standard scripts for easy invocation of underlying services (or common arrangements of services) may be worth pursuing — this is not deemed a priority in the short term,except for testing services.

An issue in user interfaces, regardless of their exact implementation or presentation, and in particular those designed for use within a research context, is the handling of errors. We may, in due course, need to provide diagnostic and performance information, possibly with different views for the three categories of expert. In the longer term it is important that as details of lower-level computational and data-handling activity are automated and hidden from the researcher, there remains some indication of the cost of high-level requests. This is necessary, so that researchers develop economic research strategies.

2.3 Enactment and Choreography

Complex computational experiments can be decomposed into workflows charting the flow of data from one activity to another[21]. The workflows which must be considered by VERCE include both the intrinsic workflows describing data-flow between core VERCE services and the user-defined workflows used to perform specific tasks with those services; most of the discussion below is equally applicable to both.

Choreography of workflows over distributed computational resources requires both the coordination of control between components and services, and the efficient transfer of data between resources; one particular subtlety is the delegation of tasks to resources which minimises the time spent awaiting delivery of data. This must be counter-balanced with the need to use dedicated compute resources (such as HPC facilities) for several seismological uses-cases, resources which typically do not permit long-term retention of data at the resources themselves.

Efficient execution of workflows is facilitated by the streaming of data continuously between components such that an activity can immediately begin execution upon initial production of subject data without awaiting the completion of production by a prior activity. Nonetheless, the VERCE use-cases require significant quantities of data to be transferred between geographically disparate resources, which may prove impractical to stream across a single channel. In such circumstances fast, parallelisable transfer of files will be required using advanced protocols, such as GridFTP[22], which permit partial, parallelised and third-party data transfers given suitable security credentials on the part of both client and server.

Execution of individual workflow components will be delegated to separate enactment platforms such as OGSA-DAI. Standard interfaces will be used to permit heterogeneous orchestration between different enactment services. These services will be suitable implementations of generally tried-and-tested application codes used by the seismology community. These may need to be embedded within a standard wrapper for execution as part of a workflow. Some of these enactment platforms and application codes will be specifically attached to certain resources (*e.g.* HPC or data-archive facilities) for which they are optimised, licensed or bound; some application codes will be generic, for execution on arbitrary systems.

Workflows need to be enactable within existing computational frameworks, specifically the Grid (rep-

[21] An existing program or service that takes parameters and yields results, often as files, can be incorporated into such a framework as follows. Incoming requests can be seen as a data stream of values, each one containing a set of parameters. When each parameter set arrives the program or service is activated with that parameter set. The results are gathered into values and despatched as an output data stream.

[22] http://www.globus.org/toolkit/

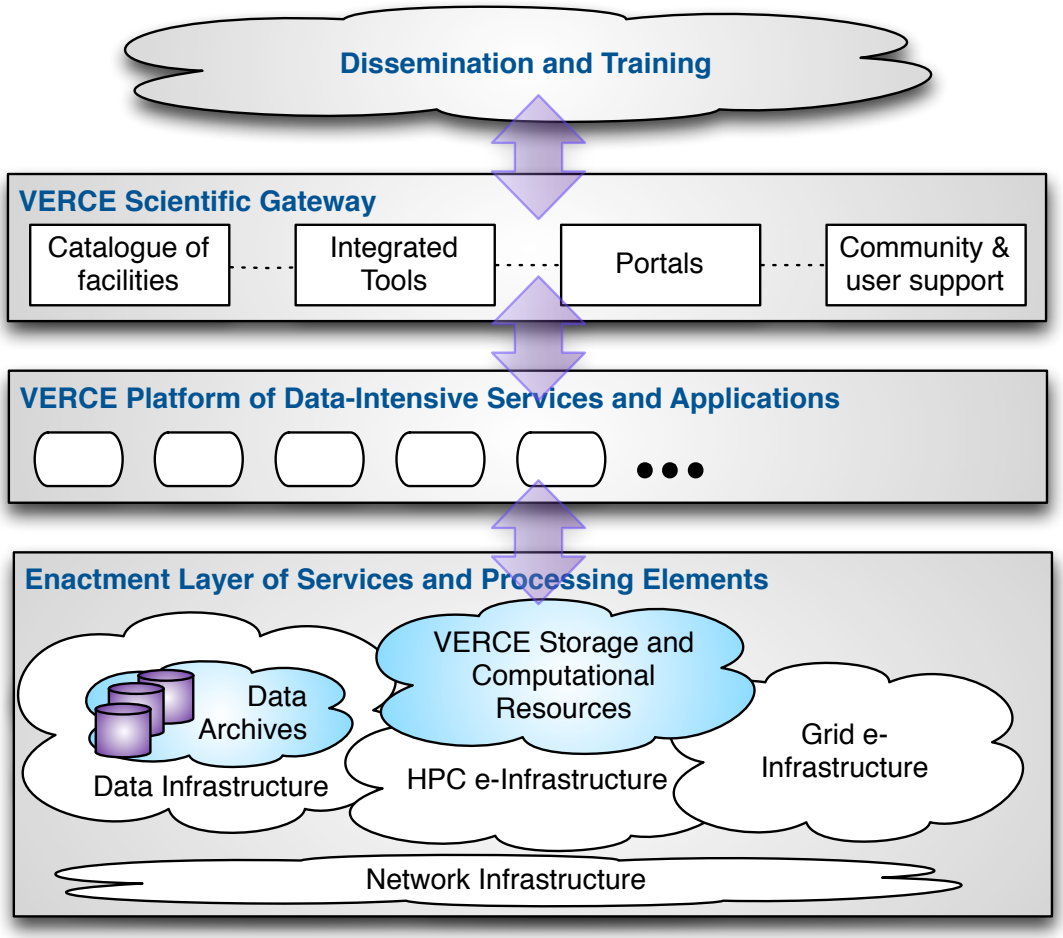


Figure 3: How the VERCE architecture is perceived. A number of candidate technologies for the main technology stack have been identified and investigated to various degrees.

resented in Europe by EGI) and PRACE (the project concerned with standardising services across European HPC centres)[23]. This ultimately means that application codes must be compliant with the requirements of those frameworks and, where necessary, approved by suitable authorities (especially in cases where codes must be installed on resources operated by project members). Workflows will need to be able to interact with existing seismology-oriented data archives such as ODC[24], GFZ[25] and IRIS[26], especially via protocols such as ArcLink as used by EIDA (the European Integrated Data Archives)[27].

3 Prototype specification

The first prototype of the VERCE architecture concentrates on existing technology and achieves a modest selection of basic goals for the architecture. It is designed to provide an immediate indication of the thinking of the JRA2 group to the rest of the VERCE consortium, permitting rapid feedback. In response

[23] As the use cases are analysed JRA2 should record which aspects of these facilities and services should be used in the VERCE e-Infrastructure – their total complexity is well beyond the VERCE project's capabilities but it is reasonable to expect to use a subset of services.

[24] http://www.orfeus-eu.org/Data-info/data.html

[25] http://www.gfz-potsdam.de/portal/gfz/home

[26] http://www.iris.edu

[27] http://verce.eu/ITCoordinationMeetingFebruary2012/EIDA-Overview.pdf

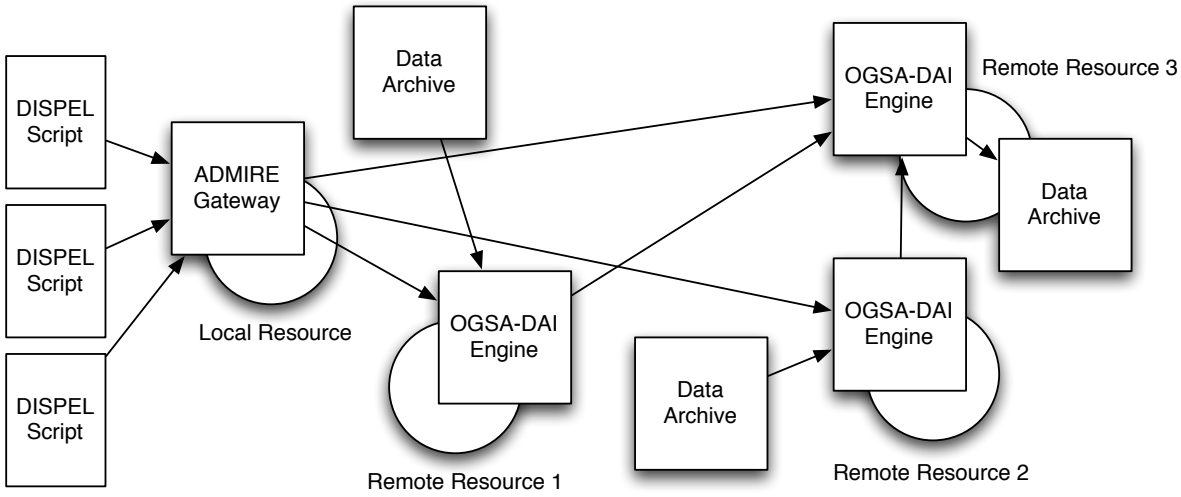


Figure 4: A basic illustration of the VERCE prototype platform in operation. Generated results can be retrieved from the resources either by the gateway, or via some other mechanism.

to that feedback, and to the specific requirements of use-cases formulated by the JRA1 work package, future iterations of the prototype will add, remove and adapt individual elements as necessary to fulfil the objectives of the VERCE project.

The VERCE prototype consists principally of a gateway through which workflow specifications can be submitted and delegated amongst a number of enactment services deployed across a number of different computational nodes, as illustrated in Figure 4. Each service executes their portion of the workflow and transmits any results onwards as necessary; different parts of the workflow may execute in series or in parallel as befits the workflow. Each service is capable of retrieving data from data sources local or remote — the choreography of services is configured to minimise unnecessary data transfer.

The purpose of this prototype is to demonstrate a particular mode of conducting experiments (via the submission of workflows), using a particular base suite of technologies (Dispel and OGSA-DAI). In principle, any number of tools can interact with the gateway (by submitting workflows and querying the gateway in accordance with a published interface), and any number of resources can be used to execute tasks (provided that a suitable interface exists which can channel data and extract it after computation). In practice, the tools, services and resources interacted with, will be chosen based on their applicability to one of a few VERCE-sanctioned use-cases.

3.1 Gateway service

The gateway service for the VERCE prototype is adapted from the one created for the ADMIRE project. The gateway consumes abstract workflow specifications written in workflow language known as Dispel, described below; it then attempts to find implementations for all processing elements before delegating execution of the workflow to available resources provided by the enactment platform.

The prototype gateway is a RESTful Java servlet deployed on Apache Tomcat[28], a lightweight server platform. The gateway can be interacted with as a website, or via HTTP directives as described by Table 1. (Note that while no particular security regime is in place in the gateway component of the prototype, it is possible to initially secure the gateway using HTTPS, which can be used to authenticate the source of requests via credential exchange and filter the list of processes accessible. As HTTPS will not provide the fine-grained authorisation needed when accessing data, in a later stage we will

[28] http://tomcat.apache.org

investigate additional means to more suitable authentication mechanisms, possible with the aid of the registry.) The use of such directives, coupled with the assignment of unique URLs for every process (workflow invocation), ensures that implementing mechanisms for interacting with the gateway is kept very simple, encouraging the construction of specific use-case portals as described earlier.

`http://verce-gateway-base-url/`

- `GET` Get a list of all processes running on the gateway.
- `POST` Submit a Dispel request to the gateway.

`http://verce-gateway-base-url/version/`

- `GET` Gets the current version of the gateway.

`http://verce-gateway-base-url/localResources/`

- `GET` Get a list of all local (to the gateway) resources with which to enact workflow elements.

`http://verce-gateway-base-url/allResources/`

- `GET` Get a list of all known resources with which to enact workflow elements.

`http://verce-gateway-base-url/process-id/`

- `GET` Get a list of all properties of the process identified by *`process-id`*.
- `DELETE` Destroy the process identified by *`process-id`*.

`http://verce-gateway-base-url/process-id/process-property`

- `GET` Get the value of the given property *`process-property`* for the process identified by *`process-id`*.

`http://verce-gateway-base-url/validate`

- `POST` Submit a Dispel request to the gateway for validation (do not execute).

Table 1: The RESTful HTTP interface for the VERCE gateway prototype.

The gateway permits both the submission of workflows for execution and the submission of workflows for validation, in the latter case producing the underlying workflow graph for user perusal. Currently, the gateway uses its own local registry to identify OGSA-DAI implementations of Dispel elements. In future iterations of the prototype, implementation mappings will be drawn from a dedicated VERCE service integrated with a larger metadata model (probably based on recommendations by the EPOS[29] and EUDAT projects).

3.2 Workflow language

Dispel is a data-flow oriented workflow composition language for executing distributed, data-intensive applications[Yaikhom and Martin, 2011]. A Dispel workflow is an abstract network of processing elements through which data can be streamed:

- A *processing element* (PE) describes a persistent computational entity. Every PE has a number of connection interfaces through which data is either consumed or produced. Data is streamed between PE instances via connections made between output and input interfaces.
- A *connection* streams data from one output interface to at least one input interface.

The composition of processing elements charts a workflow graph. Workflow graphs as well as PE and function definitions can be submitted to gateways; a gateway refers to a registry in order to identify suit-

[29] http://www.epos-eu.org

able implementations of workflow components before delegating their execution to available resources. Dispel is imperative and statically-typed, with strict, call-by-value evaluation of expressions.

3.2.1 Dispel characteristics

A principal advantage of Dispel over many other workflow languages (as used in, for example, the Pegasus [Deelman et al., 2005] or Taverna [Oinn et al., 2006] workflow systems) is its imperative nature use of functions, which allows programmers to apply common software composition patterns to synthesise workflows. In particular, the use of imperative constructs such as selection and iteration at execution-time to parametrically generate arbitrarily complex workflow graphs beyond that which can be feasibly constructed using a visual workbench (though it should be noted that this does not rule out the use of such workbenches in many scenarios – and indeed such a workbench already exists for producing Dispel in the Eclipse integrated development environment [Mouat et al., 2011]). In essence functions can be invoked and parameters propagated which affect the specific workflow constructed at execution time.

Dispel also permits the creation of new composite processing elements by specifying them in terms of existing elements; the framing of new logical constructs as variants of existing constructs allows users to specify quite complex behaviours in an intuitive fashion. This potentially could be a very powerful feature if matched with a rich library of component implementations which both permit the ad-hoc synthesis of existing implementations to achieve the functionality of a new logical component, and the registration of specific optimised code for specific compositions of components.

3.2.2 Anatomy of a Dispel script

Dispel uses a notation similar to that of Java. Workflows are constructed incrementally as a Dispel script is interpreted — this also means that functions can be invoked and executed and workflow components defined and exported to a common registry in the same language as is used to build and deploy new workflows. As a consequence, there are generally two kinds of Dispel script; scripts which define and register new components, and scripts which construct and submit workflows for execution. It is possible to define and use workflow elements within the same script, but users often want to register once and use many times, resulting in a natural division of code.

Figure 5 demonstrates the four main stages in constructing and registering new workflow elements:

- The definition of abstract types (`SQLToTupleList`). Abstract PE types can be used to define classes of PE which can be inserted into a workflow; any implemented PE matching the abstract type can be used.
- The specification of a constructor for an abstract type (`lockSQLDataSource`). As well as being used to describe particular classes of PE, abstract types can be implemented using compositions of existing components (such as `SQLQuery`), producing new composite PEs.
- The construction of new processing elements (`SQLOnA` and `SQLOnB`). Multiple implementations of a given abstract type can be created using different constructors or different parameterisations of the same constructor.
- The registration of components for later use. Dependent components are also registered, allowing new abstract types, constructors and constructed types to be recovered for later workflows and shared with other users.

Dispel employs three distinct type systems: *language* types refer to the types of variables in scripts; *structural* types refer to the syntactic structure of data elements streamed between PE instances; *domain* types refer to the semantic (principally ontological) meaning assigned to data elements. Language types (such as `String` designating a string of characters and `Connection` designating a connection object) permit validation of operations in scripts before execution, whilst structural types (`String` designating a string of characters and `[<rest>]` designating a list of tuples of any internal composition) permit validation of

```
package dispel.manual {
  // Import existing PE from the registry and define domain namespace.
  use dispel.db.SQLQuery;
  namespace db
    "http://dispel-lang.org/resource/dispel/db";

  // Define new PE type.
  Type SQLToTupleList is
    PE( <Connection:String::"db:SQLQuery"      expression> =>
        <Connection:[<rest>]::"db:TupleRowSet" data> );

  // Define new PE function.
  PE<SQLToTupleList> lockSQLDataSource(String dataSource) {
    SQLQuery sqlq = new SQLQuery;
    |- repeat enough of dataSource -| => sqlq.source;
    return PE( <Connection expression = sqlq.expression> =>
               <Connection data       = sqlq.data> );
  }

  // Create new PEs.
  PE<SQLToTupleList> SQLOnA = lockSQLDataSource("uk.org.UoE.dbA");
  PE<SQLToTupleList> SQLOnB = lockSQLDataSource("uk.org.UoE.dbB");

  // Register new entities (dependent entities will be registered as well).
  register SQLOnA, SQLOnB;
}
```

Figure 5: A Dispel script which constructs a new workflow element.

connections between workflow components. Structural types can be arbitrarily complex compositions of arrays, lists and tuples; processing elements can consume and produce arbitrary units of data (for example by use of the `Any` type). Domain types (such as `"db:SQLQuery"` and `"db:TupleRowSet"`) can be associated with external ontologies (such as found at `"http://dispel-lang.org/resource/dispel/db"`) and can be freely attached to data elements of any constructed structural type.

Figure 6 demonstrates the process of building and submitting a workflow to a gateway:

- Components (`SQLOnA` and `Results`) are imported from a registry and the instantiated (`sqlona` and `results`).
- The workflow is then constructed by connecting together all component instances and feeding in any initialisation data (`query`). Dispel permits the denotation of arbitrarily complex data streams, bridging the gap between the script and workflow data-spaces.
- Finally, the workflow is submitted by submitting any part of the workflow.

Dispel is oriented around data-flow rather than control-flow. As a result, no specification of how data should be produced or consumed is required; instead, data is pushed out of or pulled into processing elements based on the balance of their respective implemented behaviours, regulated by the enactment platform.

3.3 Enactment platform

Workflows passed through the gateway are currently implemented in the Open Grid Services Architecture, Data Access and Integration platform OGSA-DAI [Jackson et al., 2011]. OGSA-DAI is an open-source framework that executes data-flow oriented workflows consisting of a number of functional units called activities. These activities are analogous to Dispel processing elements.

```
package dispel.manual {
  // Import existing and newly defined PEs.
  use dispel.manual.SQLonA;
  use dispel.lang.Results;

  // Construct instances of PEs for workflow.
  SQLonA  sqlona  = new SQLonA;
  Results results = new Results;

  // Specify query to feed into workflow.
  String query = "SELECT * FROM littleblackbook WHERE id < 10";

  // Connect PE instances to build workflow.
                        |- query -| => sqlona.expression;
  |- "Little Black Book, page 1" -| => results.name;
                        sqlona.data => results.input;

  // Submit workflow (by submitting final component).
  submit results;
}
```

Figure 6: A Dispel script which submits a workflow.

OGSA-DAI is a highly customisable framework that allows developers to plug in activities with specific functionality, for example providing data access to various data resources, such as relational databases, web resources (such as ORFEUS[30] or remote file-systems, as well as providing data processing functionality such as data mining or filtering.

In order to facilitate processing of data close to a data resource or to enable parallel processing of several data streams, a gateway may distribute workflows across several OGSA-DAI instances with data being streamed from one instance to another. The version of OGSA-DAI used in the first iteration of the prototype has a RESTful interface like the gateway and permits binary streaming of data (in contrast to earlier versions which was based on SOAP over HTTP).

It is currently intended that OGSA-DAI be the backbone of inter-compute-centre coordination and data integration; future iterations of the prototype will focus on alternative execution platforms for *intra*-compute-centre coordination. In scenarios where access to HPC resources is required for some part of a data-intensive workflow (as will be required in most if not all VERCE use-cases), it is conceived that the VERCE gateway will be able to delegate the task of bringing data to and from a suitable HPC centre to OGSA-DAI. The processing of that data in the HPC centre will be handled using a different technology (possibly based on recommendations made by the PRACE project). OGSA-DAI will coordinate GridFTP transfers between the HPC-centre operations and the other parts of the workflow.

3.4 Integrating domain-specific code

As described previously, the building blocks of a Dispel workflow are processing elements, the execution of which is optimised and parallelised throughout the nodes of the underlying data-intensive system.

An important aspect of this experimental scenario is the dedicated integration of a domain-specific library of algorithms (Obspy) into OGSA-DAI activities to form the low-level implementation of seismological analysis PEs. This implementation choice was driven by a more general and important requirement of seismology scientists who expect to develop their analysis code using their favourite, trusted libraries, programming languages and APIs.

[30] http://www.orfeus-eu.org/

To allow for the introduction of user-defined algorithms within our workflow engine, we have implemented a framework (see Figure 7) which abstracts over aspects related to the most basic and important needs of this genre of activities. These aspects concern: the configuration of the activity that has to execute a specific analysis code within a particular runtime environment (*e.g.* Python or C); the passage of the parameters and the data between these layers; the production of the metadata in an interoperable format (*e.g.* JSON[31]) and the deallocation of memory and other resources.

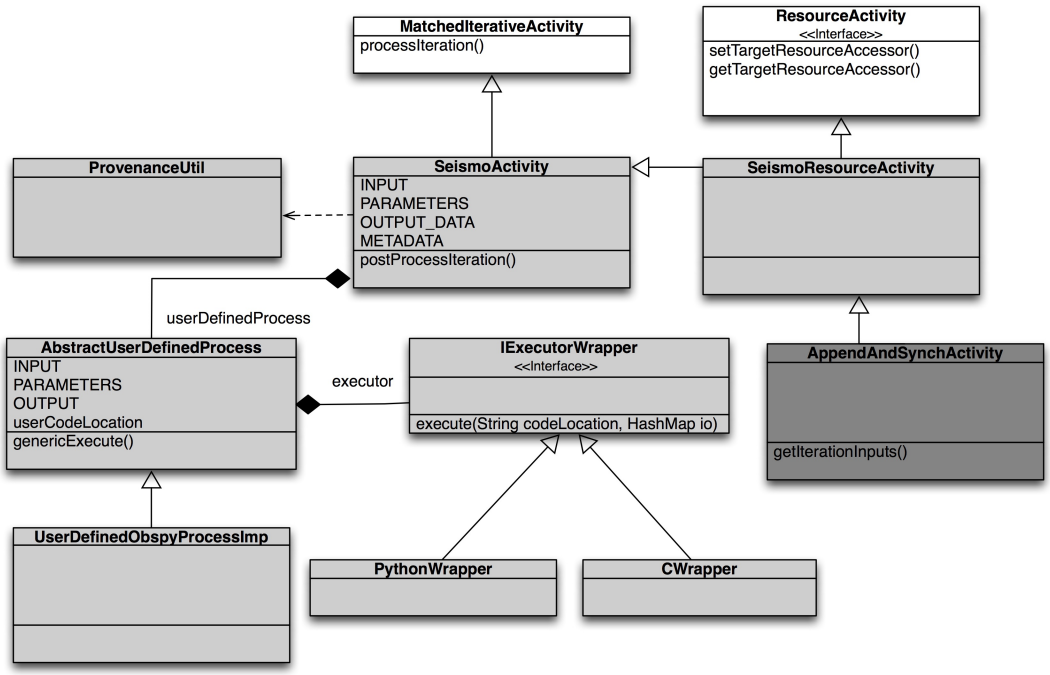


Figure 7: A UML diagram describing the framework developed to enable the integration of a domain-specific library of algorithms within OGSA-DAI.

All of the light grey coloured classes in the UML representation of Figure 7 are part of the framework. The `SeismoActivity` and the `SeismoResourceActivity`, which inherit from a core OGSA-DAI activity class and interface, are the activity classes that implement the processing PEs. A `SeismoActivity` accepts one *input* and a set of *parameters*. Once they are read, they are passed to a `userDefinedProcess` that takes care of organising this information before passing it to an `IExecutorWrapper`, which is responsible for running the script performing the actual transformation on the data. Once the transformation has been applied the result is returned to the wrapper and eventually to the `SeismoActivity` where it gets written to the output connections.

Sometimes a specific task might require to extend the `SeismoActivity`. This happens when the input is not a data stream but, for instance, a list of file references. Take as an example `AppendAndSyncActivity`. Its function is to read a list of Mini-SEED files from a filesystem, then merge and slice the time series contained into the files according to a specific time window passed as a parameter. Ultimately the activity converts the time series obtained into a data stream to be delivered through the `OUTPUTDATA` connection. Notice that the `AppendAndSyncActivity` extends from `SeismoResourceActivity` (Figure 7), which already provides all the required methods to its sub-classes to enable access to a specific resource, a file system in this case.

The `PythonWrapper` is the key component which enables communication between OGSA-DAI and the underlying Python technology. This allows us to bring the continuously-evolving library of algorithms

[31] http://json.org

being developed by seismology experts into the context of the data-intensive platform – the primary requirement of the framework's design.

After investigating the spectrum of the solutions already available to help such integration, we decided to implement the wrapper using Jepp[32], a library designed to embed CPython in Java within a heavily threaded environment, which is exactly what we have to achieve in our experiment. The exchange of information between the Java and Python environments is achieved through the passage of a `HashMap` containing the data that has to be processed and the parameters specified within the workflow. The same `HashMap` is then populated from the script, after its execution, with the processed data and metadata.

This small framework permitted us to assign a behaviour to a PE realised as a `SeismoActivity`, directly within the configuration system of the data-intensive prototype architecture. The adoption of the Spring technology[33] allowed us to configure the properties of such PEs just by editing an XML file, declaring a different configuration for each PE name, specifying which `IExecutorWrapper` to use and which analysis script to execute. In other words, different PE names can be bound to the `SeismoActivity` class, and the behaviour is determined within the `SeismoActivity` at runtime by accessing the Spring context, thus obtaining the desired configuration depending on the name of the PE invoked.

3.5 Infrastructure

A significant part of supporting data-intensive research is the construction and deployment of hardware and system software infrastructure which specifically supports data-intensive computation. In VERCE we are exploring this using the EDIM1 experimental platform in Edinburgh. It has been realised now that the requirements for high-performance data-intensive computing are not always the same as those for traditional high-performance computing. In essence, data-intensive computing emphasises access to storage over processor cycles to the extent that data-intensive research can be better served by making more data available at once to many less powerful processors than by having many stronger processors but less aggregate data availability [Szalay, 2011]. This is because computation is often enacted broadly over extensive corpora of information rather than in intense depth over relatively small datasets (indeed a significant part of the data-intensive research task is to perform computations over all data in order to identify the specific subsets of data upon which more conventional computational science should be targeted).

The 'classical' approach to large-scale computation usually involves the separation of data from computation (storage distinct from processing) — data is staged from a particular location onto and off of the computation platform as and when needed. As data volumes increase, algorithms fail to scale (often assuming in-memory processing to avoid disk access) and considerable time is spent waiting for data to be delivered to a processor. More recent data-intensive projects favour the use of a 'data brick' architecture (for example [Barclay et al., 2004, Szalay et al., 2009]), where a given node (of a cluster) has significant quantities of private storage alongside a modest processing capacity. Data is shuffled between nodes via network connections as necessary, but where possible algorithms and assorted computational processes are brought to the data.

EDIM1 [Martin et al., 2011] is the product of a joint collaboration between the School of Informatics at the University of Edinburgh and EPCC[34], designed to be more 'Amdahl-balanced' than existing data-intensive machines insofar as it offers the greatest possible capacity for applications to benefit from the parallelisation of any components where potential for such exists [Szalay et al., 2010]. EDIM1 also distinguishes itself by its use of commodity hardware to achieve the computational throughput of a much more expensive machine, being an attempt to bring high-performance data-intensive computation into the reach of smaller institutions and research groups in much the same manner as Beowulf clusters

[32] http://jepp.sourceforge.net
[33] http://www.springsource.org
[34] http://www.epcc.ed.ac.uk

were for conventional compute-intensive tasks. It has 120 nodes, each equipped with three 2 TB hard disks and a 256 GB solid-state disk (SSD), thus it has a storage capacity approaching 0.75 PB and can accommodate nearly 500,000 simultaneous streams of I/O operations. The SSDs are particularly useful for non-sequential accesses and for high rates of I/O operations per second.

In the context of the VERCE project, EDIM1 provides a useful test platform on which JRA2 can deploy code provided by JRA1. In preparation, initial executions of the cross-correlation test case described in Section 4 using the first iteration of the VERCE prototype have been performed on EDIM1. It is hoped that various measurements can be made later, which will provide additional insight when implementing the various use-cases being produced in VERCE that will then guide effective and optimal use of the data-path parallelism that EDIM1 offers.

3.6 SDX

Whilst little work has been done on visualising and analysing workflow data products within the project as yet, there has still been progress in this direction parallel to the main prototype strand. SDX stands for Seismic Data eXplorer[35] and was a core component of the RapidSeis project[36], being a desktop application primarily for analysing seismic waveform data. Waveforms are imported from mini-SEED volumes, and phases can be picked manually — the waveforms can be filtered, scaled and visualised in various ways. Specifically, SDX can do the following:

- Waveform visualisation can be filtered by alphabetic, epicentral distance, hypocentral distance and back azimuth.
- SDX allows the user to visualise waveforms in multiple different channels, including more than three at once.
- SDX supports data in a variety of formats. For example mini-SEED files and collections of such files, SDX picks and Binder events.
- SDX can output the waveform in PDF and postscript format for publication.
- SDX can be used as a stand-alone application, or be accessed via a well-defined API.

In addition to the ability to import mini-SEED files locally, SDX has developed functionality which allows files to be archived remotely. For example SDX can archive mini-SEED files from an ArcLink server and request inventory from an ArcLink server, allowing it to access EIDA (the European Integrated Data Archives).

SDX thus represents an exemplar of the specific kind of tool which could be integrated into the core gateway architecture, principally as a means to access and visualise a specific form of common data product which might be produced by VERCE use-case workflows.

3.7 Summary and List of Components

The first iteration of the architecture is represented by the elements in in the table below.

[35] http://www.nesc.ac.uk/talks/920/aeh_neries_061108.pdf
[36] http://research.nesc.ac.uk/node/494

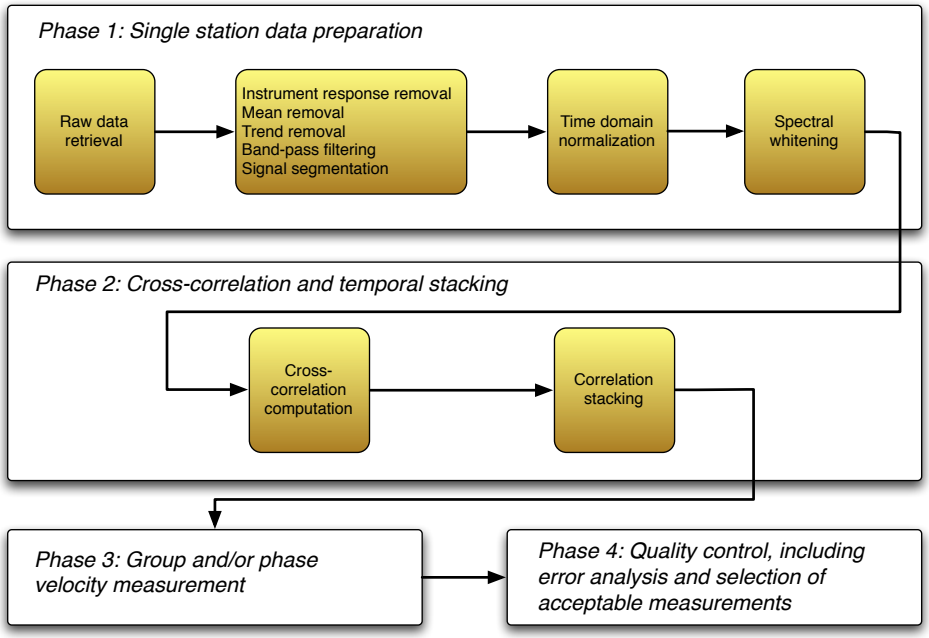


Figure 8: Phases of the seismic ambient noise processing procedure, after Bensen *et al.*

Element	Version	Notes
ADMIRE gateway	2.0.1	Gateway service for the submission and implementation of workflows written in the Dispel workflow construction language. Currently this service uses OGSA-DAI for workflow implementation.
OGSA-DAI	4.1	Open Grid Services Architecture — Data Access and Integration; the enactment platform for workflows submitted to an ADMIRE gateway. Specialises in distributed query processing and data integration. Has been augmented to better use GridFTP data transport.
Apache Tomcat	7.0	Web server environment for deploying Java servlets / server pages. Needed to run the ADMIRE gateway and OGSA-DAI.
GridFTP	5.2	Protocol for bulk data transport. Part of the Globus Toolkit (version 5.2).
GSI	5.2	Grid Security Infrastructure. X509 certificate-based security infrastructure for data movement. Part of the Globus Toolkit (version 5.2).
Java Runtime	1.6	The underlying run-time environment used for OGSA-DAI and ADMIRE.

4 Cross-correlation test case

The VERCE prototype is principally use-case driven; however the principal VERCE use-cases have not been fully specified in time for the first iteration of the prototype. Thus the first VERCE prototype has been configured to execute a relatively simple cross-correlation test case which is believed to exhibit some of the core characteristics of the use-cases to come. This test case, and its implementation using the prototype platform, is described here.

4.1 Background

In this test case we implement core steps of the ambient noise data processing procedure for obtaining surface wave dispersion measurements, described in [Bensen et al., 2007]. This procedure consists of

four phases: (1) single station data preparation; (2) cross-correlation and temporal stacking; (3) measurement of dispersion curves, and; (4) quality control. The rationale for the order of the steps, and the specific techniques selected for each step, is provided in [Bensen et al., 2007].

The ambient noise processing procedure is depicted in Figure 8, with more detail shown for the two phases implemented as part of our experimental scenario. The main purpose of the first phase is to prepare the recorded signals for cross-correlation, by removing all evidence of events such as earthquakes or storms, and instrument irregularities. This leaves only ambient noise for processing. The second phase first cross-correlates all possible pairs of seismic signals. The cross-correlation is often performed on day-long segments; 5 years' of signals from 100 stations split into such segments yield over 16 million pairs for cross-correlation. The resulting cross-correlations are then aggregated, or *stacked*, to correspond to longer time series segments – on average, stacking over increasingly longer time series improves the signal-to-noise ratio.

4.1.1 Configuration of the infrastructure

The current experimental testbed simulates the access to multiple data providers, each hosting an instance of the gateway. This scenario required the data-intensive architecture to deal with heterogeneous data and metadata, and with distributed data sources. This capability is needed because, despite embracing a common standard for raw seismological data called Mini-SEED[37], each institution has its own way of storing and managing data and metadata. Our solution must therefore adapt to different local access arrangements and metadata schemes. In our example we have two different file systems holding the raw data, whereas for the station metadata (gain, response, frequency, etc.) we had to write workflow components to use either a database structure (Seiscomp3 Database[38]) or a well-known ASCII format (RESP files[39]) stored in a file system structure.

4.1.2 Description of the workflow

Ambient noise processing is usually performed in *time windows* applied to data from a predefined set of *seismic stations*. These are considered the main inputs, together with a number of domain-specific *parameters*, which are provided to the system via portals or other types of interface, ultimately via a Dispel document. In this context the choice of the optimal time window length is relevant because it could affect the final result of the analysis. Ideally it is desirable, from a user's point of view, to be able to specify the duration of the overall period of interest as a single window. In order to address these requirements we introduced a `TimeSampler` as the first element in our workflow. This PE (processing element) behaves in some sense as a clock for the system, chopping up the overall interval of analysis (in general months or years) into sub-intervals (typically hours or days), thus giving the user the ability to adjust these parameters. By presenting intermediate results to the users, the system enables them to make dynamic parameter adjustments and thereby to steer the overall computation.

Figure 9 provides a high-level view of the main blocks composing the ambient noise workflow, integrating two geographically distinct data sources. The sampling phase generates a list of time windows $[T_1 \dots T_n]$; which we can refer to as the single time T_i. Subsequently the user's query, typically a list of stations $[S_1 \dots S_n]$ plus the time T_i, is sent in parallel to several `DataAndMetadataExtraction` modules. The main task of these modules is to interface with the data sources providing a direct-access layer and preparing a uniform output for the next steps. This addresses the heterogeneity in the data systems, handling the different data and metadata storage and management mechanisms, as shown in Figure 10. For this purpose, specific implementations have been created and then mapped into a generic PE (`WFRetrieve`)

[37] http://www.iris.washington.edu/data/seismograms
[38] http://www.seiscomp3.org
[39] http://www.iris.edu/KB/questions/69/What+is+a+RESP+file

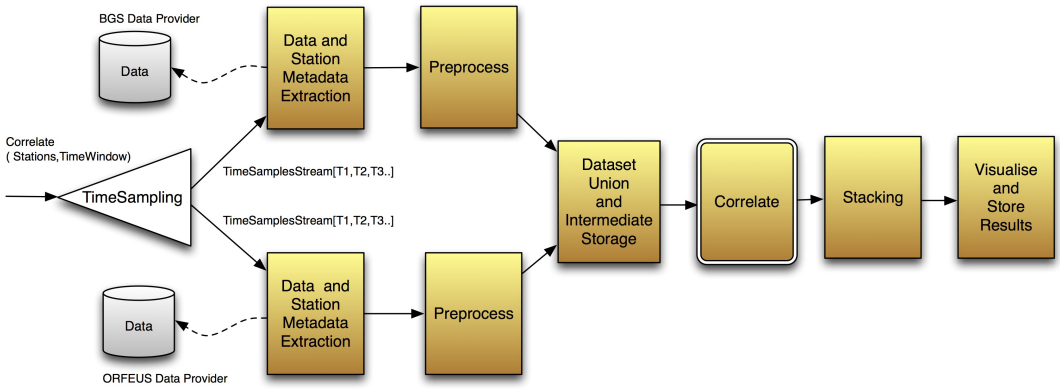


Figure 9: The ambient noise workflow using two seismic archives.

whose specific implementation depends on the particular gateway at which it is deployed, thus providing a consistent and transparent interface to the seismic data and metadata.

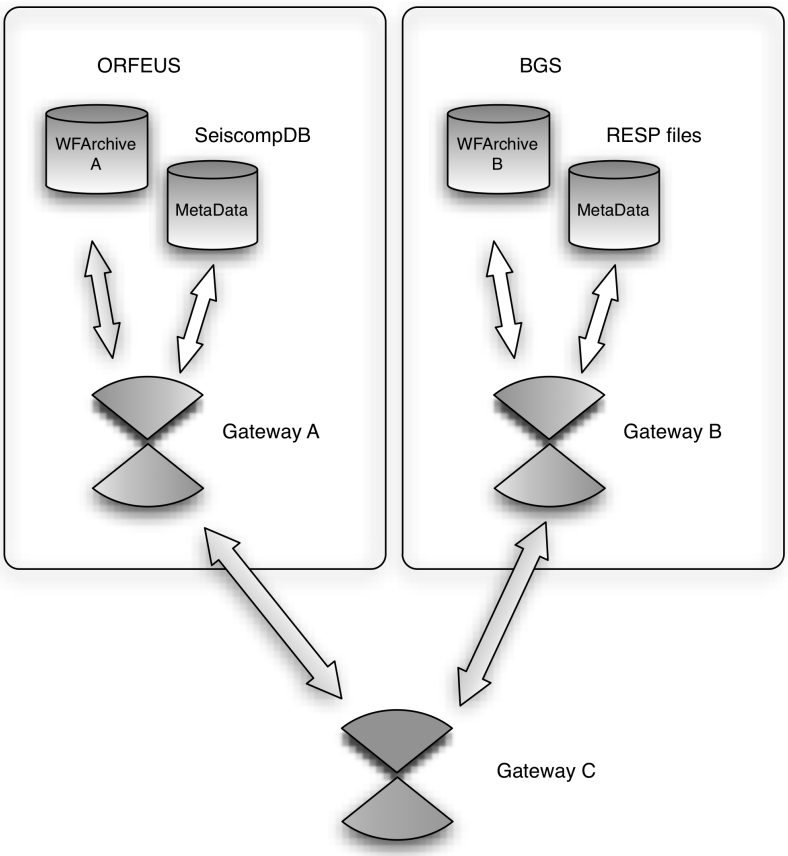


Figure 10: System diagram of a possible ambient noise correlation infrastructure deployment.

Once data and metadata have been fetched from the archives, they can be streamed into subsequent processing elements. In order to minimise data shipment we moved some preprocessing steps, including *filtering*, *normalisation*, *instrument response removal*, *whitening* and *decimation*, close to the data themselves. These are recurrent operations that do not entail large computation and that contribute to reducing the data flow. For each operation an activity has been implemented making reuse of existing domain-specific libraries.

```
/* Extracts and synchronises time series based on the time window in input*/

PE<SeismoTrace> extractWFTimeSeries(String datares,String provenanceRes,
 String channelCode, String station, String network)
 {
  Tee tee = new Tee;
  ListConcatenate lc1 = new ListConcatenate;
  WFRetrieve retrieve = new WFRetrieve;
  WaveformAppendAndSync appsyn = new WaveformAppendAndSync;
  SeismoMetadataTuple metastoreex1 = new SeismoMetadataTuple;
  SeismoMetadataTuple metastoreex2 = new SeismoMetadataTuple;
  |-provenanceRes-|=>metastoreex1.resource;
  |-provenanceRes-|=>metastoreex2.resource;
  |-datares-| => retrieve.resource;
  |-datares-| => appsyn.resource;
  String cha="ch="+channelCode;
  String sta="sta="+station;
  String net="net="+network;
  tee.output[0]=>lc1.input[0];
  |-repeat enough of [cha,sta,net]-|=>lc1.input[1];
  lc1.output=>retrieve.parameters;
  retrieve.metadata=>metastoreex1.metastring;
  retrieve.wflocations=>appsyn.input;
  tee.output[1]=>appsyn.parameters;
  appsyn.metadata=>metastoreex2.metastring;
  metastoreex1.datasetid=>metastoreex2.stepbackid;
  return PE(<Connection window=tee.input> =>
  <Connection output=appsyn.output;Connection datasetid=metastoreex2.datasetid>);
 }
```

Figure 11: DISPEL code for retrieving seismic traces from distributed datasources.

The `WFExtraction` PE is a composite processing element consisting of two elements, the `WFRetrieve` PE and an `AppendAndSync` PE, which are responsible for shaping the data stream based on the requested time window. Figure 11 shows how we implemented the extraction phase using Dispel.

The preprocessing steps are then assembled in a pipeline structure, such that at each step the stream is modified in some respect. This approach provides great flexibility and allows the data-analysis expert to extend, change and re-shuffle the workflow by just rearranging the pipeline or plugging in new PEs. Figure 12 illustrates the function responsible for assembling the pipeline `makeArrayPipeline` – it accepts as parameters the generic `SeismoStage` PEs (line 3), which are in turn the result of three different functions shown at lines 25, 30 and 33.

At the end of the pipelines we obtain preprocessed streams which are almost ready for cross-correlation. However, some effort is still required to merge the streams together choosing the most appropriate traces. Because each data centre is completely independent from the other, they could, for example, provide data that originated from the same station. Therefore, we must take into account the existence of duplicates and set up mechanisms for an optimal choice based on some quality parameters. In our case we chose the copy with the least total "Number of Gaps" (NoG) present in the traces as the discriminating factor, which perhaps seems quite a rough measure in that it does not consider the duration of these gaps. Nevertheless, we assume that the data are pre-checked by the data centres.

A merger PE, `StreamHarvester`, has been implemented to accomplish this task: after selecting the best n trace samples matching the initial query, the emerging data can be pushed further and eventually preserved in an intermediate storage system that is useful for recovery and successive re-elaboration of the workflow. We believe that in order to set up a robust and stable automated data processing procedure it

```
/*Builds the preprocessing pipeline*/

PE <SeismoPipeline> makeArrayPipeline(PE <SeismoStage>[] TheStages) {
 Integer len = TheStages.length;
 SeismoStage[] stages = new SeismoStage[len];
 PE<SeismoStage> Stage = TheStages[0];
 stages[0] = new Stage;
 for (Integer i = 0; i<len-1; i++) {
   DeliverToNull tonull = new DeliverToNull;
   PE<SeismoStage> Stg = TheStages[i+1];
   stages[i+1] = new Stg;
   stages[i].output => stages[i+1].input;
   stages[i].datasetid => stages[i+1].stepbackid;
   stages[i].metadata=>tonull.input;
  };
 return SeismoPipeline( <Connection input = stages[0].input;
        Connection stepbackid = stages[0].stepbackid> =>
         <Connection metadata = stages[len-1].metadata;
         Connection datasetid=  stages[len-1].datasetid;
         Connection residue =  stages[len-1].output> );
  };

/*The current implementation includes the following three stages:
instrument and mean removal, filtering and decimation, prewhitening*/

PE<SeismoStage> removeInstrumentMeanAndNormalize(
 String station,String channel,String datares,String provenanceRes)
{
...
}
PE<SeismoStage> filterAndDecimate(String provenanceRes){
...
}
PE<SeismoStage> whiten(String provenanceRes){
...
}
```

Figure 12: DISPEL code for building the pre-processing pipeline.

is vital to adopt proper data quality control measures. The traceability of the experiment together with quality monitoring is a good indicator for the validation and verification steps.

We are now ready for the primary objective of our analysis: pair-wise cross correlation of the pre-elaborated time series. A symmetric cross correlator, as shown in Figure 13, could fulfill the requirement.

It is important to notice though, that from a data-analysis expert's point of view the correlation phase is similar to the Dispel snippet in Figure 14. Here the effort is focused mainly on expressing the functionality descriptively, without any knowledge of the underlying infrastructure. How this feature is then enacted depends on the specific deployment characteristics.

The last part of our data-analysis is summarised in Figure 15. After computing the cross correlations of the n streams coming from the preprocessing pipelines $[P_1 \ldots P_n]$ they are made persistent in an appropriate storage space. Following the approach already discussed, each set $[XC_1 \ldots XC_k]$ of correlations refers to a particular time T_i generated from the initial slicing into sub-periods. It is actually common practice in this type of problem to partition the computation into small time periods. However this requires an additional step before providing the results: in order to enhance the information by increasing the signal-to-noise ratio, the intermediate correlations are stacked to assemble the overall duration of interest.

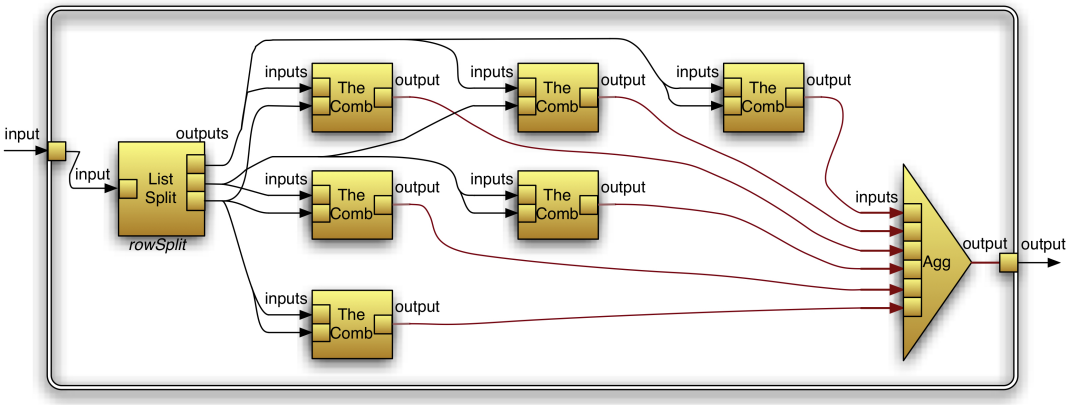


Figure 13: The correlation phase of the workflow exploiting symmetry.

```
/* Computes cross correlation between input1 and input2:
the maximum delay is parametrised (timeShift)*/

 PE<XCorrelationF> crossCorrelate(
 String timeShift, String datares, String provenanceRes){
    WFXCorrelator xcorr =new WFXCorrelator;
    |-datares-|=>xcorr.resource;
    |-repeat enough of ["tshift="+timeShift]-|=>xcorr.parameters;
    return PE(<Connection input1=xcorr.input;Connection input2=xcorr.input2> =>
     <Connection output=xcorr.output;Connection metadata=xcorr.metadata>);
   };

...

PE<XCorrelationF> XCorrelation =
 crossCorrelate(timeShift, correlationRes, provenanceRes);
 Integer n=stations.length;

for(Integer x=0;x<n;x++){
  for(Integer y=x+1;y<n;y++){
    PE(<Connection input> => <>) PlotterPE =
    plotCorrelation(stations[x],stations[y],channel,processedRes);
    XCorrelation xCorr = new XCorrelation;
    PlotterPE plot=new PlotterPE;
    WFXCorrelationStacker stacker=new WFXCorrelationStacker;
    ...
    xCorr.output=>stacker.input;
    stacker.output=>plot.input;
    plot.output=>rr.input;
  }
}
```

Figure 14: DISPEL code for performing the cross-correlation of n streams.

The final results are presented as images showing the correlation functions between pairs of stations. Therefore there is an immediate visual feedback for the seismologists that can be very helpful in adjusting and tuning the correlations while they are still running. In Figure 16 we show an example of different outcomes obtained by changing just one parameter in the correlation phase. Eventually these images could be dynamically updated as new streams are computed and presented in a portal, providing a very useful monitoring system.

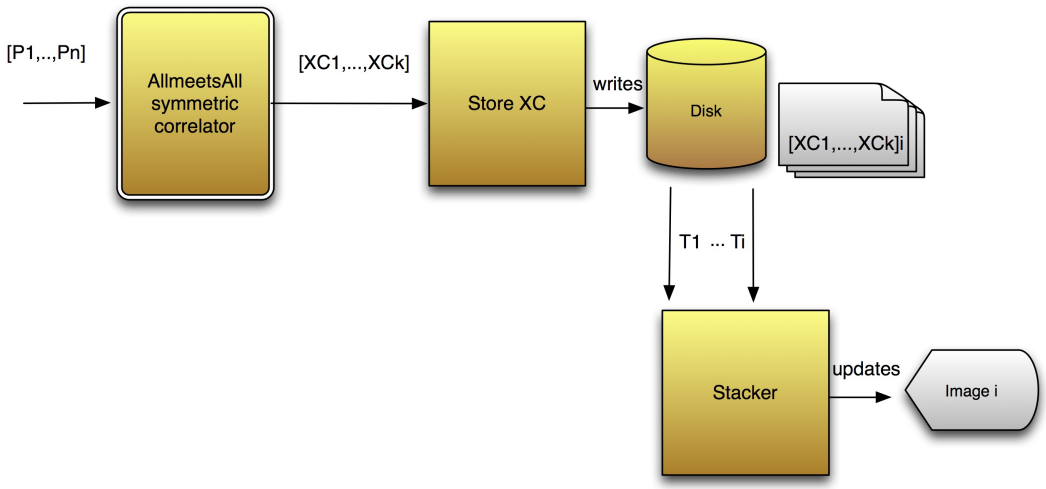
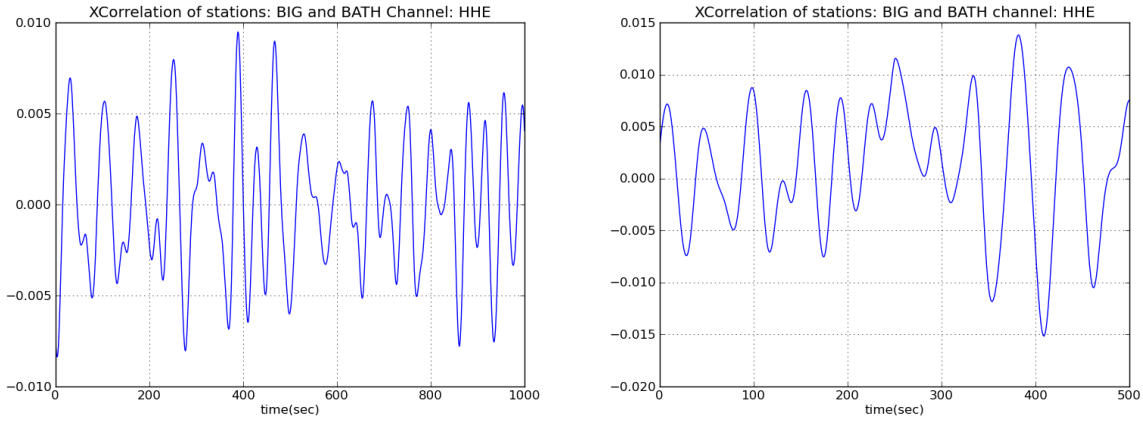


Figure 15: Storage and stacking of the correlation results.



(a) (b)

Figure 16: Cross-correlation using different maximum time delays (in seconds) (a) tshift = 500s (b) tshift = 250s.

In this section we showcased the efficacy of the current prototype platform via a cross-correlation test use-case. In particular, the DISPEL language has been shown to be expressive enough to describe the processing elements involved, providing enough abstraction so as to encourage the desired separation of roles. In VERCE, we do not anticipate that scientists (domain experts) will have to write their own DISPEL scripts – they will instead be provided with higher-level user interfaces tailored to classes of general use-cases identified. However, in this preliminary evaluation, we show that DISPEL would be useful both to data-analysis experts for designing processing elements and setting out template workflows and as a back-end to more friendly user interfaces. The systematic study of additional use cases, as these will become available, will provide us with insight for building such interfaces as well as for potentially adapting DISPEL to new requirements.

5 Additional technology survey

As already discussed, the VERCE platform relies on a number of existing technologies, and will soon rely on even more as it is further developed. The next steps should take into account the relevant external technology for three reasons:

1. to stimulate discussion within the VERCE project about what we should do and how we should do it in future phases;
2. to inform further selection or modification of the existing VERCE architecture components by JRA2; and
3. to be a source of further components to be selected by JRA2 in a future architectural iteration.

We address this by a general scene-setting preamble and then a list of components. These are rather broad at the moment with the intention of serving purpose (1) above. A more focused list, of a few carefully chosen options will be set up to serve purposes (2) and (3).

We survey three of the technological areas which must be further explored; in this particular iteration focusing on workflow systems (which may complement, inspire, or if necessary replace the current gateway technology used in the prototype), data transport (necessary for the underlying management of data used by workflows) and job submission systems (necessary for interaction with managed resources such as high-performance computational clusters).

5.1 Workflow Management

In this section we discuss the current state of workflow management technology. It briefly describes the availability and current use of such technology, discusses perceived required functionality of this technology and non-functional considerations. JRA2 will work with other work packages to identify and confirm workflow requirements.

5.1.1 Workflows and management systems

This section introduces workflows, life cycles and the automated systems that manage them. This also introduces the terminology to be used in this and later deliverables when discussing workflow management.

A *Workflow* comprises a set of tasks or operations, dependencies between the tasks, and the data resources to be used during the enactment of that workflow. The execution of a workflow often produces more data. A simplified data-mining workflow is illustrated in Figure 17. The dependency in a workflow is one of two types: a control flow which represents the passing of execution privilege between computational tasks, or a data flow which simply passes data from one task to another. This distinction, and the advantages that a workflow with one flow type may have over a workflow with flows of the other type, are discussed in more detail in Section 2.2; suffice it to say at this stage that many scientific workflow systems favour data flows between computational tasks, as this more easily describes their experimental scenario. It also offers the workflow engines for optimising the location and scheduling of the tasks.

Figure 18 depicts a *workflow lifecycle*, adapted from [Deelman et al., 2009]. It comprises four phases: workflow composition, resource mapping, execution and monitoring, and provenance data capture. Various tools and technologies may be used during each phase. In general, a *workflow management system* provides the tools for workflow composition and resource mapping, while a *workflow execution engine* is responsible for executing workflows on available data and compute resources.

Workflow composition — construction of a high-level workflow known as an *abstract workflow*. This workflow identifies the software components and data needed for a particular a computation, without details about the physical resources used in its the execution. The information about the components and

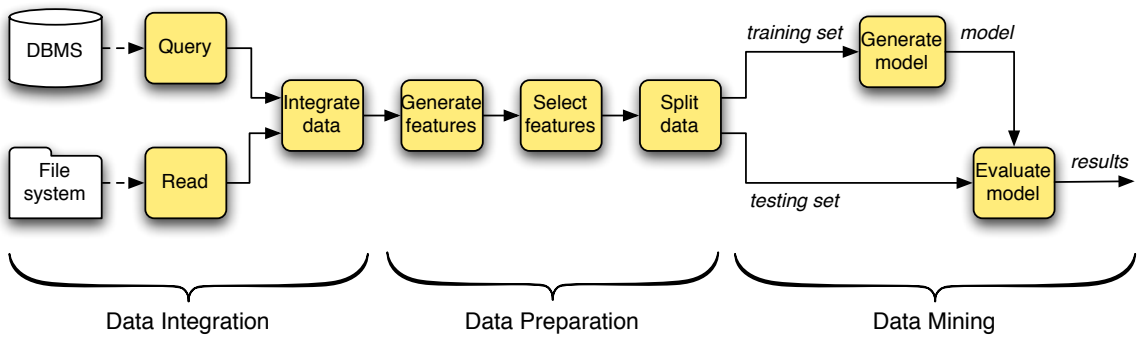


Figure 17: An example data mining workflow.

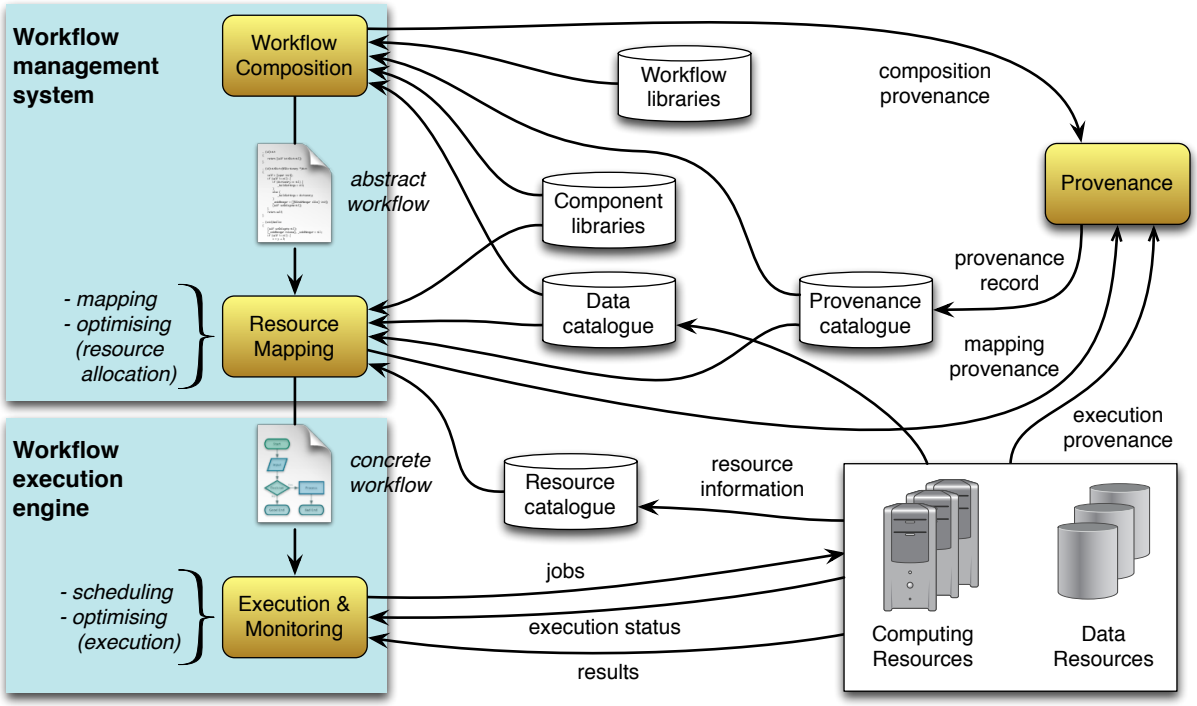


Figure 18: Workflow lifecycle adapted from [Deelman et al., 2009].

data are stored in *component libraries* and *data catalogues* respectively. Abstract workflows can be created from scratch by defining workflow tasks using a particular representation from a workflow system. For instance, DAGMan [Couvares et al., 2007] allows users to create a workflow as a DAG (Directed Acyclic Graph) and execute it sequentially. Triana [Taylor et al., 2007] provides a graphical-connection tool for data-flow creation which is later represented in simple XML. Some workflow creation systems, such as Wings [Gil et al., 2007b], provide users with an existing workflow template from which to build their complex abstract workflows. Alternatively, users can select existing workflows from workflow libraries such as ^{my}Experiment, a virtual research environment that supports collaboration and sharing of workflows and experiments [Roure et al., 2009].

Resource mapping — mapping of abstract workflows into executable plans named *concrete workflows*. Concrete workflows specify the suitable resources from the execution environment to be used in a particular computation. Workflow mappers obtain resource information from a *resource catalogue*. Selection of appropriate resources may affect the overall execution of a workflow; a good mapping can improve resource usage efficiency and execution performance. Thus, various optimisation techniques have been applied to refine the executable plan. For instance, one of the workflow refinement techniques used in

Pegasus is to perform workflow reduction using available data products [Deelman et al., 2004] – by consulting a replica catalogue (i.e. catalogue that gathers information about the locations of data products), Pegasus determines which intermediate data are available and removes these redundant tasks from the workflow.

Execution and monitoring — enactment of the concrete workflow in the execution environment and monitoring of the performance of workflow execution. The workflow execution engine is responsible for scheduling the tasks on assigned resources and receiving back the execution results. Some workflow execution engines such as Condor[Litzkow et al., 1988, Thain et al., 2005] work independently and can integrate with different workflow management systems; others are built within the framework of the workflow management system, such as ASKALON [Fahringer et al., 2007]. There is plenty of scope for optimisation during the execution phase. For instance, Singh *et al* examine the various factors that affect the completion of astronomy and biology workflows [Singh et al., 2005], and their findings show that job submission rate, scheduling interval and dispatch rate all influence the execution performance. Based on the information collected, they have optimised these applications and reduced the completion time. Abramson *et al* use their expertise in parametric modelling using Nimrod toolkit[40] to develop a new workflow orchestration module in Kepler to dynamically spawn parallel threads to optimise the execution of massively parallel parameter sweep workflows [Abramson et al., 2008].

Provenance capture — recording of the history of workflow execution and data creation. Scientific experiments are very likely to be repeated with different parameter values and data sets, and the experiments may be improved with each iteration. Moreover, provenance information may be useful for recovery purposes should the execution system fail to complete the workflow for any reason, and in determining future optimisation approaches and parameters settings. Several workflow management systems provide services to manage provenance data — *e.g.* Kepler[Altintas et al., 2006] and Pegasus[Kim et al., 2008]. [Stevens et al., 2007] describes how provenance is used to manage knowledge of *in silico* experiments. Davidson *et al.* discuss the provenance challenges that arise in scientific workflow systems to capture the provenance of complex data and workflow evolution [Davidson et al., 2007]. Many provenance frameworks tailored for scientific workflows have been developed over the years, e.g. [Bowers et al., 2007, Lim et al., 2010, Simmhan et al., 2008]. A good survey of data provenance techniques in e-Science is presented in [Simmhan et al., 2005]. In [Liew et al., 2011], the authors propose a systematic way of capturing measurement data as part of performance database (PDB) stores.

Other workflow lifecycles have been proposed in both business and science including [Görlach et al., 2011, Ludäscher et al., 2009]. However, they all make the common observations below:

- phases of a lifecycle are viewed from the scientists' perspective, i.e. steps involved in creating and running workflows;
- scientists play the main role, i.e. they compose, operate and analyse workflows;
- scientific workflows are exploratory, i.e. it is common to reuse existing workflows and refine them in a trial-and-error manner;
- scientific workflows tend to be repeated, i.e. scientists re-run the same workflows with different parameters settings and/or datasets; and
- run-time monitoring is important, i.e. scientists monitor the progress and may decide to abort or suspend the execution.

These observations help to identify the common requirements for workflow management and execution systems, discussed in Section 5.1.3.

[40] http://messagelab.monash.edu.au/Nimrod

5.1.2 Workflow systems in use by VERCE

A survey was conducted of VERCE partners to identify their current use of workflow technology. Tables 2 and 3 lists the workflow systems that have been mentioned in the survey results, and includes a few others that appear to be in common use in various scientific communities. Detailed documentation and reviews of most of these systems are available from the links provided.

The IT and specialist computing VERCE partners report a few workflow systems in use, though since they service the wider scientific community it is as yet unclear whether the VERCE earthquake and seismology community actually use these. These tools include the UNICORE Workflow System[41], Taverna[42], TRIANA[43] as well as KNIME[44]. Another potential option that will merit more in-depth future investigation is the technology adopted by MAPPER, including GridSpace2 and RealityGrid Steer for workflow deployment [Groen et al., 2011].

The seismology VERCE partners have reported that they mainly use Python and scripting languages for defining their scientific workflows. Though there is an awareness of more sophisticated options available such as Kepler and Taverna, there appears to be limited actual experience of the use of automated workflow systems.

A more in-depth discussion will be necessary within the next iteration of the architecture development to identify and prioritise the workflow management related requirements. This discussion will have a well-founded starting point, as common detailed requirements have been identified in other scientific communities, and several of them are likely to be required by VERCE participants. These generic requirements are presented in the next section.

As has been mentioned in this report, the VERCE architecture builds upon the foundations laid by the ADMIRE prototype data-intensive architecture. An integral part of the ADMIRE architecture is Dispel, a data-flow oriented workflow composition language for executing distributed, data-intensive applications. Dispel may be used both for composing workflows, and for mapping their enactment onto available resources. The characteristics and benefits of Dispel are introduced in Section 3.2.

Dispel has been used to implement the test case detailed in Section 4. It is not expected that Dispel in its present form will meet all of the needs of the VERCE community. Options for resolving the gaps include further development of Dispel, integrating other existing technology into the architecture, and a combination of further development and integration. The chosen route will depend mainly on the requirements of the VERCE consortium, clarified and prioritised on an ongoing process during this project.

Key to VERCE's success will be the design and implementation of libraries of processing elements and services that provide many of the common functions required by the scientific community. The further development of Dispel within VERCE will focus around encouraging such activities. Another important feature, which will aid the adoption of the VERCE prototype platform, is the provision of a harness to facilitate the definition of processing elements and related computational entities in programming languages such as Python and Java.

5.1.3 Workflow system requirements

Many studies [Deelman and Gil, 2006, Gil et al., 2007a, Goble and Roure, 2009, Zhao et al., 2008] have described the role of workflow management and execution systems in managing workflows, as well as

[41] http://www.unicore.eu/
[42] http://www.taverna.org.uk/
[43] http://www.trianacode.org/
[44] http://www.knime.org/

Name	Workflow composition and submission	Workflow mapping and execution	Domains	Latest release	Reference
UNICORE (UNiform Interface to COmputing Resources) Workflow System[a]	Workflow control constructs and features offered by Workflow Engine are accessed by graphical Eclipse-based URC (UNICORE Rich Client) and UCC (UNICORE Commandline Client) using an XML-based language. Portals and clients for submitting workflows in other languages may be built	Two-layered architecture: Workflow Engine offers a range of workflow control constructs and features, it deals with high-level execution of workflow, translating a workflow into a set of tasks to be run in a specific order; Service Orchestrator is responsible for executing individual tasks in a workflow, identifying best resources for each task, handling job execution and monitoring.	Non-specific	UNICORE Workflow 6.4.0, Mar 2012	UNICORE is part of the EMI (European Middleware Initative) project. It is maintained by Jülich Supercomputing Centre, Germany. (http://www.unicore.eu)
Kepler	GUI-based Kepler workbench	Built on Ptolemy II component assembly framework for modeling, designing and simulating concurrent, real-time embedded systems. Workflows are described in MoML (Modelling Markup Language for XML) using Ptolemy 'actors' (defining processing tasks), while the execution of the workflow is driven by Ptolemy 'directors' (defining scheduling and execution)	Includes ecology, chemistry, and geology (GEON)	Kepler 2.3, Jan 2012	Maintained by cross-project collaboration and led by originators of the project: University of California at Davis, Santa Barabara, and San Diego. (https://kepler-project.org)
Taverna[b]	Taverna Workbench used for composing or searching for workflows. Workflow components are web-based or local (e.g. data manipulation and file I/O) services.	Taverna Engine used for enacting workflows. Also powers the Taverna Workbench, and the Taverna Server for remote execution of workflows. Workflows are described in t2flows or SCUFL2 (Simple Conceptual Unified Flow Language 2)	Includes biology, chemistry, medicicin, music, meteorology	Taverna Server 2.3	Part of the myGrid project specialising in data and knowledge-intensive e-laboratories. Taverna is hosted at the School of Computer Science, University of Manchester, UK. (http://www.taverna.org.uk)

Table 2: Related workflow management systems.

[a] In use at VERCE IT partner sites LRZ and CINECA by PRACE users. EMI project end-date, Mar 2013

[b] In use at VERCE partner site CINECA, for biomedical applications. Guaranteed funding to 2014

Name	Workflow composition and submission	Workflow mapping and execution	Domains	Latest release	Reference
Pegasus (Planning for Execution in Grids)	Includes Java and Python APIs, Pegasus GUI, Wings, Triana, Xbaya, CGSMD (Collaborative Genetic Studies in Mental Disorders) portal with predefined workflows	CWG (Concrete Workflow Generator) maps abstract workflow (represented in DAX - DAG in XML) into a concrete one (executable DAG for DAGMan); DAGMan and Condor used for scheduling and execution.	Includes astronomy, bioinformatics, earthquake science, ocean science	Pegasus 4.0, Mar 2012	Project of the USC Informaton Sciences Institute and the Computer Science Department, University of Wisconsin Madison, USA. http://pegasus.isi.edu
Meandre	GUI-based workbench, and ZigZag language for description of workflows	ZigZag files compiled to .mau (Meandre archive unit) files which may then be executed by a Meandre engine (on different environments: local machine, server, cloud, other)	Humanities	Meandre 1.4.9, May 2011	Developed by the Software Environment for the Advancement of Scholarly Research (SEASR), funded by the Andrew W. Mellon Foundation at NCSA. http://seasr.org/meandre
K-Wf Grid (Knowledge-based Workflow system for Grid applications)[a]	User interaction through web portal that includes a GWUI (Grid Workflow User Interface) for construction of workflows	Mapping and execution responsibility of the GWES (Grid/Generic Workflow Execution Service), using XML-based language called GWorkflowDL (Grid Workflow Description Language)	Use cases included flood forecasting, traffic management and enterprise resource management	Unknown	K-Wf Grid project ended Mar 2007, project website reference no longer correct (http://www.kwfgrid.eu)
Gridspace2	Virtual laboratory framework, where users may submit workflows via scripts written in popular computer languages, such as Python, Perl and Ruby. Gridspace2 is currently under investigation for its relevance to the VERCE community.	Unknown	Unknown		http://dice.cyfronet.pl/products/gridspace

Table 3: Related workflow management systems (continued from previous page).

[a]GWES was in use at VERCE IT partner site LRZ a few years ago, but current status unknown. It has been enhanced/used in projects other than K-Wf Grid (http://www.gridworkflow.org/kwfgrid/gwes-web/)

the challenges that arose. From these and other work we derive a list of requirements that need to be met in order to support the full workflow lifecycle:

Collaboration — Scientific research is collaborative. Scientists across organisational and geographical boundaries share data, services (i.e. codes and applications) and computing resources in running experiments. For instance, the Southern California Earthquake Center (SCEC)[45] is a community of over 600 scientists from more than 60 institutions world-wide that conducts geophysical research to develop a comprehensive understanding of earthquakes. Their research in seismic hazard analysis requires incorporating physics in their geological models and running a variety of earthquake simulation applications on Grid-based computing environments. To create a collaborative environment, workflow systems must provide tools to describe the tasks and their dependencies as workflows, support the execution of the workflow in the correct order in the distributed environment and manage the data and metadata [Maechling et al., 2007]. Another example of an emerging scientific community built around the sharing and incremental improvement of workflows is myExperiment[46].

Usability — As a tool to assist scientists managing their computational experiments and data analyses, usability is a fundamental requirement to assure users' satisfaction. Scientists expect an effortless and efficient way of using workflow systems to conduct their experiments, where they can focus on the scientific discovery without having to consider the low-level execution of the computation tasks. The Telescience project [Lin et al., 2007] demonstrates the role of portals as a workflow controller, enabling users to manage data, services and collaborative tools through a simplified interface.

Reusability — Scientific workflows are exploratory [Barga and Gannon, 2007, Ludäscher et al., 2009]. Workflows are constructed by scientists and change frequently to incorporate their observations during each iteration of experiments. For instance, scientists add/remove experimental steps, or try different approaches to perform the tasks, where existing sub-workflows/components are reused in constructing new workflows. During a rerun, scientists may want to make use of results from previous executions of the sub-workflows, whenever is possible. Ludäscher *et al* introduced the idea of a "smart rerun" that only executes the part of the workflow that is affected by the changes [Ludäscher et al., 2006]. Furthermore, scientists often rerun the same workflow with different datasets. Supporting reusability requires a well structured provenance capture system about the workflows and the generated data.

Reproducibility — Reproducibility is the core of scientific method, where scientists repeat techniques and analysis methods done by others in validating their hypothesis [Gil et al., 2007a]. The reproducibility requirement is different from reusability. Its main focus is on enabling users to rerun the workflow to obtain as close to identical results as possible. However, an essential element of the solution to both requirements is the same, which is provenance. For modern experiments running on distributed and heterogeneous environment reproducibility is difficult to achieve. It requires workflow systems to keep track of the tasks, parameter values, data sources, computing resources, mapping configuration, and, the data generated at each step of the execution. Workflow systems need to capture provenance data throughout the workflow lifecylce.

Flexibility — Workflow systems have to be flexible in integrating and accessing heterogeneous resources. Jones [Jones, 2007] highlights the importance of flexibility to support biodiversity e-Science research, which is challenging the limit of different resources that workflow systems can cope with. Typical biodiversity research needs to access different data and catalogues, e.g. species catalogues, geographical data and climate data, in different data standards, and analyse these using different tools that are not interoperable. Some data are proprietary with access restriction, and some tools are accessible at particular locations, e.g. web service. Thus, the workflow systems need to provide an integrated environment to manage the diversity of data and associated tasks, such as handling the communication between different tools and aggregating distributed datasets.

[45] http://www.scec.org/
[46] http://www.myexperiment.org/

Scalability — The scaling requirement should be viewed from four dimensions [Gil et al., 2007a]: number of tasks in workflows, number of workflows, number of resources (both computing and data), and number of users. The SCEC CyberShake project [Callaghan et al., 2010] has tested the scalability of workflow systems, dealing with approximately 840,000 individual tasks. The workflow can be split into 80 sub-workflows, each with more than 10,000 tasks. A single run of the CyberShake workflow uses up to 800 processors on the TeraGrid, and generates 417,886 seismogram files (approximately 9.5GB of data). A single SCEC scientist can execute more than 2 million jobs a week. This large-scale experiment has challenged the capability of workflow systems in handling the execution pipelines, staging files from/to between the sites, and optimising such large-scale execution.

Dynamic — Workflow systems should support the exploratory nature of science and provide a dynamic execution environment that adapts to changing context and infrastructure. Scientists often do not have a fixed analysis in mind. They look at the results from the initial stage to decide the later analysis steps, thus, the execution is result-driven. Gannon *et al* share their experience with Linked Environments for Atmospheric Discovery (LEAD) project[47], in creating an interactive way for mesoscale weather prediction [Gannon et al., 2007]. The simulation phase of the weather prediction cycle is a good example of result-driven execution. The system introduces finer computational meshes for further computation on the geographical areas that have shown interesting results during the first execution that scan across the entire landscape. The LEAD project also demonstrates resource adaptability, where the system will eliminate a computation that fails to track the evolving weather, and allocate the computing resources to other simulation instances.

Robustness — Fault tolerance is an important requirement, especially in the distributed and heterogenous environment. The execution may fail due to parameter misconfiguration, missing input data, network disruptions, etc. Workflow systems should provide a recovery path for workflow execution. Simmhan *et al.* suggest a few ways to enhance fault resilience in Trident, such as garbage collection (e.g. terminate unfinished tasks and reverse update to restore original database), data replication and rerunning failed workflows from the provenance information [Simmhan et al., 2009].

5.1.4 Next steps for workflows

Three concurrent activities will be undertaken to refine understanding and to shape future VERCE-architecture workflow developments in the next few iterations of the prototype:

1. analysis of the use cases identified by NA2 and JRA1 to understand where workflows are used already, where they could be used additionally to benefit scientific usability and data-handling;
2. analysis of how far the DISPEL definition and technology should be extended to meet VERCE requirements that are currently not satisfied; and
3. identification of a short list of other workflow management systems that should be further analysed for possible use in the VERCE e-Infrastructure.

5.2 Data Transfer

A common problem for many scientific applications is the replication of (often large) data-sets from one system to another. Typically this requires reliable transfer (protection against transmission errors) such as provided by TCP, typically access control based on some sort of authentication, and sometimes confidentiality against eavesdroppers, which can be provided by encryption. There are many protocols and tools which can be used for file transfer, some of which are outlined here in alphabetical order[48]:

[47] http://portal.leadproject.org/

[48] The reader is remind that this is a large list and is diverse to provoke discussion. A much more focussed list based on a careful selection will be used for the next steps.

Bulk data-movement technologies

Protocol	Description
BBCP	is a point-to-point network file copy application written by Andy Hanushevsky at SLAC as a tool for the BaBar collaboration. It is capable of transferring files at approaching line speeds in the WAN. BBCP seems to be a very similar utility to bbftp below, with the exception that it does not require a remote server running. In this behaviour, it is much more like a SCP in that data transfer requires only user-executable copies on both sides of the connection. For larger files, the multi-streaming transfer utility BBCP is recommended.The BBCP utility is capable of breaking up your transfer into multiple simultaneously transferring streams, thereby transferring data much faster than single-streaming utilities such as SCP and SFTP. Source: http://www.slac.stanford.edu/~abh/bbcp/
BBFTP	is a multi-stream file transfer application that is recommended in place of SCP or FTP for the data transfer of large files over long distances. It provides a secure control channel over SSH and allows data to be transferred in cleartext to reduce overhead in unnecessary encryption. Source: http://doc.in2p3.fr/bbftp/
BitTorrent	is an example of a peer-to-peer file-sharing protocol. It employs local control mechanisms to optimise the global problem of replicating a large file to many recipients, by allowing peers to share partial copies as they receive them. Source: http://www.bittorrent.org/
FDT	is an Application for Efficient Data Transfers which is capable of reading and writing at disk speed over wide area networks (with standard TCP). It is written in Java, runs on all major platforms and it is easy to use. FDT is based on an asynchronous, flexible multithreaded system and is using the capabilities of the Java NIO libraries. FDT can be used to stream a large set of files across the network, so that a large dataset composed of thousands of files can be sent or received at full speed, without the network transfer restarting between files. Source: http://monalisa.cern.ch/FDT/
FTP	is one of the earliest protocols used on ARPAnet and the Internet. The File Transfer Protocol supports simple file operations over a variety of operating systems and file abstractions, and has both a text and a binary mode. FTP uses separate TCP connections for control and data transfer. Source: http://en.wikipedia.org/wiki/File_Transfer_Protocol and http://www.ietf.org/rfc/rfc959.txt
GridFTP	GridFTP is an extension of the standard File Transfer Protocol (FTP) for use with Grid computing. It is defined as part of the Globus toolkit. GridFTP provides reliable and high-performance file transfer for Grid applications. GridFTP addresses incompatibility between storage and access systems. GridFTP provides a uniform way of accessing the data, encompassing functions from all of the different modes of access, building on and extending the widely accepted FTP standard. FTP was chosen as a basis for it because of its widespread use, and because it has a well-defined architecture for extensions.

continued on next page

continued from previous page

Protocol	Description
	Source: www.globus.org/grid_software/data/gridftp.php. The OGF standard may be found at http://www.ogf.org/documents/GFD.20.pdf and http://www.ogf.org/documents/GFD.21.pdf
HTTP	the Hypertext Transfer Protocol is the basic protocol used by the World Wide Web. It is quite efficient for transferring files, but is typically used to transfer from a server to a client only. Source: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol and http://www.w3.org/Protocols/rfc2616/rfc2616.html
RCP	is Berkeley's Remote Copy Protocol, a convenient protocol for transferring files between Unix systems, but lacks real security beyond address-based authentication and clear-text passwords. Therefore it has mostly fallen out of use. Source: http://unixhelp.ed.ac.uk/CGI/man-cgi?rcp
Rsync	is a software application and network protocol for Unix-like and Windows systems that synchronises files and directories from one location to another while minimising data transfer using delta encoding when appropriate. An important feature of rsync not found in most similar programs/protocols is that the mirroring takes place with only one transmission in each direction. The rsync command can copy or display directory contents and copy files, optionally using compression and recursion. Source: http://en.wikipedia.org/wiki/Rsync
SCP	The file-transfer application of the SSH protocol. It provides various modern methods of authentication and encryption, but its current implementations have some performance limitations over "long fat networks" that are addressed under the SSH topic. Source: http://en.wikipedia.org/wiki/Secure_copy
UDT	A UDP-based bulk transfer protocol, optimised for high-capacity (1 Gb/s and above) wide-area network paths. It has been used in the winning entry at the Supercomputing'06 Bandwidth Challenge. Source: http://udt.sourceforge.net/
WEBDAV	the Web-based Distributed Authoring and Versioning systems constitutes a set of HTTP extensions for remote file management, synchronisation and collaborative authoring of web resources. Currently it is mainly used for synchronising e-mail and calendar desktop applications with repositories in the cloud. Because of its wide availability and its reliance on HTTP, a standard and well-understood protocol, it has also been considered for use for bulk file transfer by initiatives such as EGI. Source: http://webdav.org/

At this stage in the project, most attention is being given to GridFTP (Grid File Transfer Protocol) as the *de facto* standard for transporting large data files over TCP or UDP between distant locations in Grid contexts. GridFTP is part of the core Globus Toolkit [Foster, 2006]. VERCE currently uses the GridFTP server and client of version 5.2.0 of the Globus Toolkit — preference is being given to the distribution disseminated by IGE (Initiative for Globus in Europe)[49].

5.2.1 Message Transfer

Apart from moving large data sets between remote sites, in VERCE there is also the need for providing support for another class of data transmission, that of relatively short messages. Such messages would

[49] http://www.ige-project.eu

facilitate orchestration, the collection of diagnostic information and coordinate provenance. They could be transmitted between Grid or HPC sites, end-user web or desktop client applications, etc. Reliable and timely message delivery would often be needed, as some of these messages would be coordinating the run-time of job execution, instead of them aiding batch or background components.

Candidate systems for message streaming, queueing and consumption are rabbitMQ[50], Apache Foundation's ActiveMQ[51], stormMQ[52], etc. These systems are currently being evaluated with respect to their suitability to the VERCE initiative as well as regarding their adoption by related initiatives, projects and tools.

5.2.2 Next steps for data movement

The following activities will be undertaken; they will overlap as their schedule will depend on progress and availability of colleagues to discuss both requirements and technical constraints.

1. Analyse the use cases to identify the requirements for bulk data movement (in conjunction with NA2 and JRA1).
2. Analyse the constraints imposed by site-regulations, e.g. for security, that limit choices (in conjunction with SA1, SA2 and SA3).
3. Observation of current choices during VERCE campaigns.
4. Selection of a carefully selected short list of bulk-data movement technologies that need to be considered as additions (in consultation with EUDAT and EGI).
5. Open discussions with VERCE colleagues on the requirements for reliable coordination messages (in conjunction with SA2's consideration of monitoring).
6. Selection of a carefully selected short list of message-passing systems for further analysis if the requirement is established.

5.3 Job Management

While technologies like OGSA-DAI provide a generic platform for enacting distributed tasks, the classes of computation required to perform the experiments of interest to VERCE themselves require the delegation of specific computationally-intensive sub-tasks to high-performance computation resources such as are situated at BADW-LRZ, CINECA and SCAI. Such facilities require that tasks are submitted according to a well-defined protocol, such that tasks can be scheduled using an existing job submission system. In order to better understand the requirements for such systems (not simply in terms of submission protocol, but also in terms of how complex activities are best deconstructed to be partially or wholly executed on HPC resources), a survey of some job management systems has been carried out by JRA2 – the following systems are listed in alphabetical order[53].

[50] http://www.rabbitmq.com/
[51] http://activemq.apache.org/
[52] http://stormmq.com/
[53] This is very much a preliminary list. It will be reduced drastically to those that are actively in use at the sites used by the seismology application codes, and then further reduced to match the priority use cases.

Job management technologies

Manager	Description
Computing Center Software	CCS is a system developed at German institute Paderborn Center for Parallel Computing. Basic functionalities are: scheduling based on making reservations (like Maui), features for check pointing of all job forms (in development), graphic interface, integrated cluster monitoring systems, defining advanced job scheduling policies (reservations, backfilling). http://pc2.uni-paderborn.de/
Condor	is a job management system which is primarily intended for high-throughput applications. Basic advantages include; CPU harvesting, a special language ClassAds for job and client description, and possibilities of process checkpointing and migration. Condor allows linking of several clusters in such a way that one cluster, when it needs additional resources, can forward tasks to another. Condor allows on demand computing (*i.e.* certain users can request a certain group of resources at a certain time). Condor is suitable for Grid systems because integration into Grid systems can easily be achieved, and it contains elements for authentication and linking with Globus facilities. Another advantage is that it can be used on a Windows platform, as well as the typical UNIX platform used by many Grid systems. A disadvantage is that Condor is intended primarily for high-throughput computing; even though it can be used or performing parallel jobs, that is not the primary way it operates. Check pointing is possible only for programmes which are in translations connected with Condor booklets. Generally speaking, check point features refer only to serial applications. http://research.cs.wisc.edu/condor/
Load Sharing Facility	LSF is a product of company called Platform. Platform LSF is an extremely advanced system, which contains many features a JMS should have: it comes in a form of a distribution with all components of cluster middleware, it supports almost all platforms, has possibility of defining different forms of job scheduling (advanced reservations, pre-emptive scheduling), advance resource management (check pointing, job migration, error resistance, load balancing), advanced graphic interfaces for access to cluster functionalities, integration with Grid systems and possibility of linking several clusters. http://www.platform.com/workload-management/high-performance-computing
Loadleveler	is an IBM job management system intended for IBM AIX platform. Its architecture is based on Condor, therefore they have similar features. Despite that Loadleveler is intended primarily for high-performance computing, and it allows working with parallel jobs. Loadleveler is currently being adjusted to the Linux/x86 platform. http://www-03.ibm.com/systems/software/loadleveler/
Maui	is an open source system intended for job scheduling. Maui is used primarily as job scheduler in other JMSs, but it can be used independently (see Clubmask cluster distribution). Maui can be integrated with the following job management systems: OpenPBS, PBSPro, Torque, SGE, LSF and Loadleveler. Maui has the following functionalities: job scheduling based on making reservations and defining extremely complex job scheduling policies. Maui allows scheduling according to the following principles: fair share, backfilling, fixed resource reservations, assigning priorities based on job characteristics (e.g. owner or a group of owners, job size related to space or time). In addition to that, Maui contains a set of tools for retrieval of job execution history and different forms of job statistics and resource usage. Maui's disadvantage is that efficiency of job scheduling depends on how well users describe their requests. The request-definition problem is not specific to Maui. Maui doesn't appear to have any organisation continuing its development.

continued on next page

continued from previous page

Manager	Description
	http://www.nsc.liu.se/systems/retiredsystems/grendel/maui.html
Moab	Moab Workload Manager is a commercial product of Cluster Resources aiming at extending the functionalities of job management systems. Basic functionalities of Moab Workload Manager are: possibilities of defining advanced job scheduling policies, integration of more job management systems and other resource management systems (e.g. data warehouse management system), interfaces for integration with other cluster middleware systems (e.g. accounting and cluster monitoring systems), intuitive interface in a form of a web portal (Moab Access Portal) and support for integration with Grid systems. Functionalities related to job scheduling are identical to functionalities offered by Maui. http://www.clusterresources.com/
OpenPBS, PBSPro and TORQUE	Portable Batch System is an old and well-tested job management system which has been divided into OpenPBS and PBSPro. OpenPBS is an open source system intended for academic use, however further development has been given to PBSPro, making the future of OpenPBS uncertain. OpenPBS development does continue though under the name Torque (Tera-scale Open-source Resource and QUEue manager). Advantages of PBS are: interface for integration with parallel libraries, queue defining feature, advanced job scheduling and a feature for using an arbitrary scheduler module. PBS contains an interface which allows parallel libraries to start parallel processes using PBS services. In that way, PBS has total control over parallel processes and can collect data on resource usage. Currently, LAM/MPI and MPICH parallel libraries are supported. PBS allows simple creation of global queues with certain features. Using queues, it is possible to easily define job scheduling policies. PBS disadvantages are as follows: further development of OpenPBS is uncertain; the graphic interface of PBS allows only working with jobs and monitoring client utilisation — configuration of clients, queues and other components of the system is done by commands from the command line or by adjustment of text files — the graphic interface for working with jobs is impractical; as opposed to SGE, PBS does not represent clients as queues therefore there are no possibilities of defining sensors and complex client features; furthermore, OpenPBS has no features for assigning job fields as SGE does. As opposed to OpenPBS, PBSPro comes with support and additional functionalities. Some of additional functionalities are: CPU harvesting, enhanced Scheduler module, integration with Globus services, and support for Windows operating systems.

continued on next page

continued from previous page

Manager	Description
	http://en.wikipedia.org/wiki/Portable_Batch_System
SLURM	SLURM is an open-source resource manager designed for Linux clusters of all sizes. It provides three key functions. First it allocates exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (typically a parallel job) on a set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work. SLURM is in use at some VERCE sites and will be evaluated further during the course of the project. https://computing.llnl.gov/linux/slurm/
Oracle/Sun Grid Engine	SGE is Sun (now Oracle)'s open source job management system. A basic advantage of SGE is that it has an intuitive graphical interface, which enables access to all functionalities of the system. Another advantage is that cluster clients can be encapsulated into queues, thereby allowing complex descriptions of certain clients and implementations of arbitrary client load sensors. Using arbitrary sensors and client configuration SGE can be used for CPU harvesting. SGE allows calendar definitions. Calendars are used to define periods in which certain clients will not be available. That in turn allows using SGE for automatic cluster extension, with resources which are available only at certain time periods (*e.g.* at night). Users are allowed to define numerous job characteristics, including the choice of required (hard) and optional (soft) conditions. SGE allows starting job fields (*i.e.* multiple execution of the same programme) where, in each instance, different arguments can be given. http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html

DRMAA[54] or Distributed Resource Management Application API is a high-level Open Grid Forum API specification for the submission and control of jobs to a Distributed Resource Management (DRM) system, such as a Cluster or Grid computing infrastructure. The scope of the API covers all of the high-level functionality required for applications to submit, control, and monitor jobs on execution resources in the DRM system. The main objectives of DRMAA are:

- Develop an API specification for the submission and control of jobs to one or more Distributed Resource Management (DRM) systems.
- The scope of this specification is all the high level functionality which is necessary for an application to consign a job to a DRM system including common operations on jobs like termination or suspension.
- The objective is to facilitate the direct interfacing of applications to today's DRM systems by application's builders, portal builders, and Independent Software Vendors (ISVs).

Currently no specific job submission system has been chosen for integration into the prototype, the concern of the first iteration being more with allowing basic workflow execution and submission via gateway. Nonetheless, interaction with job-submission systems installed on various computational resources (specifically high-performance compute clusters) is considered to be of great importance in future iterations. Currently the focus of attention is on interoperability with UNICORE[55], Grid resources using Globus[56] or gLite[57], and the DRMAA-compliant job-submission systems embedded in those frameworks.

[54] http://www.drmaa.org/
[55] http://www.unicore.eu
[56] http://www.globus.org/toolkit/
[57] http://glite.cern.ch

5.4 Next steps

The sections above collate an initial survey of potential components that may be used in subsequent iterations of the VERCE project. This information may be used in two ways:

1. to provoke thought about functionality that has already been identified as useful in other e-Science projects; and
2. as a place to look when choosing a candidate component to recommend to SA2.

The following steps will be undertaken before such recommendations are made.

1. Clearly established agreement with NA2 and JRA1 will have confirmed that the functionality is necessary.
2. For a putative recommendation, an update of its current definition, maintenance support and use in VERCE project sites.
3. A review as to whether it, or an alternative, is being taken up in cognate projects, including EUDAT, IGE, MAPPER, EGI and EarthCube.
4. A review as to whether it is compatible with any standards mandated by GEOSS, INSPIRE , EPOS and the International Federation of Digital Seismograph Networks (FDSN)[58].
5. A preliminary evaluation of possible functional and non-functional properties and acceptability to sites that would need to deploy it.

6 Future development

The development of the VERCE architecture will respond to the specification of the use-cases by NA2 and JRA1, which the VERCE e-Infrastructure is intended to support. As research continues and the feedback of the VERCE consortium is factored in over the course of the project, new technologies will be integrated into the prototype and a number of missing issues not yet addressed by this particular iteration of the prototype will be given their due attention. In this section can be found an overview of some of the questions and technologies being considered for the next iteration of the VERCE prototype. The priority for the next iteration is to link more closely with the scientific-gateway R&D in SA3, this will include support for catalogues and registries including provenance (Section 6.1). A close collaboration with JRA1 and SA3 will drive the process of integrating data-intensive activities with HPC application runs (Section 6.2). Implementing these will require selecting and achieving subgoals, such as those illustrated below.

For provenance:

1. a choice of provenance purpose
2. a choice of provenance and catalogue representations
3. a choice of provenance and catalogue store
4. a choice of the requirements on computational elements to report provenance
5. a choice of reliable message queueing to deliver those reports (see Section 5.2.1)
6. a choice of provenance and other metadata aggregators;
7. a choice of tools for viewing and using provenance information, and
8. a choice of mechanisms and tools for harvesting metadata for inclusion in catalogues.

For HPC integration:

1. harmonisation of usage policies
2. mutually acceptable single-sign-on and AAA arrangements[59]
3. co-reservation, co-scheduling and urgent computing mechanisms

[58] http://fdsn.org/

[59] We will watch discussions about this between EGI and PRACE concerning the possibility of using the technology

4. data-interchange mechanisms
5. heterogeneous workflow orchestration
6. integration and presentation of job monitoring information; and
7. co-ordinated clean-up after both successful and failed jobs.

Given the complexity of both of these tasks, it is important for VERCE to pursue the following.

1. Interact with cognate projects to see if the required solutions can be imported or delegated and whether synergy can be achieved in running, improving or, in the worst case, implementing the required functionality. For foundational mechanisms, such as the representation of provenance or the reliable message passing, the investigation of options should include major and reliable distributed system technology campaigns, such as those at the Apache Software Foundation[60].
2. Accept that partial solutions and *ad hoc* work-arounds may be necessary for several iterations of the architecture and platform, pending internal and external developments.

6.1 Provenance and recovery

The processing needed to perform the analysis required by the use cases will carry information about the workflow process which will not only contribute to the results, but will also be necessary for replication of experiments. It is interesting to understand what happens during the execution of a workflow, where several distributed components are used to accomplish complex computational tasks.

6.1.1 Workflow Preservation

Workflow preservation for the purposes of replication of results, tracking of changes, versioning, as well as for additional purposes to be negotiated with the Earth sciences community can be achieved through effective *provenance* structures. There are different types of provenance [Cheney et al., 2009], developed independently by different communities in order to fulfil different purposes. The overall types of provenance, however, can be summarised as follows:

History would facilitate answering questions relating to authors, contributors as well as to tracking changes to content, data, policies, etc.

Explanation would facilitate justification for obtaining certain intermediate or final results. This class of provenance functionality could, for instance, help document research or other decisions during the course of large experiments.

Derivation would involve reasoning over execution traces, types of resources, etc., in order to answer questions such as "How did we get this result?".

For the purposes of VERCE, the first two categories of provenance – History and Explanation – should be investigated for inclusion, also in line with other related projects and initiatives so as to maximise platform compatibility. The last category – Derivation – would probably introduce prohibitive amounts of computational and communication.

Most scientific workflow systems (such as VisTrails[61] and Taverna [Oinn et al., 2006]) are aware that preserving selected data about an execution is extremely relevant when validating the obtained results,

LCAS/LCMAPS (https://wiki.nikhef.nl/grid/Site_Access_Control), which was recently integrated into the Globus Toolkit as this mechanism is also integrated in gLite. Globus can now interface with higher-level authorisation tools like Argus (http://www.switch.ch/de/grid/argus/index.html).

[60] http://www.apache.org/

[61] http://www.vistrails.org

which is a fundamental requirement of any scientific computation. This is achieved by collecting provenance information, a task which opens, as mentioned in [Davidson, 2008], several challenging research questions. Storing and accessing provenance information allows for later examination of the derivation process. It tells the user details about the data quality at a certain point of an experiment, and can also be used to re-execute part of the workflow using partial results already available in intermediate data stores.

Attempts to answer problems related to information management infrastructures, workflow provenance visualisation and database integration are looking at solutions that converge to a single and interoperable framework (The Open Provenance Model) [Moreau et al., 2008, Ellqvist, 2010] for recording provenance data, where a higher level of abstraction aims to improve interoperability among these workflows systems towards a better understanding of the execution of a scientific computation that connects different tools, processing services, databases and computational infrastructures. More recently, the Provenance Working Group of the W3C[62] has been trying to establish a common data model for provenance of entities and resources, for primary use on the web. Their model, PROV-DM, is now in the final phases before formal publication[63].

Preservation is therefore achieved through access to results which remain persistent over time; but this does not help in providing a deep understanding of the process, if these results are not reproducible. Therefore the data has to be coupled also with the workflow description and its configuration at the time of the execution, together with low-level provenance traces. This package will be built incrementally as the required subsystems are identified in various parts of VERCE and integrated into the platform.

Several studies on workflow preservation are currently on going within the Wf4ever project[64]. VERCE is planning to follow these studies to have a better insight of what could pragmatically help in achieving these goals.

6.1.2 Provenance traces

Some of the most recent workflows systems are moving towards the adoption of enactment platforms that actually perform the computation itself, rather then interconnecting external services. The previously mentioned OGSA-DAI, as well as Yahoo S4[65], are two of the current examples of such platforms, providing a general framework for the implementation of production-level workflow tools for distributed analysis and processing of continuous data streams.

They can provide a collection of processing elements which are scalable and should be configurable within their implementation logic. The scientific code of several processing elements used in VERCE is likely to be developed directly by the domain experts, and it has to be validated and dynamically deployed into the production infrastructure. Adopting such workflow systems gives the opportunity to collect execution information at a very high resolution, within the core of the scientific calculation; a task that would be much more complicated or even impossible if systems focusing only on the orchestration of independent services were adopted. Provenance traces about the execution of processing elements will be useful both for the user (who will be able to improve his or her algorithms) or for the system in case of failure (which can use the information to recover processes). The latter needs information related to the availability of intermediate data, meaningful error messages describing why the execution failed, and indications of whether performances can be improved by automating the delegation of tasks to available resources.

On the other hand, there may be multiple ways in which experimenters might choose to interact with the VERCE platform, not excluding the possibility of even completely bypassing web portals or workbench

[62] http://www.w3.org/
[63] http://dvcs.w3.org/hg/prov/raw-file/default/model/prov-dm.html
[64] http://www.wf4ever-project.org
[65] http://incubator.apache.org/s4/

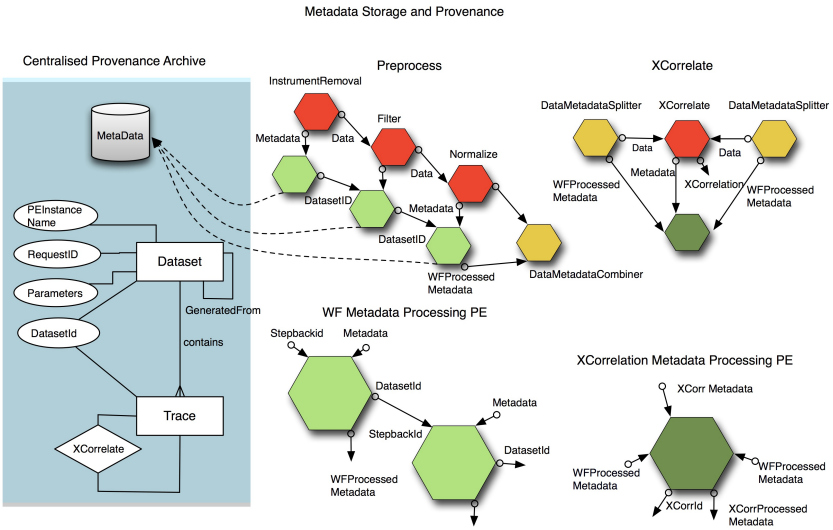


Figure 19: A centralised approach to storing and processing the provenance of distributed PEs.

tools, while still using remote computing resources and data stores through, for instance, the command-line. In this case, gaps in the provenance records would become inevitable, unless a more generic and workflow-independent approach to keeping provenance was adopted. Prioritising provenance and providing additional supporting services is a matter of negotiation with the Earth sciences community, as the VERCE platform would need to be as unobtrusive as possible in order to be useful and sustainable. Supporting parts of provenance through a workflow-independent subsystem would also make compliance with related initiatives and projects easier, to the benefit of VERCE users.

In [Tan, 2004] we find a description of two main strategies in the recording of provenance traces – the eager and lazy approaches. While the first is often based on the collection of annotations at each step of processing, carrying them along until the end of computation, the latter suggests that the processing of provenance should happen only when needed. It assumes also that the provenance information of a certain result can be inferred by running a query on a database that is obtained as a transformation of a source database performed by the executor of a computational task.

Our choice for collecting provenance will take into account several aspects of the lazy approach, leaving the workflow developer to choose at which point in the computational process it is relevant to store provenance metadata. When needed, this information is extracted by a set of dedicated provenance modules, processed and stored, as for error messages. The kind of provenance information collected can be also considered as *retrospective provenance* [Clifford et al., 2008], where we record not only information about processing steps, but also information related to the execution environment. Provenance modules can be connected together in order to keep track of the story of metadata produced from analysis components to which they are attached — obtaining in such a way the provenance traces (see Figure 19). At the end of computation, a query on the relational database can provide all of the relevant provenance information that led to a specific output result.

A set of potential steps to follow towards integrating provenance tracing into VERCE is as follows.

1. Ongoing discussions with NA2 and JRA1 work-packages in order to clarify the type of provenance functionality they would require.
2. Output from point 1 will be integrated with the requirements the data-engineers envisage for diagnosis and recovery.
3. The cost in enactment time and storage will be estimated and initial policies for comparison, archiving and discarding provenance information will be drafted.

4. Continuing experiments with the experimental provenance subsystem, shown in Figure 19, to help with steps 1 to 3, *inter alia*, and to pioneer a first version of provenance capture (at least) for seismic noise correlation during the next six-month iteration of the architectural design.
5. Monitor contemporaneous research and operational deployments via the standards WG, the provenance research community contacts and cognate projects in Europe and world-wide, e.g. in the framework of GEOSS, EarthCube and IFSN.

6.2 Interaction with High-Performance Compute facilities

The requirements of HPC facilities tend to be more restrictive in comparison to the distributed facilities provided by infrastructure like EGI. To access expensive HPC facilities, (*e.g.* PRACE resources), one has to expect various imposed limitations that are inherent as a result of security requirements.

One of the first barriers is limited direct access to the HPC facilities or services. The severity of the restriction is dependent on the HPC centre. In the strictest case there are centres which would not allow direct access from any shared resources to their HPC facilities. Data-management services (*e.g.* GridFTP), are not allowed to run on the actual HPC facility. Instead, an external machine is used to provide the required service. Additional wrapper scripts are provided to help users move the required data from the external machine to the HPC facility. At some less restrictive centres, a shared file system is mounted on both the external machine and the HPC facility to allow file sharing. At some more relaxed HPC centres, direct access to the HPC facility is possible. However, such centres are uncommon. As such, one of the issues to take care of regards how to overcome the limited access issue.

The second barrier is the limited access to the outside world. To prevent abuse of the HPC facilities, access to the internet at large is restricted to ports that are used by mandatory services. HTTP ports (*e.g.* TCP ports 80, 8080, 443, 8443) are typically blocked. As such, direct access to and from seismic data archives will prove to be challenging.

Another barrier to overcome is the selection of tools and services. Most HPC centres do not support 'unnecessary' tools and services for which functionalities are already provided by other tools and services. Most tools and services are being screened not only for functionalities but for security. Since HPC resources are expensive, any compromise to the system due to badly implemented security is viewed seriously.

Finally, it is important to use the expensive HPC facilities in the optimal way. Performance of any application codes below a 3% peak performance is deemed as unsuitable for the particular HPC resource. Due to the varying architectures of HPC machines, the performance of application codes can be greatly affected by the machine specific optimised libraries, compiler flags and the efficient setup of computing cores in neighbouring nodes on a system.

The (non-)existence of a global file system is another point to considered. In order to optimally use a fast transfer protocol on HPC facilities, in addition to the network bandwidth, a fast file system is necessary. However, not all fast file systems are globally mounted on HPC facilities. Many file systems are either limited in size or are in fact non-persistent. Data will be removed as soon as a job has completed, which makes it necessary to have some other storage to which re-usable data products have to be transferred immediately.

6.3 Development of the next iteration of the prototype

Thus far most attention has been given to a putative 'hub' of the VERCE architecture — that being the gateway (based on ADMIRE technology) through which a myriad of potential tools and services can submit workflows to be executed using a selection of different technologies on a range of different resources. The emphasis going forward for the JRA2 work package must be twofold:

1. the hub must be reinforced, addressing any lacking functionality as identified by the SA2 work package; and
2. interfaces to the tools, services, resources and technologies, which are supposed to be connected via the hub, must be designed and constructed, as motivated by the use-cases produced by the NA2 work package and identified by the JRA1, SA1 and SA3 work packages.

Given these two overarching objectives, there exists a number of *potential* points of focus for the next six months.

Workflow component management: For the gateway approach being taken by VERCE, there is a need for a robust registry for workflow components which can map the logical components used in Dispel to actual code executable on subscribed resources. This registry needs to be able to:

- accept code contributions from any legitimate source and automatically update the registry with their latest definition;
- track different versions of the same code in order to ensure replicability of experiments;
- choose between multiple candidate implementations for the same logical component based on context.

The registry should be tightly integrated with VERCE data-management services in general, particularly those that regard replication and provenance. The registry functionality will need to support registry and catalogue requirements in the VERCE science gateway.

Foreign code integration: At a basic level, any tool or service can be interacted with given the production of a suitable interface, represented in a workflow by an activity which consumes suitable input or produces suitable output. This process extends to existing code written in various languages such as Python and C++; as already demonstrated in the cross-correlation test case, it is possible to wrap arbitrary code in a generic OGSA-DAI activity in order to execute it, removing the requirement that the code be rewritten (in the case of OGSA-DAI, in Java). This is important because data-analysis experts already have libraries of code which they rely upon and they may be unwilling to lose access to those libraries in order to adapt to a new system. Thus one possibility is to expand upon the work already performed and produce new generic wrappers in OGSA-DAI for a greater range of programming code, immediately giving access to a large volume of accumulated expertise.

Regarding HPC enactment: Whilst OGSA-DAI is a useful technology for integrating data from remote sources and piping it to other services, it is perhaps not so suitable for orchestrating the enactment of workflows within more tightly constrained, localised environments such as to be found within a given high-performance compute cluster. Since HPC figures heavily in the goals of VERCE but has not yet been adequately addressed by this early stage in the project, it is almost certainly advantageous to allow the VERCE gateway to delegate enactment of parts of a given workflow to be executed within a super-computing context on another enactment platform.

- One promising possibility is to support the translation of Dispel workflow elements into UNICORE job submission directives; it is believed that this will be broadly supported by existing HPC facilities.
- Another (possibly complementary) possibility is to use SAGA (the Simple API for Grid Applications) to translate Dispel workflow elements into a common job specification which can then be translated by any given HPC facility in accordance with its own preferences. This will require facilities to introduce further support for SAGA in general however.

Regarding secure enactment: Further work within the VERCE consortium is required in order to establish the preferred security model for the VERCE architecture in general. Any decisions made will impact the software stack used by VERCE on multiple levels, from requiring the gateway service to acquire initial user credentials from any linked portal services to requiring the correct

delegation of proxy credentials to a significant proportion of all resources delegated tasks. The initial work on integrating GridFTP into OGSA-DAI with X509 certificate exchange anticipates this, but further work will be necessary.

Within SDX: If SDX technology is to be further integrated into the rest of the VERCE architecture, then it is imperative to find a way to extract seismology data accessible via the VERCE gateway such that it can be analysed and stored locally using the SDX graphical user interface; in essence SDX becomes a primary example of an existing tool integrated into the greater VERCE architecture via the central hub of the gateway.

Of these, it is likely that only a couple can be substantially investigated and implemented before the next architecture specification cycle, with a few more given preliminary exploration / implementation subject to the distribution of manpower and synergy with the work of the JRA1 and SA3 groups in particular.

Another consideration for the further development of the VERCE architecture is the continuing relationship between VERCE and other EU projects, in particular EUDAT (for data management), EPOS (for services relating to solid earth science as well as data sources), EGI (for Grid-based services) and PRACE (for exploiting HPC-oriented resources for computation). Each of these projects is in the midst exploring technologies and forming formal recommendations, recommendations which VERCE is both in a position to influence and which VERCE is well-advised to adhere to where possible so as to ensure the interoperability of services in general.

7 An ICT Architect's Role and Vision

The role of a building's architect is to use their experience and judgement to balance a number of conflicting pressures in an attempt to satisfy the clients commissioning the building, the planning and regulation authorities, the users of the building and external observers. At the end, the best compromise should also be coherent, constructible within the available resources and timescale, fit with the current *and anticipated* built environment, and attract further work.

In the context of ICT, and in this case in VERCE, there are very similar desiderata. The architect, here JRA2, must have a vision of the goals and principles, which guide the negotiation of conflicting pressures. Just as a building's architect must repeatedly interact with clients to tease out exactly what they need, what they can afford and ultimately to gain approval for particular decisions, so too must JRA2 meet with the Earth scientists, through NA2 and JRA1, as well as with the project's and work packages' leaders, to pursue understanding, influence thinking and gain commitment to critical decisions.

Just as a building's architect must meet with planning officers, JRA2 must understand the extent to which external plans, from ESFRI, from INSPIRE, from EUDAT, from EPOS, from ENVRI, etc, should be taken into account as they will shape, and in some cases mandate, the future form of the environment in which the VERCE e-Infrastructure must operate. Coherence with their evolving provision is crucial for two reasons:

1. to facilitate the new science that a consistent e-Infrastructure facilitates, and
2. to make maintenance of both policy and technology economically feasible.

A use case prepared by Wouter Los in an ENVRI[66] meeting in Vienna on 20 April 2012 illustrates the first point. We had just seen a demonstration of OpenSearch[67] bringing together data on the volcanic ash cloud when Eyjafjallajökull erupted in 2010. Wouter, who leads the LIFEWATCH[68] and ENVRI projects, proposed that an investigation should be undertaken to show how marine species' populations

[66] http://envri.eu/
[67] http://www.opensearch.org
[68] http://www.lifewatch.eu/

were affected by the ash cloud. This will require EPOS data, data from EuroAgro, from meteorological records, from flights sampling ash levels and from LIFEWATCH's biodiversity observations. A coherent e-Infrastructure across such multiple Research Infrastructures[69] will make it easier to undertake multi-disciplinary investigations and encourage researchers to pursue them.

The second issue can be well illustrated by considering some of the policy and technical challenges that need to be permanently addressed to make integration across HPC centres, data archives and CPU- or data-intensive facilities feasible. The continuous negotiation to maintain trust and interworking as the facilities evolve cannot be achieved by a group the size of VERCE's research community. It needs more political muscle and both pan-European and International conversations. At the same time, the technology for all aspects of interworking will be complex and will require substantial on-going maintenance to remain operational. Therefore, this technology will only be sustainable if all those who need such facilities collaborate and use common solutions. These may be negotiated by organisations such as PRACE, EGI and ESFRI, and implemented by consortia of European projects or by the ICT industry.

The VERCE project has chosen to take a staged, agile approach to the design, implementation and operation of its e-Infrastructure. This is akin to developing a building site in succession of stages to enable initial parts of the client's business to move in sooner, with the understood caveat that there may be some disruption as subsequent stages are completed. In like manner, JRA2 are facilitating early partial versions of the VERCE e-Infrastructure to enable initial VERCE use cases to be undertaken. As with the physical building metaphor, this is intended to have two effects:

1. to accelerate the return on investment and facilitate earlier use by the clients, Earth scientists represented by NA2 and JRA1, and
2. to ground the discussion on the further development of the e-Infrastructure on more concrete and better shared understanding about the business of Earth science and the potential of the e-Infrastructure.

It is incumbent on architects to lead discussions on the conceptual vision to ensure that they and their clients share the same vision. The current vision is described here as a step in that process.

7.1 A Vision of the VERCE e-Infrastructure

The VERCE e-Infrastructure is intended to serve a large community of geophysicists and Earth scientists in Europe, to be an element of relevant international endeavours, such as GEOSS[70] and FDSN[71], and to be a formative influence on EPOS. EPOS itself will be an integral part of the group of ESFRI research infrastructures for Environmental Sciences, whose common e-Infrastructure will have been designed by ENVRI.

Soon after the end of VERCE, the user community will span a range of levels of experience in the Earth sciences, including the following groups.

1. Leading and very experienced seismologists, typified by those leading the VERCE project, which JRA2 interacts with at present.
2. New researchers, such as post-doctoral research assistants and PhD students, who have Earth science as their primary focus but are now users of the VERCE-pioneered, but now sustained virtual research environment (VRE). Note that new users will not invest time learning to use a VRE, unless they are confident it will be sustained – you do not move into a building that will collapse.
3. Researchers with any level of experience in other domains who wish to access VERCE/EPOS data, services and methods.

[69] http://ec.europa.eu/research/esfri
[70] http://www.earthobservations.org/geoss.shtml
[71] http://www.fdsn.org

4. Learners in universities and schools who may want access for a project, or in the former case to learn about Earth science methods, tools, data and standards.
5. Members of the public, including journalists and citizen scientists, who may just want information, or who would like to use data, methods and services.

This full range of users are equivalent to the future occupants of our building. The current VERCE project participants are almost universally in category 1. If VERCE is to have the intended impact and a user community that warrants further investment and therefore makes sustainability attainable, it must attract users in categories 2 to 4. Work packages NA3 and NA4 will alert the wider community, who will only adopt and support the VERCE products if they are easily accessible in a coherent and well-organised VRE. Coherence and good organisation cannot be skin deep. The consistency of the VRE will dependent on the underpinning e-Infrastructure having consistent elements and behaviour.

This broader, future community will need aids to finding the resources on offer in the VERCE/EPOS VRE. A VRE is a research environment that presents the resources: data, catalogues, metadata, tools and methods in an integrated and easy-to-use fashion for a distributed community of research users. These users will not be aware of the available features or the underlying code and data. In contrast, those formulating the current use cases must already be fully aware of the available data, application codes, tools and services, otherwise they would not have the knowledge to make the VERCE platform innovative.

To achieve the desired convenience for the future population of researchers, several features are needed that are not on the critical path for individual use cases; examples are listed below.

1. Directories and catalogues containing relevant metadata that, in conjunction with query and browsing tools, help users find the data, methods and services that they need.
2. Pre-packaged standard methods, with good affordance, documentation and diagnostics that provide an "intellectual ramp" to the adoption of new methods.
3. Data discovery and exploration tools that help users find *and understand* the data they need for their research.
4. Data and method ingest mechanisms that encourage others contribute to the facilities accessible through the VRE.
5. High-level tools for composing workflows, with effective, optimised enactment across heterogeneous resources, including those provided by other research infrastructures.
6. Unobtrusive usage management and accounting, with arrangements that are free for modest resource-bounded methods. It is important that users can have such things as a personal data cart, personal catalogue, private annotations, to let them experiment and learn before committing a lot of effort arranging to be fully fledged users of the VRE.
7. A collection of tools, information and mores that encourages collaboration and sharing, such as those pioneered in myExperiment[72].

The early VERCE developments cannot diverge to include all of these elements. However, it should shape its structures so that they can eventually be supported. A key technology for this is the description of research components: data, software, methods, tools, services and library elements. A further underpinning requirement is provenance mechanisms and tools operating at the appropriate level of detail. This will then allow later phases of VERCE to incorporate tools that help discovery, guide composition, assist with method application and permit high-level workflows to be optimised.

[72] http://www.myexperiment.org

References

[Abramson et al., 2008] Abramson, D., Enticott, C., and Altinas, I. (2008). Nimrod/K: towards massively parallel dynamic grid workflows. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 24:1–24:11, Piscataway, NJ, USA. IEEE Press.

[Altintas et al., 2006] Altintas, I., Barney, O., and Jaeger-Frank, E. (2006). Provenance collection support in the kepler scientific workflow system. In Moreau, L. and Foster, I., editors, *Provenance and Annotation of Data*, volume 4145 of *Lecture Notes in Computer Science*, pages 118–132. Springer Berlin / Heidelberg.

[Atkinson et al., 2011] Atkinson, M., Galea, M., Liew, C. S., and Martin, P. (2011). ADMIRE D2.9 — final report on the admire architecture, with an assessment and proposals for its development. http://admire-project.eu.

[Barclay et al., 2004] Barclay, T., Gray, J., and Chong, W. (2004). Terraserver bricks — a high availability cluster alternative. Technical Report MSR-TR-2004-107, Microsoft Research.

[Barga and Gannon, 2007] Barga, R. and Gannon, D. (2007). Scientific versus business workflows. *Workflows for e-Science*, pages 9–16.

[Barga et al., 2010] Barga, R., Gannon, D., Araujo, N., Jackson, J., Lu, W., and Ekanayake, J. (2010). Excel DataScope for Data Scientists. In *e-Science All Hands Meeting*.

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

[Bowers et al., 2007] Bowers, S., McPhillips, T., Wu, M., and Ludäscher, B. (2007). Project histories: Managing data provenance across collection-oriented scientific workflow runs. In Cohen-Boulakia, S. and Tannen, V., editors, *Data Integration in the Life Sciences*, volume 4544 of *Lecture Notes in Computer Science*, pages 122–138. Springer Berlin / Heidelberg.

[Callaghan et al., 2010] Callaghan, S., Deelman, E., Gunter, D., Juve, G., Maechling, P., Brooks, C., Vahi, K., Milner, K., Graves, R., Field, E., Okaya, D., and Jordan, T. (2010). Scaling up workflow-based applications. *Journal of Computer and System Sciences*, 76(6):428 – 446.

[Cheney et al., 2009] Cheney, J., Chiticariu, L., and Tan, W.-C. (2009). Provenance in databases: Why, how, and where. *Found. Trends databases*, 1(4):379–474.

[Clifford et al., 2008] Clifford, B., Foster, I., Hategan, M., Stef-Praun, T., Wilde, M., and Zhao, Y. (2008). Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience*, 20(5):565–575.

[Couvares et al., 2007] Couvares, P., Kosar, T., Roy, A., Weber, J., and Wenger, K. (2007). Workflow Management in Condor. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science*, pages 357–375. Springer London.

[Davidson, 2008] Davidson, Freire, J. (2008). Provenance and scientificworkflows:challenges and opportunities. *SIGMOD International conference on Management of data, ACM*, pages 1345–1350.

[Davidson et al., 2007] Davidson, S. B., Boulakia, S. C., Eyal, A., Ludäscher, B., McPhillips, T. M., Bowers, S., Anand, M. K., and Freire, J. (2007). Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50.

[Deelman et al., 2004] Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.-H., Vahi, K., and Livny, M. (2004). Pegasus: Mapping scientific workflows onto the grid. *Grid Computing*, pages 11–20.

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

[Deelman and Gil, 2006] Deelman, E. and Gil, Y. (2006). Managing large-scale scientific workflows in distributed environments: Experiences and challenges. In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, page 144.

[Deelman et al., 2005] Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., Laity, A., Jacob, J. C., and Katz, D. S. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237.

[Ellqvist, 2010] Ellqvist, T. (2010). *Supporting Scientific Collaboration Through Workflows and Provenance*. Linköping studies in science and technology: Thesis. Department of Computer and Information Science, Linköpings universitet.

[Fahringer et al., 2007] Fahringer, T., Prodan, R., Duan, R., Hofer, J., Nadeem, F., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.-L., Villazon, A., and Wieczorek, M. (2007). ASKALON: A Development and Grid Computing Environment for Scientific Workflows. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science*, pages 450–471. Springer London.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California.

[Foster, 2006] Foster, I. (2006). Globus toolkit version 4: Software for service-oriented systems. In *Lecture Notes on Computer Science: IFIP International Conference on Network and Parallel Computing*, volume 3779, pages 2–13. Springer-Verlag.

[Gannon et al., 2007] Gannon, D., Plale, B., Marru, S., Kandaswamy, G., Simmhan, Y., and Shirasuna, S. (2007). Dynamic, adaptive workflows for mesoscale meteorology. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science*, pages 126–142. Springer London.

[Gil et al., 2007a] Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., and Myers, J. (2007a). Examining the challenges of scientific workflows. *Computer*, 40(12):24–32.

[Gil et al., 2007b] Gil, Y., Ratnakar, V., Deelman, E., Mehta, G., and Kim, J. (2007b). Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In *AAAI*, pages 1767–1774.

[Goble and Roure, 2009] Goble, C. and Roure, D. D. (2009). The impact of workflow tools on data-centric research. In Hey, T., Tansley, S., and (Editors), K. T., editors, *The Fourth Paradigm: Data-Intensive Scientific Discovery*, pages 137–145. Microsoft.

[Görlach et al., 2011] Görlach, K., Sonntag, M., Karastoyanova, D., Leymann, F., and Reiter, M. (2011). Conventional workflow technology for scientific simulation. In Yang, X., Wang, L., and Jie, W., editors, *Guide to e-Science*, Computer Communications and Networks, pages 323–352. Springer London.

[Groen et al., 2011] Groen, D., Hoekstra, A., Borgdorff, J., and et al (2011). Deliverable D4.1 Review of Applications, Users, Softwares and e-Infrastructures. Deliverable report, MAPPER Project.

[Jackson et al., 2011] Jackson, M. J., Antonioletti, M., Dobrzelecki, B., and Hong, N. C. (2011). *Distributed Data Management with OGSA-DAI*, chapter 4, pages 63–85. Springer-Verlag.

[Jones, 2007] Jones, A. C. (2007). Workflow and biodiversity e-science. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science*, pages 80–90. Springer London.

[Kim et al., 2008] Kim, J., Deelman, E., Gil, Y., Mehta, G., and Ratnakar, V. (2008). Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 20(5):587–597.

[Liew et al., 2011] Liew, C. S., Atkinson, M. P., Ostrowski, R., Cole, M., van Hemert, J. I., and Han, L. (2011). Performance database: capturing data for optimizing distributed streaming workflows. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369:3268–3284.

[Lim et al., 2010] Lim, C., Lu, S., Chebotko, A., and Fotouhi, F. (2010). Prospective and retrospective provenance collection in scientific workflow environments. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 449 –456.

[Lin et al., 2007] Lin, A. W., Peltier, S. T., Grethe, J. S., and Ellisman, M. H. (2007). Case studies on the use of workflow technologies for scientific analysis: The biomedical informatics research network and the telescience project. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science*, pages 109–125. Springer London.

[Litzkow et al., 1988] Litzkow, M., Livny, M., and Mutka, M. (1988). Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*.

[Ludäscher et al., 2006] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., and Zhao, Y. (2006). Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065.

[Ludäscher et al., 2009] Ludäscher, B., Weske, M., McPhillips, T., and Bowers, S. (2009). Scientific workflows: Business as usual? In Dayal, U., Eder, J., Koehler, J., and Reijers, H., editors, *Business Process Management*, volume 5701 of *Lecture Notes in Computer Science*, pages 31–47. Springer Berlin / Heidelberg.

[Maechling et al., 2007] Maechling, P., Deelman, E., Zhao, L., Graves, R., Mehta, G., Gupta, N., Mehringer, J., Kesselman, C., Callaghan, S., Okaya, D., Francoeur, H., Gupta, V., Cui, Y., Vahi, K., Jordan, T., and Field, E. (2007). SCEC CyberShake Workflows—Automating Probabilistic Seismic Hazard Analysis Calculations. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science*, pages 143–163. Springer London.

[Martin et al., 2011] Martin, P., Atkinson, M., Parsons, M., Carter, A., and Francis, G. (2011). Edim1 progress report. Technical report, University of Edinburgh: School of Informatics & EPCC.

[Moreau et al., 2008] Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., and Paulson, P. (2008). The open provenance model.

[Mouat et al., 2011] Mouat, A., Janciak, I., Simo, B., Jarka, M., Laclavik, M., Buil-Aranda, C., and Baxter, R. (2011). ADMIRE D5.6 – tools development progress report and roadmap for further development. http://admire-project.eu.

[Oinn et al., 2006] Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M. R., Senger, M., Stevens, R., Wipat, A., and Wroe, C. (2006). Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100.

[Roure et al., 2009] Roure, D. D., Goble, C., and Stevens, R. (2009). The design and realisation of the ^{my}Experiment Virtual Research Environment for social sharing of workflows. *Future Generation Computer Systems*, 25(5):561 – 567.

[Simmhan et al., 2009] Simmhan, Y. L., Barga, R., van Ingen, C., Lazowska, E., and Szalay, A. (2009). Building the trident scientific workflow workbench for data management in the cloud. In *Third International Conference on Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09.*, pages 41 –50.

[Simmhan et al., 2005] Simmhan, Y. L., Plale, B., and Gannon, D. (2005). A survey of data provenance in e-science. *SIGMOD Rec.*, 34:31–36.

[Simmhan et al., 2008] Simmhan, Y. L., Plale, B., and Gannon, D. (2008). Karma2: Provenance management for data-driven workflows. *INTERNATIONAL JOURNAL OF WEB SERVICES RESEARCH*, 5(2):1–22.

[Singh et al., 2005] Singh, G., Kesselman, C., and Deelman, E. (2005). Optimizing grid-based workflow execution. *Journal of Grid Computing*, 3(3):201–219.

[Stevens et al., 2007] Stevens, R., Zhao, J., and Goble, C. (2007). Using provenance to manage knowledge of *in silica* experiments. *Briefings in Bioinformatics*, 8(3):183–194.

[Szalay et al., 2009] Szalay, A., Bell, G., Vandenberg, J., Wonders, A., Burns, R., Fay, D., Heasley, J., Hey, T., Nieto-SantiSteban, M., Thakar, A., van Ingen, C., and Wilton, R. (2009). Graywulf: Scalable clustered architecture for data intensive computing. In *42nd Hawaii International Conference on System Sciences*.

[Szalay, 2011] Szalay, A. S. (2011). Extreme data-intensive scientific computing. *Computing in Science and Engineering*, 13(6).

[Szalay et al., 2010] Szalay, A. S., Bell, G. C., Huang, H. H., Terzis, A., and White, A. (2010). Low-power amdahl-balanced blades for data intensive computing. *ACM SIGOPS Operating Systems Review*, 44:71–75.

[Tan, 2004] Tan, W.-C. (2004). Research problems in data provenance. *IEEE Data Engineering Bulletin*, 27(4):42–52.

[Taylor et al., 2007] Taylor, I., Shields, M., Wang, I., and Harrison, A. (2007). The triana workflow environment: Architecture and applications. *Workflows for e-Science*, pages 320–339.

[Thain et al., 2005] Thain, D., Tannenbaum, T., and Livny, M. (2005). Distributed computing in practice: the condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356.

[Yaikhom and Martin, 2011] Yaikhom, G. and Martin, P. (2011). DISPEL: Data-Intensive Systems Process Engineering Language Users' Manual (Version 1.0). Technical report, School of Informatics, University of Edinburgh.

[Zhao et al., 2008] Zhao, Y., Raicu, I., and Foster, I. (2008). Scientific workflow systems for 21st century, new bottle or new wine? *SERVICES '08: Proceedings of the 2008 IEEE Congress on Services - Part I*, pages 467–471.

Glossary

API Application Program Interface, an inter-software communication specification used for accessing functionality or services from programs.

cloud See cloud computing.

cloud computing A (business) model for enabling the delivery as a service of shared computing resources such as CPUs, networks, storage and applications to multiple users.

component One of the computational elements involved in a data-intensive or computational process, such as: application codes, scripts, workflows, services, catalogues, registries, data collections, data resources, functions, gateways, libraries, PEs, PE instances, format definitions and types.

DAGMan Directed Acyclic Graph Manager, Pegasus engine for executing workflows on available compute resources.

data Any digitally encoded information that can be stored, processed and transmitted by computers. Includes text files, database records, images, video sequences and recordings.

data archive The long-term storage of scientific data and methods.

data integration The process of combining data residing at different sources and providing the user with a unified view of these data. This process emerges in a variety of situations both commercial (when two similar companies need to merge their databases) and scientific (combining research results from different repositories) domains.

data mining The process of automatically extracting patterns from data using techniques such as classification, association rule mining and clustering.

data-analysis expert An expert in building or using knowledge discovery methods in a data-rich environment. In the context of VERCE, for example, they build libraries, such as ObsPy or workflows, such as cross-correlation or visualisers, such as SDX.

data-intensive An adjectival phrase that denotes that the item to which it is applied requires attention to the properties of data and to the ways in which data are handled.

data-intensive computing Computing that necessitates attention to any relevant property of data, including their volumes, distributed locations, and the heterogeneity of their formats and storage structures.

data-intensive engineer An expert in designing, providing, tuning, operating and improving the use of computational platforms for data-intensive tasks.

Dispel Data-Intensive Systems Process Engineering Language, a workflow composition language for data-intensive applications.

distributed computing The collective use of distributed resources, including data and applications, to solve a computational problem.

domain expert A person who is skilled in a particular field of research or decision making. In the context of VERCE, seismologists and later other Earth scientists.

e-Infrastructure The ICT element of a research infrastructure, i.e. a distributed collection of data, storage and compute resources, interconnected by digital communications and organised to serve a common research purpose. It includes the hardware, software, middleware, staff, operational procedures and policies needed to make it operate for that purpose, and requires maintenance to function in the evolving digital environment and to meet the changing needs of its user communities.

EDIM1 Edinburgh Data-Intensive Machine 1, University of Edinburgh experimental architecture for data-intensive computing.

enactment The execution of a workflow on a computational platform; this generally involves coordinated use of multiple and often heterogeneous communication, data and compute resources.

ESFRI European Strategy Forum on Research Infrastructures.

gateway A software subsystem, typically at the middleware level, that accepts requests for computational and data-handling tasks. It vets those requests to establish whether they are valid, e.g. are syntactically and semantically consistent, and are authorised. Requests that are not validated are rejected. Requests that are accepted are passed to other software systems, at the same or other locations, for execution. The gateway may partition and translate requests in order to combine heterogeneous services.

grid A system that is concerned with the integration, virtualisation, and management of services and resources in a distributed, heterogeneous environment that supports collections of users and resources (virtual organisations) across traditional administrative and organisational domains (real organisations).

GridFTP Grid File Transfer Protocol, an extension of the standard FTP for use with grid computing.

high-performance computing (HPC) Use of powerful processors, high-speed networks and parallel supercomputers for running computationally intensive applications.

IDE Also known as Interactive Development Environment, a software system designed for supporting software writing, often including a source code editor, a debugger and build automation tools.

INSPIRE Infrastructure for Spatial Information in Europe, an EU directive aimed at enabling the access, sharing and re-use of spatial data for governance and policy making purposes.

IRIS Incorporated Research Institutions for Seismology.

Kepler Open source scientific workflow management system.

KNIME Open source system for data mining.

Meandre Semantic-driven data-intensive workflow execution environment.

metadata Data that describes data. Metadata may include references to schemas, provenance, and information quality.

myExperiment Collaborative virtual research environment for sharing scientific workflows.

OGSA-DAI Open Grid Service Architecture Data Access and Integration, an open source product for distributed data access and management.

ontology In computer science, a formal explicit specification of a shared conceptualisation.

ORFEUS Observatories and Research Facilities for European Seismology.

Pegasus Workflow management service, mapping and executing workflows on available compute resources.

portal In the context of knowledge discovery, a tool designed for a particular group of domain experts that can be used via their browsers; it enables them to establish their identity and rights, and to pursue conveniently a set of research tasks for which the portal is designed.

processing element – PE A software component that encapsulates a particular functionality and can be used to construct a workflow.

registry A persistent store of definitions and descriptions of data or software components and their relationships accessed by tools and other elements of a distributed research environment. It is intended to facilitate discovery and use of the components.

repository A store holding software definitions, other shared code and data, that supports distributed concurrent access, update and version management.

Research Infrastructure The collection of equipment, resources, organisations, policies and community support that enables a particular discipline to conduct research. Normally, this refers to the advanced facilities that enable frontier research, such as the research infrastructures endorsed by ESFRI.

research object A research item which some researcher wishes to identify. It may be a collection of primary or derived data, code, a workflow, a service, an ontology, a set of metadata, etc. It may be a paper or a talk. Often it is a composition of such elements.

science gateway A consistently presented set of facilities designed to be a convenient working environment for researchers in a particular domain, in this case seismology. It should bring together access to all of the capabilities and resources such a researcher needs: including catalogues of available data and tools, established methods and arrangements for applying them with specified parameters to specified data.

Taverna Open source scientific workflow management system.

Trident Microsoft workflow management system.

VERCE architecture A high-level and coherent design for the VERCE e-Infrastructure; it evolves as the seismological goals and digital environment evolve and become better understood. It should guide the development of successive VERCE platforms.

VERCE e-Infrastructure An envisaged result of VERCE, as an integrated computational and data environment that presents a coherent virtual research environment in which to conduct seismology research and eventually research in other Earth sciences.

VERCE Platform The current realisation of the VERCE e-Infrastructure at any time in the VERCE project. Initially this is not fully integrated and may only constitute a partial implementation. Nevertheless, it is sufficient both to pursue research identified as priority seismology use cases and to develop and test the design of the VERCE e-Infrastructure. The VERCE platform is an approximation to the VERCE e-Infrastructure. These approximations should converge on the VERCE e-Infrastructure by the end of the VERCE project.

virtual research environment (VRE) A presentation of (ideally all of) the resources a researcher may need in a consistent and easily used form. These resources include catalogues, data, metadata, libraries, tools, workflows, programs, services, visualisation systems and research methods.

W3C World Wide Web Consortium, an international community of member organisations and the public that works to define and promote standards for web technologies.

web service A software system designed to support interoperable machine- or application-oriented interaction over a network.

workbench In this context a work environment for a computationally adept worker, such as a data-analysis expert, a data-intensive engineer or an application developer. It may be an IDE, an advanced editor or a command-line interpreter. It should provide all of the operations those workers need for creating, building, analysing, testing, debugging and making available the seismology and e-Infrastructure components. Many of the tools in a workbench will be familiar and widely used, a few will be specific to VERCE.

workflow A process of composed data-handling tasks, computational tasks and human interactions intended to implement a research method or established working practice.

wrapper A design pattern where a piece of code allows computational or data-handling components to work together that normally could not because of incompatible interfaces.

XML Extensible Markup Language.

ZigZag Language used by Meandre for describing the directed graphs that define workflows.