# D-JRA2.1.2: VERCE Architecture and Tools for Data-Intensive Applications

## 28/10/2013

---

| | |
|---|---|
| *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-JRA2.1.2.pdf |
| *Authors*[1]*:* | Malcolm Atkinson (UEDIN), Michelle Galea (UEDIN), Iraklis Klampanos (UEDIN) |
| *Location:* | http://www.verce.eu/Repository/Deliverables/RP3 |
| *Type of document:* | Deliverable |
| *Dissemination level:* | Public |
| *Status:* | Draft |
| *Due date of delivery:* | 30/10/2013 |
| *Reviewers:* | Siew Hoon Leong, Alberto Michelini |
| *Keywords:* | JRA2, data-intensive, compute-intensive, distributed-systems, architecture, e-Infrastructure, tools. |

---

| *History* | *Author* | *Date* | *Comments* |
|---|---|---|---|
| 1 | I. Klampanos (UEDIN) | 18/09/2013 | Initial draft. |
| 2 | I. Klampanos (UEDIN) | 30/09/2013 | First take on Task Forces, Registry and SCI-BUS sections. |
| 3 | I. Klampanos (UEDIN) | 01/10/2013 | Included significant additions by Malcolm Atkinson and minor corrections by Michelle Galea. |
| 4 | S.H. Leong (LRZ) | 24/10/2013 | Minor corrections. |
| 5 | I. Klampanos (UEDIN) | 28/10/2013 | Added up-to-date table of software packages. |

[1] Alphabetical order

## Copyright notice



VERCE, *Virtual Earthquake and seismology Research Community e-science environment in Europe*, is a project co-funded by the European Commission as an Integrated Infrastructure Initiative within the 7th Framework Programme. VERCE began in October 2011 and will run for 4 years.

# Contents



# List of Figures



# List of Tables

# Executive Summary

JRA2 is responsible for leading the development of a high-level architecture for enabling and enhancing research in seismology. The JRA2 activities are intended to prototype new distributed, high-level integration services and tools, intelligently and intuitively exploiting European computational resources and therefore enable new methods for seismology research. These prototypes may then be adopted and developed into production components by other workpackages.

JRA2's activities aim to

- identify critical components and services for the VERCE platform
- identify, appropriately adapt and integrate existing seismology data resources and analysis tools
- derive a toolset for the effective and efficient development and parametrisation of scientific data-intensive workflows
- aid the design and prototyping of the VERCE scientific gateway by consulting with the scientific community, advising on potential paths to integration, etc.

The activities and actions undertaken within JRA2 are guided by its role as the VERCE architect. Purely technical goals aside, 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. In each six-month cycle, the JRA2 work package will develop, prototype, communicate with partners and report on a set of increments to the architecture, designed to take VERCE closer to its goal.

## RP Objectives

Deliverables:

**D9.1.1** D-JRA2.1.2: Annual revision of the VERCE architecture: catalogue of prototyped or upgraded services and tools and code package available to SA2 and to research developers (former D-JRA2.2.1) catalogue of new or upgraded portlets in the scientific gateway tuned to the requirements of researchers (former D-JRA2.3.1).

Milestones:

**MS24** M-JRA2.2 Delivery of the VERCE architecture and catalogue versions with a demonstration of their capability.

### Work Progress

- Collaboration with SCI-BUS project, to lead to a MoU
- Exploratory discussions with ER-FLOW and SHIWA projects.
- VERCE Dispel Registry development and its integration with the Dispel Gateway

### Achievements

- Successful conclusion of the VERCE Task Forces

**In Progress and Next Steps**

During the next six months we expect to:

- have an evaluation-ready portal dealing with the forward-modelling use-case, based on SCI-BUS and gUSE technologies.
- help specify and put in place the necessary data-intensive-related VERCE architectures, backed by adequate registry descriptions.
- investigate the option to supplement the portal with a graphical (and possibly multi-workflow-language) workflow editor.

# 1 VERCE Task Forces

The introduction of the VERCE Task Forces (TFs) was a means to establish cross-WP collaboration in order to progress the driving use-cases of noise cross-correlation (data-intensive — DI) and forward modelling and simulation (high-performance computing — HPC). It was agreed that the work of the Task Forces concluded successfully in April 2013, having achieved the following:

- Better understanding and documentation of the requirements of scientists. Such requirements relate both to using the VERCE infrastructure (end-users) as well as to the technological particularities of the participating sites.
- Through a TF-JRA2 feedback loop, JRA2 was guided to informed decisions regarding additional software requirements and potential collaborations with other projects, such as with SCI-BUS.
- Demonstration of the stream-based Dispel workflow approach both from an architectural as well as from a user perspective.
- Development of a Python-based framework which allows scientists to contribute their scientific code for immediate inclusion in the Dispel seismology library and to run that code in different contexts.
- Collaborative development of the aforementioned processing elements.
- Integration of Dispel with VERCE HPC resources – this work will be superseded by current VERCE–SCI-BUS developments.
- Proof-of-concept presentations covering both use-cases, during the April 2013 review meeting.

As the technical requirements became clearer and the Task Forces groups larger, we decided that such organisation of the practical work within VERCE is neither sustainable nor adequate. We are currently in the process of restructuring VERCE so that we can proceed onto the next step of achieving a production-ready infrastructure for seismology. A restructuring was suggested by JRA2 and a specific draft of the proposition was circulated and discussed over Skype meeting in early July 2013 (see Appendix A). The decision at the steering committee meeting, held in Charles de Gaulle Airport on 30 July 2013 delegated this re-organisation to a Project Executive Board (PEB) that will meet frequently and shape technical decisions in accord with the project's strategic goals. JRA2 looks forward to the PEB overseeing the transition of work from proof-of-concept prototypes to production components and services.

# 2 VERCE Registry

During the last reporting period we specified and implemented the Dispel component of the VERCE Registry. The VERCE registry is designed with the aim of storing and presenting information regarding workflow and other research components. It is typically interrogated by graphical user interfaces (GUIs) to aid the researcher, while at the same time it provides computational component specifications and metadata to the execution engines for optimisation. Furthermore, such a registry should be aware of execution modes (test, development, etc.) and states (incomplete, failed, complete, etc.), being in communication with the provenance system and operational enactment logs.

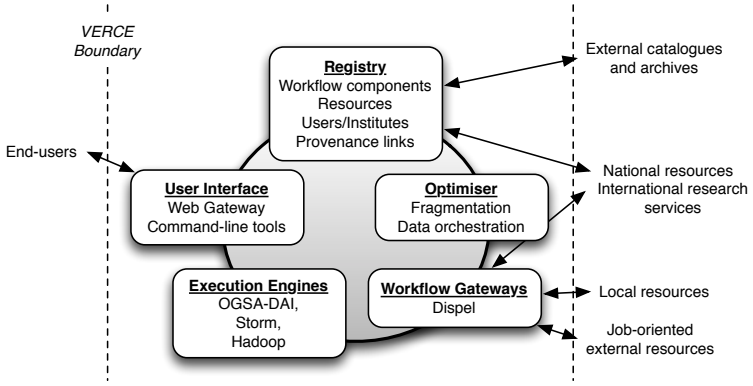



Figure 1 – Components interacting with the workflow registry.

Consistency across a large distributed system such as VERCE is achieved by the registry behaving as a reference source when code is deployed. Efficiency is achieved by fragmenting the workflow to run elements at the locations near their data, by mapping to specialised implementations, and by inserting transformations when necessary into the data streams coupling the distributed elements. All of which depend on consultation with the registry.

## 2.1 Architectural Overview

In a distributed scientific setting, the information registry stores information relevant to a number of entities, internal or external to the platform (Figure 1). As a facilitator for information and meta-data, the registry is interrogated by services, such as the enactment service and the user interface, in order to optimise workflow creation, execution effectiveness, parallelisation, data movement and presentation based on contextual and domain-specific information.

## 2.2 Dispel components

The current registry implementation is able to register and enquire about the following Dispel entities via a RESTful interface.

**Type** the core Dispel modelling unit. Types are most frequently used to define PEs and can be broadly divided into abstract and concrete. Abstract PE types only define available connection interfaces, while concrete types specify an internal topology of PEs and other components, realising the intended functionality.

**Literal** is a named Dispel literal. Such literals may be scientific constants, data-store references, etc.

**Implementation** an arbitrary block of programming code, usually implementing the functionality of a PE. Such implementations could range from scientific code written in Python to code performing job submission to HPCs, etc.

**Function (or constructor)** a Dispel structure used for defining arbitrary Dispel topologies, wrapping them into PE types. Functions are evaluated and expanded into enactable workflow segments during run-time.

**Connection** a typed data {in|out}let, part of the definition of PEs.

**Data Type** a language-specific entity used for specifying the data types of connections of PEs. Connections may be designated as being input or output and are typically named per PE definition. Each connection may carry two types: a *structural* type, or *s-type* and a *domain* type, or *d-type*. Typical s-types are 'Boolean', 'Stream', 'String', etc., whereas d-types depend on the application domain and can be of an arbitrary structure (*e.g.* semantic, graph-based descriptions, etc.).

### 2.3 Current Status and Future Work

The above entities, which include all of the seismology processing elements implemented in Python, are currently implemented and integrated with the Dispel gateway on a development/testing capacity. We are currently extending the VERCE registry model to include information about data and computing resources. We intend for the VERCE registry to make use of external registries wherever possible, not least as part of VERCE's collaboration with SCI-BUS (Section 3).

A prototype registry service will be available for testing and evaluation. This should lead to the adoption of a product service via SA2 within the next six months.

The complete specification of the Dispel component of the VERCE registry can be found in Appendix E.

While the details will certainly evolve, the use of registries to enable consistent operations and definitions is essential for the large, heterogeneous, multi-organisational research infrastructures needed for future Earth science. We therefore argue that it is imperative that they are adopted, used and refined in projects such as VERCE.

## 3 Collaboration with SCI-BUS

JRA2 has identified SCI-BUS as a potentially useful collaborator since the beginning of 2013, where initial discussions took place over emails and in Edinburgh. This process intensified after the last VERCE review meeting in April 2013, in Paris. Atkinson and Frank were asked to take this forward (a decision taken during the Steering Committee meeting which took place at Charles De Gaulle Airport on 30 July 2013), and was later confirmed at a meeting in Erice, Italy on 30 August 2013 (see Appendix B).

Working with SCI-BUS would be beneficial for VERCE due to two major technological overlaps: (1) job submission to HPC and other remote resources through its DCI-Bridge and related components; and (2) portal technology to help scientists perform such submissions and monitor their progress. In VERCE, we had already started working on independent solutions, for instance by integrating job submission functionality within Dispel workflows. However by collaborating with SCI-BUS, we help VERCE to become more relevant and sustainable.

After a meeting which took place at the University of Westminster, London, scheduled in order to develop mutual understanding and a common plan, JRA2, in collaboration with the SAs as well as with the project management, has now reached an agreement between the two projects. A roadmap covering the next six months (Appendix D) was agreed, which subsequently influenced the MoU between VERCE and SCI-BUS, currently ready for signature (Appendix C). On a practical level, the two projects are already working together in order to integrate the SCI-BUS technology with the VERCE architecture. For more details on the technical aspects of the collaboration, please refer to Deliverable D-SA3.3.

# 4 Software Summary Table

Table 1: Summary table of recommended, developed and used software. The column *Development* refers to software being developed internally or externally to VERCE, while the column *Status* provides an assessment of the software's robustness. The column *State* refers to the assessment state, ✔ indicating that a given software package has been assessed, while ◗ indicating that it is still under assessment. When a package has software prerequisites, these prerequisites are also taken to be part of the VERCE architecture implicitly.

| Name | Version | Description | Devel. | Status | State |
|---|---|---|---|---|---|
| **OGSA-DAI** | **4.2+** | Distributed data management software. http://sourceforge.net/projects/ogsa-dai/ | Internal | Stable | ✔ |
| **Dispel Gat'y** | **verce-2.2.2** | Workflow processing engine. http://sisprojs.ipgp.fr/repos/verce/All/JRA/JRA2/gateway | Internal | Stable | ✔ |
| **Verce Reg'y** | **0.2** | VERCE Registry of processing elements and related entities. http://sisprojs.ipgp.fr/repos/verce/All/JRA/JRA2/VerceRegistry | Internal | Stable | ✔ |
| **gUSE** | **3.5.8** | Science gateway framework that allows users to make use of grid and cloud infrastructures. gUSE is used as part of the collaboration between VERCE and SCI-BUS. http://guse.hu | External | Stable | ✔ |
| **jSAGA** | **0.9.14+** | SAGA job-submission client library. The use of jSAGA has been largely replaced by submission to computing resources through gUSE; however, for the time being is being considered as "under evaluation" until our work with gUSE has been completed. http://grid.in2p3.fr/jsaga/ | External | Stable | ◗ |
| **Java** | **6+** | General-purpose programming language by Oracle. http://www.oracle.com/us/technologies/java/overview/index.html | External | Stable | ✔ |
| **Python** | **2.7** | General-purpose scripting language. http://www.python.org | External | Stable | ✔ |
| **Tomcat** | **5.5+** | Java servlet container. http://tomcat.apache.org | External | Stable | ✔ |
| **ObsPy** | **0.8** | Toolkit for seismological computations. | External | Stable | ✔ |

# Appendices

## A Notes on VERCE Technical Teams - JRA2 Proposition for VERCE Restructuring

# Notes on VERCE Technical Teams

Link to working document:
https://docs.google.com/document/d/1hSvAiA5QpSnpzYG4eNjpnizd9cd_faY86EhwzpEsnxU/edit?usp=sharing

## Revision History

| *Date* | *Author* | *Description* |
|---|---|---|
| 1/7/2013 | Alessandro and Iraklis | Initial draft |
| 4/7/2013 | Multiple VERCE members during and after a joint TF skype call | A number of refinements |
| 10/7/2013 | Malcolm, Luca and Alessandro | Further refinements based on discussions at KNMI |
| 11/7/2013 | Iraklis | Minor changes to the formatting of the document |
| | | |
| | | |
| | | |

## Preamble

The definition of the suggested teams below are merely notes and work in progress and their composition will depend on whether and how people would like to participate. The deliverables listed below are high-level descriptions of interfacing components which would be required for integrating the various parts.

### [1] Cross-correlation (seismo group 1)

Definition of workflows and implementation, definition and development of PEs in Python, to support the scientific requirements.
We would expect that this work will lead to an augmented VERCE-specific Dispel library.

#### Deliverables

To be decided and set by the seismology teams or research groups.

#### Members (prelim. )

Mario (IPGP), Xavier (?, Grenoble), Celine (LMU), Licia (INGV), Gaia (INGV), Lucia (?, INGV), Peter (INGV), , Peter (INGV), Andre (SCAI), Cerlane (LRZ);
(led by scientists and supported by ITs)

#### Links to VERCE work packages and justification

### [2] Forward/Inverse modelling (seismo group 2)

Definition of workflows and implementation, definition and development of PEs in Python, to support the scientific requirements.

We would expect that this work will lead to an augmented VERCE-specific Dispel library.

### Deliverables

To be decided and set by the seismology teams or research groups.

### Members (prelim.)

Lion (LMU), Federica (INGV), Emanuele (INGV), Peter (?, INGV), Cerlane (LRZ);

### Links to VERCE work packages and justification

## [3] Data layer (irods, data replication, transfers, etc.)

Setting up the VERCE iRODS testbed.
Cover all VERCE sites supporting relevant metadata and local data management and access policies.
Define rules for data and metadata transfers between sites as well as documentation including extensibility and other features.

### Deliverables

1. Installation of federated IRODS on all relevant VERCE sites
   (IPGP, ISTerre, SCAI, CINECA, INGV, UEDIN, KNMI)
2. Testing for cataloguing and data-transfer
   a. Transfers between IRODS and non-IRODS (generic GridFTP) sites
3. Interface with VERCE-wide ICAT? or similar. Basic operations should include data inquiry by metadata (time, station names or location, event?, other; return relevant VERCE IRODS repositories and filenames.
4. Exploration of the use of PIDs for data objects of importance.

### Members

**Visakh** (SA1, IPGP), David (SA1, IPGP), Xavier (ISTerre), Michele (CINECA), Michael (SA1, SCAI), Peter (INGV)

| *Del#* | *Due date* | *Other teams* | *Notes* |
|---|---|---|---|
| 1 | 14/08/2013 | | |
| 2 | | | |
| 3 | | | |

### Links to VERCE work packages and justification?

IRODS testbed notes:
https://docs.google.com/document/d/157HzAnMYwxsLcKqFtNe-N-aKcdvTDCZcAa1iFViFo2o/edit?pli=1

## [4] Dispel gateway, execution and optimisation

Dispel gateway maintenance/updates and bug fixing.
Exploration with Storm. On-site optimisation strategies. Further development and maintenance of the integration framework for python PEs as well as high-level optimisation of Dispel workflow execution.

### Deliverables

1. Plan for regular releases
2. Interfacing with additional subsystems (e.g. DCI-Bridge)
3. Dispel-to-Storm interpreter
4. Unified framework for importing and enactment of 3rd-party PEs
5. Model for [rule|weight]-based partitioning - useful features and actions
6. Workflow partitioning and expansion component

### Members

**Amy**, Luca, Alessandro, Iraklis, Mario [additional members from SA1 and from the seismology community]

| *Del#* | *Due date* | *Other teams* | *Notes* |
|---|---|---|---|
| 1 | 2/8/2013 | | |
| 2 | 1/11/2013 | | |
| 3 | 30/8/2013 | | Parallel to 4 |
| 4 | 30/8/2013 | | |
| 5 | 20/12/2013 | 7, 8 | Will be updated later |
| 6 | 20/12/2013 | 7, 8 | Will be updated later |

### Links to VERCE work packages and justification

## [5] Dispel library definition (dispel.lang)

Definition of a minimal set of basic and generic Dispel PEs, which will support seismological and other VERCE-related operations. I.e. a root Dispel library with suggestions for maintaining, release cycles, etc.

### Deliverables

1. Initial release of dispel.lang
2. Regular subsequent releases of library specifications and components, including additions requested by the use-case groups.

### Members

**Malcolm**, Michelle, Paul?, Iraklis, Amy, Someone from the seismology community (teams 1, 2)

| *Del#* | *Due date* | *Other teams* | *Notes* |
|---|---|---|---|
| 1 | 4/10/2013 | 1, 2, 4 | |
| 2 | | | |
| 3 | | | |

### Links to VERCE work packages and justification

## [6] Security

VERCE strategies for secure data transfers and workflow execution throughout the VERCE sites. Priority should be given to the Forward/inverse modelling use-case (due to the latest project review).

### Deliverables

1. Basic report on security infrastructures used at VERCE sites - current status quo
2. General guidelines for VERCE components

### Members

Cerlane, Andre, David, Peter, Michele, Iraklis, Malcolm (to provoke management discussion on policy)

| *Del#* | *Due date* | *Other teams* | *Notes* |
|---|---|---|---|
| 1 | | | |
| 2 | | | |

### Links to VERCE work packages and justification

## [7] Registry

Should provide a number of “registries”, for Dispel, resources and data, possibly followed by more. These will be accessible through a RESTful interface. Evaluation and reuse of specifications such as GLUE 2, or appropriate derivatives.

### Deliverables

1. Dispel components registry (REST)
2. Resources and data information registry (REST)
3. Model for and integration with external catalogues

### Members

**Iraklis**, Amy, Mario (to be aware of what’s happening here)

| *Del#* | *Due date* | *Other teams* | *Notes* |
|---|---|---|---|
| 1 | 2/8/2013 | 4, 5 | Basic RESTful interface and documentation |
| 2 | 4/10/2013 | 4, 5, 6 | Basic RESTful interface and documentation |
| 3 | 29/11/2013 | | |

### Links to VERCE work packages and justification

Mainly JRA2, SA1, SA2

## [8] Provenance

Should provide lineage information (data stream transformations dependencies and processes Involved) and should be accessible at runtime. Should extract/process/store datasets metadata which might be defined by the process itself in terms of annotations.

### Deliverables

1. Model and specification of provenance levels
2. Integration in the Gateway/Enactment Engine and Query REST API (Depending on query patterns (Seismology Use-Cases)
3. Interface and/or some integration with the Registry -> optimisation

### Members

**Alessandro**, Amy, Iraklis, Mario, Federica and one more for the DI use case

| *Del#* | *Due date* | *Other teams* | *Notes* |
|---|---|---|---|
| 1 | 12/13 | Dispel Gateway, Enactment, Optimisation, Data Management | Metadata refinement (matching on PROV concepts) and distributed vs centralised storage strategies and access evaluation. |
| 2 | cyclic from 9 / 13 | Dispel Gateway, Enactment, Forward Modeling, Science Gateway | Use Case driven API releases, based on query patterns defined by iterative users' feedback (possibly supported by visual interfaces and rapid prototyping). |
| 3 | 12/13 | Registry / Optimisation | Evaluation extraction/access strategies and relevant metadata |

### Links to VERCE work packages and justification

## [9] Science gateway

A portal and associated technologies which will allow the users to execute workflows related to the scientific use-cases defined by the corresponding teams. Should account for extensibility especially towards other projects (SCI-BUS or similar). Should also take into account technologies which would allow the graphical specification/editing of workflows (e.g. jsPlumb?)

### Deliverables

1. Incremental specification and design - requirements from other components/groups
2. Basic interface for forward-inverse modelling use-case

### Members

Claudia, Alessandro, Michelle, Xiao, Emanuele + one from DI, (Temporarily Sandra)

| *Del#* | *Due date* | *Other teams* | *Notes* |
|---|---|---|---|
| 1 | cyclic from 7 / 13 | Provenance, Forward Modeling, Cross Correlation, Security, Data Management? Dispel Gateway | bi-weekly to monthly iterations with users and other teams on achieved results based on progress |
| 2 | 10/13 | Provenance, | Based on first set of iterations with the other teams. |

| | | | |
|---|---|---|---|
| | | Forward Modeling | |
| 3 | | | |

## Links to VERCE work packages and justification

Summer School on Workflows and Gateways

# B Notes on VERCE Meeting 30 August 2013, Erice

# Notes on VERCE meeting 30 August 2013, Erice

## Present

Malcolm Atkinson, Torild van Eck, Amy Krause, Alberto Michelini, Alessandro Spinuso, and Jean-Pierre Vilotte

**PEB and SC**

The discussion at CDG was resumed, and the institution of the PEB with a joint chair of a seismologist and a computer scientist was confirmed. It should start ASAP in September. Membership and chairmanship needs confirming. It should contain and be led by those technically active in VERCE. The engagement of seismologists from each seismology site needs arranging. It will need to tele-meet regularly, at least once/two weeks, but should define its own method of working.

**J-PV to oversee finalisation of initial membership & choice of chair.**

The roles of SC & PEB need to be clearly separated. The SC to tele-meet monthly and have a prioritisation and strategic role, not a technical one.

**J-PV to circulate an announcement of PEB & SC new model and current priorities to all of VERCE**

**HPC Beta-test Planning**

This has highest priority. It should proceed along the lines already agreed and be available via the VERCE Science Gateway by February 2014 in a state suitable for training and evaluation. There was discussion over the inputs and where they should come from. It was agreed that

1) One-on-one or very selected group dialogue would be used to resolve questions,
2) SA1 would be asked to provide the computational, storage and security platform needed,
3) Immediate discussions would take place with SCI-BUS on its contribution to the scientific gateway, AAAI management and job handling,
4) That the mechanism for data staging developed by EUDAT would be considered later, and
5) That mechanisms for selecting Earthquake data in QuakeML form, and reference seismometers, would be in use the current arrangements with the possibilities of using other web services and optimisations being considered later.

The PEB will initially focus on managing this work and report unresolved issues to the SC.

**Data-Intensive Planning**

Arrangements for running the processing elements that should still be written to be compliant with distributed streaming execution would be explained so that seismologists can compose and run streaming workflows entirely by scripting in Python.

**AS to send out a tutorial note.**
**SA1 to supply platform for this**

A meeting to clarify and plan the development of a data-intensive seismology framework with effective scalability, data management and multi-DCI use is needed. It should specify goals (details of the scope and characteristics and test cases) and develop a plan to deliver a version worthy of beta-test by May 2014.

**J-PV to convene this meeting in late October or very early November**

**Deliverables**

The deliverables due in September should be concise and minimal (normally three pages).

**J-PV to circulate note to VERCE WP leaders ASAP**

**Working with SCI-BUS**

At the CDG meeting MPA & Anton Frank undertook to draw up an MoU with SCI-BUS for ratification by the two project leaders. We have arranged a meeting on technical collaboration with SCI-BUS in London on 12 & 13 September.

**AS & AK to arrange details and attend that meeting**

**MPA to follow up with Tamas Kiss on MoU draft & to bring in AF to the MoU process**

# C Memorandum of Understanding Between SCI-BUS and VERCE (Draft as of 17 September 2013)


# MEMORANDUM OF UNDERSTANDING

# BETWEEN

# SCI-BUS

# AND

# ASSOCIATED PARTNERS

The purpose of this Memorandum of Understanding (MoU) is to define a framework of collaboration between SCI-BUS, an European Project supported by the FP7 Capacities Programme under contract n°RI-283481, with address in MTA SZTAKI, 1111 Budapest, Kende u. 13-17, Hungary, and Institut national des sciences de l'univers (National Institute for Earth Sciences and Astronomy), hereafter referred to as "the Party", representing VERCE, Virtual Earthquake and seismology Research Community in Europe e-science Environment, European Project. Hereafter SCI-BUS and the Party referred to as "the Parties".

The Parties recognise, by this MoU, the opening of a wider and longer-term cooperation in activities, which will bring visible benefits.

The Parties are desirous of entering into this Memorandum of Understanding to declare their respective intentions and to establish a basis of cooperation and collaboration between the Parties upon the terms as contained herein.

## AREAS OF COOPERATION

The main goals of the activities developed in the context of this MoU are:

1. Explore the development of a scientific gateway for a community represented by the Party.

2. Promote and disseminate the scientific gateway and contribute to disseminate the SCI-BUS and VERCE joint success stories.

Commitments of SCI-BUS:

- Give training on SCI-BUS technology, and contribute to at least one joint training event inside a Party event/summer school.
- Coordinate the collaboration between the Parties.
- Give 2 person months labour support for developing the science gateway of the associated partner.
- Inform the Party about all the material, guides, training courses that may be interesting for this collaboration.
- Disseminate the collaboration and promote the science gateway developed by the Party as success story or testimonial.
- Promote the Party as associated partner on the SCI-BUS web page and at events organized by SCI-BUS.
- Enable the associated partner participate at the SCI-BUS project meetings.
- Register for the Party User Forum.

Commitments of the Party VERCE:

- To explore the development of a science gateway based on SCI-BUS technology (WS-PGRADE/gUSE).
- Define a roadmap of the planned work in consultation with SCI-BUS staff, and with six-monthly reviews and optional revisions agreed between the Parties.
- Disseminate and promote the gateway for the target community as well as internationally.
- The Party intends to set up the science gateway, if adopted, as a production service for its user community.
- The user community won't be related to any area under ethical compromise or biosecurity such us military research or bioethical studies.
- Participate at least in one of the SCI-BUS organized annual gateway workshops.
- Register for the SCI-BUS User Forum.

## FINANCIAL ARRANGEMENTS

This Memorandum of Understanding will not give rise to any financial obligation between the Parties. VERCE and SCI-BUS will bear their own costs and expenses in relation to this Memorandum of Understanding.

## COMMUNICATION

The Parties shall keep each other informed on all their respective activities and on their progress and shall consult regularly in areas offering potential for cooperation. Each of the parties shall designate a "point of contact" that shall be responsible for monitoring the implementation of this MoU and for taking measures to assist in the further development.

## EFFECT AND DURATION

This Memorandum of Understanding will come into effect on the date of signing and will remain in effect for a period of 2 years. This Memorandum of Understanding may be extended for a further period as may be agreed in writing by the Parties.

Signed in duplicate at ……*(place)*……., on ….*(date)*.....

Signed by, for and on behalf of
SCI-BUS

Dr. Peter Kacsuk

Project coordinator

Signed by, for and on behalf of
VERCE

Professor Jean-Pierre Vilotte

Project coordinator

# D VERCE, SCI-BUS Six-Month Roadmap

# Roadmap for SCI-BUS & VERCE Collaboration October 2013 to March 2014

Malcolm Atkinson, Peter Borsody, Zoltán Farkas, Anton Frank, Peter Kacsuk, Tamas Kiss, Iraklis Klampanos, Amy Krause, Siew Hoon Leong (Cerlane), Clàudia Ramos Garcia, Alessandro Spinuso, and Gabor Terstyanszky
17 September 2013

## Introduction

The following roadmap was developed at a meeting in University of Westminster on 12 & 13 September 2013 between representatives of SCI-BUS and VERCE. This is an adjunct to a proposed MoU between those projects and a practical plan designed to meet VERCE's immediate requirements. The numbered items are approximately in temporal order, but they also overlap. A names in blue against an item means, someone from VERCE who will be the lead contact.

1, Deployment of gUSE for test purposes at Edinburgh – completed by end of September

- Latest version from SourceForge — Iraklis Klampanos
- Possible Linux distribution to be explored — Iraklis Klampanos
- Globus and Unicore submitters are required, investigate the status of the Unicore submitter, just for simple submission, may not be required immediately. Only in 6 month's time. Keep VERCE in the Unicore discussion loop. (SCI-BUS lead) — Cerlane Leong
- gLite only later on low priority (not required at present)
- connect to targeted HPC resources — Cerlane Leong
- run test workflows (supplied by SCI-BUS to test connections, etc.) — Alessandro Spinuso
- support from Dario Ferrer (UoW)

2, Development of HPC workflows for job submission – completed by mid October

- only the HPC part to be tested (using real forward-modelling code) — Amy Krause
- use command-line first — Amy Krause
- Support from Peter Borsody (UoW)

3, Investigation of security issues – completed by mid October

- Which certificate formats, which MyProxy servers, etc. — Amy Krause

4, Investigation of Dispel workflows – completed by end of October

- Use a specific script embedded in a WS-PGRADE workflow (option 3) — Amy Krause
- Support from Peter Borsody (UoW)
- Investigation of SHIWA (option 2) or DCI bridge (option 1) based solution is later (after 6 months) (not needed until later, decide lead contact then) — tbd

5, Integration of the VERCE custom portlet with the workflows using ASM – completed by end of December with first HPC workflows, end of January, with Dispel workflow — Alessandro Spinuso

- Can be started once first HPC workflows are available — Alessandro Spinuso
- Tests even on dummy workflows — Alessandro Spinuso
- Support from Peter Borsody (UoW)

6, Submission of VERCE work to local clusters – completed by end of December

- Determination of job-submission system to be used at IPGP, INGV, Universities of Liverpool and Edinburgh, and other VERCE sites that will work on data-intensive seismology Geneviéve Moguilny (tbc)
- Determination of AAI requirements at each site Geneviéve Moguilny (tbc)
- Test of sample job submissions using test workflows Geneviéve Moguilny (tbc)

7, Production portal installation at KNMI – completed by end of January

- Start in January Alessandro Spinuso
- Mirroring the Edinburgh installation Alessandro Spinuso
- Edinburgh continues as development environment, with KNMI running production

8, Demonstrator for VERCE forward-modelling HPC Milestone – completed by end of February Iraklis Klampanos

## Requirements for next 6 months – completed by end March 2014

- Independence of gUSE/WS-PGRADE from Liferay version – install WS-PGRADE with any (up to date) Liferay, if not possible, then with a specified Liferay version by VERCE, needed for the Liferay document-store utilisation, joint discussion required on this (led by SCI-BUS), Alessandro Spinuso
- VERCE investigates current gUSE security for VERCE purposes, and feeds it back to SCI-BUS Cerlane Leong
- iRODS integration with data bridge: VERCE provides access to iRODS resources for testing whether the data bridge can handle iRODS resources, investigation done by SZTAKI. Visakh Muraleedharan
- VERCE needs data-bridge workflow integration (connecting to iRODS) within 6 months Visakh Muraleedharan
- SCI-BUS to investigate how to access information via the ASM API in the gUSE internal workflow repository and DCI configuration information. Iraklis Klampanos

# E VERCE Dispel Registry Design

# Supporting Collaborative Scientific Workflow Development: The Dispel Information Registry

Iraklis A. Klampanos, Paul Martin and Malcolm P. Atkinson
School of Informatics, University of Edinburgh, UK
Email: iraklis.klampanos@ed.ac.uk

***Abstract*—This paper reports experience designing technology to support large-scale distributed computations that arise in scientific research and in many other modern contexts. The challenge is to support data-intensive work across multiple autonomous sites, while experimentation and collaborative development are simultaneously encouraged across the same computational infrastructure. We specify appropriate registry modules and their interactions with other core components, designed to achieve the aforementioned goals within the context of a fine-grained workflow e-Science platform. We demonstrate our method's suitability through an ambient-noise cross-correlation example drawn from the field of Seismology.**

## I. Introduction

There is a widespread and increasing requirement for data-intensive computation in science, engineering, medicine, government and many other contexts [7]. The term "data-intensive" is used to characterise computation that either requires or generates large volumes of data, or has complex data access patterns due to algorithmic or infrastructural reasons. In most of these examples, the research proceeded by bringing all of the required data into one administrative context and then exploration was conducted by teams that included the domain scientists, data-analysis specialists and computer scientists. Similar arrangements are typical in many other contexts, such as those reported in [13], [14], [5].

We argue that there are many cases where this collection of data and software under one administrative regime is not the optimum solution. First, the owners of the data are spread across organisations and they may wish to restrict access to certain data sets while also collaborating on methods with colleagues elsewhere. Second, the data or software is rapidly changing at some sites, so that co-location reduces timeliness. Last, the cost or time involved in extracting, transforming, moving and organising the data into a single location can be too great.

We posit that scientific workflow platforms provide the necessary functionality for automatically orchestrating the actions pertaining to the above data-intensive activities, while providing researchers with an abstraction that makes the inherent growth of, especially distributed, software libraries and data more manageable. Such platforms should enable researchers to browse, extend and create new computational components, organise remote data resources and stores in a way relevant to the work at hand and allow them to compose or edit workflows deployable to the distributed infrastructure. Given sufficient support mechanisms, the platform should be able to facilitate different degrees of supervision during the execution of a workflow, as well as test and production enactment.

In this paper we study the workflow information registry as a logical component of such a platform. We focus on e-Science, due to its inherent resource and administration inhomogeneity, as well as due to its, often, collaborative nature. The registry holds and is able to present information regarding workflow and other research components. It is typically interrogated by UIs to aid the researcher, while at the same time it provides computational component specifications and metadata to the execution engine for optimisation. Furthermore, such registry should be aware of execution modes (test, development, etc.) and states (incomplete, failed, complete, etc.), being in communication with the provenance system and logs. It follows that consistency is achieved by the registry behaving as a reference source when code is deployed. Efficiency is achieved by fragmenting the workflow to run elements at the locations near their data, by mapping to specialised implementations, and by inserting transformations when necessary into the data streams coupling the distributed elements, all of which, to an extent, via consultation with the registry. An earlier implementation of this model can be found in [1] and the use of the registry for optimisation is reported in [10].

The next section introduces related work and scientific research procedures. Section III presents requirements that would need to be met by the information registry for use within a scientific setting. Section IV introduces the design and logical structure of the information registry component as well as its interactions with other related components of the e-Science platform. Section V provides an example use-case drawn from seismology and how the registry can be used to aid optimisation and orchestration. Last, in Section VI we provide concluding remarks and pointers for future work.

## II. Rationale and Related Work: Distributed, Data-Intensive Workflows

Experiments conducted by researchers can no longer be considered as isolated operations performed at one locale within a modest time frame. Instead, experiments may require the orchestration of resources over a prolonged period, with a gradual accumulation of results data throughout. In addition, researchers work in collaborative environments, where they are increasingly expected to exchange procedures (reflected by software), results and analyses, often implicitly or explicitly requiring the movement of large volumes of data.

This scenario calls for the flexible deployment of persistent computational services across many different contexts, and the establishment of high-throughput data channels between them. These services must command substantial local resources and be able to re-configure themselves on demand. Where

necessary, services have to be wound down to release scarce resources and re-deployed closer to active data sources. This is not a scenario which permits manual coordination on large scales; certainly not on the part of a user constituency which is ostensibly concerned with science over software engineering and resource management.

### A. Workflows

In the context of persistent services, a workflow can be seen as a, possibly indefinite, configuration of data-intensive machinery towards a specific purpose. Connections are established between service instances which re-gear themselves towards providing the processing elements prescribed by the workflow. Data flows into processing elements via such channels and flows out to elements further down the workflow. Flow control is handled by the enactment services themselves, in accordance with agreed protocols.

*a) Architectural overview:* There must exist a common interface and behavioural specification for data-intensive machinery (persistent services) which can be deployed on distributed resources, forming a distributed enactment platform. There must also exist a choreography service to broker the choreography of persistent services; control over the workflow must be distributed between the choreography service and the enactment services in such a manner as to allow easy monitoring of progress while conferring onto the enactment services the autonomy they need to conduct their part of the workflow in the most independent and flexible way possible. It is also clear that there must be a standard way to specify workflows to be imposed on the enactment platform. This suggests the need for a language in which to describe the processing elements used in a workflow and the connections between them, specifying the data-types involved and any desired characteristics of the overall workflow, which might affect how it should be deployed and choreographed onto available resources.

*b) The Dispel Workflow Specification Language:* The role of a workflow language is to provide a standard *lingua franca* for describing the choreography of logical components collectively composing a workflow.

Since we view the enactment system as a collection of concurrent persistent computational services acting in concert, our view of workflows is data-oriented rather than control-oriented, as defined in [3]. The deployment of processing elements onto enactment resources is taken as given, the task being delegated to the choreography service. Data elements are streamed as and when ready from one processing element to another, allowing components of a given workflow to operate in parallel as long as all inputs can be provided. The buffering and replication of data is delegated to the individual enactment services, allowing connections to be established directly between co-dependent elements without need for buffers and filters to be defined explicitly.

Dispel is a workflow language introduced by the ADMIRE project[1]. Dispel is imperative, rather than declarative (similar to Pig Latin [12]). A Dispel script essentially describes a high-level process to construct a workflow rather than a specific workflow instance. This allows the use of imperative constructs, such as iteration, selection and sub-procedures to be employed in order to describe arbitrarily complex, parametrisable workflows. Dispel's imperative nature also allows the construction of composite processing elements from the composition of existing workflow elements without resort to external configuration or registration.

Upon interpretation (generally by submission to a gateway) a workflow is produced. A Dispel workflow is an abstract network of processing elements between which data can be streamed. More specifically, a *processing element (PE)* is an operator with a number of connection interfaces through which data is either consumed or output, while a *connection* streams data from one output interface to at least one input interface.

Dispel relies on the concept that a standard library of abstract processing elements can be mapped into a variety of different execution contexts (*e.g.* OGSA-DAI[4]), provided that there exists a single logical specification of the behaviour of each element for which there may exist many compliant implementations. This separation of workflow language from specific execution contexts distinguishes Dispel from many similar languages such as SCUFL[8], MoML[9] or ZigZag[11].

A Dispel script is typically used to either describe a logical workflow for submission, or to define and make available new workflow components. It can, therefore, be seen as providing a means of interaction between users and the platform, through the platform's registry, the required properties and design of which are presented in this paper. Its purpose is therefore diverse, as it can be used for deployment/enactment of a workflow or for, implicit or explicit, registration of workflow entities, with both actions requiring appropriate entity resolution at the registry level.

### B. Scientific Experimental Procedure

Replicability is essential for verifying and validating experiments, as well as to ensure trust in shared processes. Given the potentially disparate nature of resources conscripted into the enactment platform, it may be that processes executable on different resources may require different underlying implementations. Even if this is the case however, there should be a standard logical description of any given processing element which accurately describes its behaviour in all valid contexts. This entails an external registry service dedicated to the maintenance and publication of those descriptions which can then be referred to in workflow specifications dispatched to the coordination service. A consequence of this is that if the logical operation of a component *does* change over time, then the conduct of experiments using that component will change; it must be possible therefore to both conduct a previous experiment under the new specifications (benefiting from any improvements made) and conduct the experiment as it was before (so as to verify prior results).

## III. Registry Requirements

In order to address the requirements of global consistency, local agility and simplicity, posed by the distributed and large-scale nature of modern science, we build our architecture

[1]Advanced Data Mining and Integration Research for Europe: http://www.admire-project.eu.

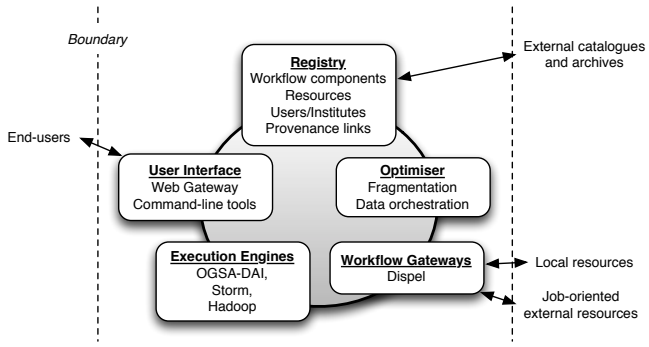



Fig. 1: Components interacting with the workflow registry.

around three crucial elements: a data-intensive workflow language, an enactment/execution layer and a registry of computational components. Together these elements form the core of our distributed architecture, with their combined semantics dictating patterns of control and data passing, orchestration as well as user-interaction. In this section we discuss the main requirements of the information registry, mainly in relation to the other two components, as a service that ensures consistency across the distributed infrastructure.

### A. Distribution

The environment we study is inherently distributed, both in terms of computing as well as of human resources. The modern scientist will typically make use of a number of disparate computing resources, such as grids, high-performance (HPC) facilities, private institutional resources, Web or desktop applications, depending on the task at hand. Inevitably, the same applies to the location of relevant data, be it initial, intermediate or resulting data. The scientist is typically required to explicitly arrange data movement and processing as well as archiving and analysis of results. Furthermore, an increasingly important requirement in modern science is to support scientific collaboration. Through common projects or other means, scientists need to collaborate with colleagues working in different countries as productively as with their officemates. To collaborate means being able to share methods, datasets and results, replicate and extend other experiments, provoke discussions and generally be able to feedback newly acquired knowledge into the experimental process.

In order to automate part of this process, we need a solution, incorporating a registration framework, able to take into account the inherent heterogeneity and be consistently accessible from remote locations. Further, to support scientific collaboration, it would need to store some information about research organisations, people and scientific networks. Based on this information, such an e-science framework would be able to automate certain sections of the experimental process and facilitate collaboration.

### B. Registrable Programming and Scientific Components

In a distributed scientific setting, the information registry will need to hold information useful and relevant to a number of entities, internal or external to the platform (Figure 1). As a facilitator for information and meta-data, the registry is interrogated by services, such as the enactment service and the user interface, in oder to optimise workflow creation, execution effectiveness, parallelisation, data movement and presentation based on contextual and domain-specific information. Furthermore, it is conceivable that designers and engineers may not have complete knowledge of the extent or the form/specification of the information that would eventually be required to be maintained within the registry. Therefore, the integration with and reuse of external scientific catalogues and related resources is required. In this work, apart from the core registrable entities, we assume that the registry allows integration with arbitrary external data sources through some extensible mechanism (*e.g.* through the use of PIDS), without this mechanism being the focus of this study. This allows us to concentrate on the core entities, which would be relevant to any e-science platform that aims to deliver a distributed programming, collaborative environment: (a) processing elements and other language (for the purposes of this work, Dispel) components and (b) scientific (or research) objects:

*1) Language Components:* The language components that should be registered are those that would be required by the platform's remote enactment engines to realise and execute a workflow graph. The registry can also be seen as hosting and managing the consistent, distributed programming library, it should contain definitions and specifications of components that the domain experts and the data-intensive engineers should have at their disposal when creating a new workflow or when they are customising or modifying an existing one. In the case of Dispel, such components include all nameable entities that may appear in a Dispel program. These components are the following:

*a) Type:* the core Dispel modelling unit. Types are most frequently used to define PEs and can be broadly divided into abstract and concrete. Abstract PE types only define available connection interfaces, while concrete types specify an internal topology of PEs and other components, realising the intended functionality. Even though PE types are not the same as their enactable or executable counterparts, this difference is not crucial to the discussion of the information registry, hence we refer to both entities interchangeably.

*b) Literal:* is a named Dispel literal. Such literals may be scientific constants, data-store references, etc.

*c) Implementation:* an arbitrary block of programming code, usually implementing the functionality of a PE. Such implementations could range from scientific code written in Python to code performing job submission to HPCs, etc.

*d) Function (or constructor):* a Dispel structure used for defining arbitrary Dispel topologies, wrapping them into PE types. Functions are evaluated and expanded into enactable workflow segments during run-time.

*e) Connection:* a typed data outlet, part of the definition of PEs.

*f) Data Type:* a language-specific entity used for specifying the data types of connections of PEs. Connections may be designated as being input or output and are typically named per PE definition. Each connection may carry two types: a *structural* type, or *s-type* and a *domain* type, or *d-type*. Typical

s-types are 'Boolean', 'Stream', 'String', etc., whereas d-types depend on the application domain and can be of an arbitrary structure (*e.g.* semantic, graph-based descriptions, etc.).

*g) Workflow:* a partial or complete workflow, expressed using combinations of the elements above, able to carry out a well-specified task when deployed. Aside from other language components, workflows may also be associated with arbitrary domain-specific objects (see Section III-B2 below).

*2) Scientific and Domain-Specific Objects:* The workflow-based platform should be able to register and keep track of relevant scientific objects in a form usable by components as diverse as the user interface and the enactment layer. Domain-specific objects of interest may vary greatly in form, function and size. However, with respect to their relationship to the scientific workflow, such objects can be broadly categorised as being:

*a) External to the workflow:* are domain-specific objects which play no part in the enactment or execution of a workflow, but which have some significance to the experiment, user or other entity supported by the platform. Examples of such objects may include a relevant results data-set to be used for comparison or replication purposes, a copy of or reference to a published article to complement an experiment, etc.

*b) Internal to the workflow:* are domain-specific objects and references to datasets, which are results of specific stages of a workflow. Such objects can be expressed in terms of language components, versions of which would also be present in the registry (Section III-B1), in combination with user-specified parameters and previous outcomes. The registry should keep track of scientific objects of interest by associating language components with provenance trace information, obtained through the provenance component of the platform.

### C. Identification and Versioning

Essential to the operation of any registry or object store is a robust identification and versioning model. A registry component suitable for such distributed, collaborative workflow platform should provide for dynamic assignment of identifiers to stored digital entities. It should also provide for automatic versioning, where appropriate – i.e. when a user adds a new processing element to the distributed Dispel library, the registry should assign a persistent id to it, while it should also be able to keep track of future versions automatically. The automated approach to object identification and versioning is essential due to the complexity and potential scale of such a system.

As the registry of such system provides information to a number of other components, to keep ids and versions consistent leads to two further requirements, namely that object ids need to be consistent across remote and heterogeneous sites and computing resources as well as between past and future versions of the same container object. To illustrate this last point, consider the situation where a workflow has been registered, making use of PE $a$, and that $a$ is then modified by some researcher into $a'$. Whether $a \equiv a'$ should depend on the context of use of the workflow, and by implication, on the context of reference of the object. For provenance purposes, $a$ should always signify a different object than $a'$, however from an enactment perspective, if the same workflow were to be deployed again after the refinement of $a$ into $a'$, the latter version should probably be used by default.

To add collaboration into versioning would require that the overall identification and versioning model is able to cater for individual users and groups and that its default behaviour during object resolution takes into account current common practices. At the same time, traceability of the resolution process of individual objects and divergence from the default behaviour would also be desirable.

### D. Dynamic, Flexible Enactment and Optimisation

Optimising the enactment and execution of scientific workflows is a non-trivial research task, which requires information and meta-data regarding the available execution contexts. The role of the registry in this regard can be of central importance. The workflow platform, through an appropriate registry sub-component, should be able to query metadata regarding associated computing resources. Such metadata may include descriptions of resources in terms of processing power, disk capacity, location, average workload, front-ends or access points, supported authentication mechanisms, subscription information of relevant research groups, etc. Some of this metadata could be stored and managed internally, while there should be interfacing to external metadata resources where appropriate. Irrespective of the location of this metadata or of the internal management mechanisms, the registry should provide the workflow platform with a comprehensive and consistent API, mapping the underlying information to project-specific requirements.

## IV. Registry Design and Structure

In order to address the diverse set of requirements of our system, we build on and extend the package metaphor, which is already present in the Dispel workflow specification language [1]. This metaphor defines a hierarchical system for organising and naming digital objects of interest. In traditional programming, packages are used as containers of source code units, which are somehow related. From the perspective of the registry, packages may contain either programs (written in Dispel or other languages) or other kinds of digital objects. The registry, therefore, associates each registrable entity with a package.

Using packages to complement the Registry's identification system has a number of advantages: it is a human-readable system, which users are able to reference directly from either source code or GUIs, its hierarchical structure leads to intuitively unique fully-qualified names for the contained objects, and it is a system many users are familiar with, as it exists in other programming environments too (e.g. natively in Java, C#, etc – through explicit directory structures in other languages).

*c) Introducing Workspaces:* While the use of packages is a feature we wish to retain, for the reasons outlined above, it also has disadvantages when used for object identification in a large, heterogeneous programming environment. As briefly discussed in Section III-C, the main drawback arises when users are allows to modify registered entities, as such modifications would potentially lead to ambiguity when enacting a previously registered workflow (or an enactable subgraph of one).

Consider a concrete version of the example of Section III-C: A workflow $F$ makes use of the seismology-related PE `eu.verce.seismo.InstrumentCorrection`, which modifies a waveform to compensate for characteristics of the observing station. Suppose then that a scientist who makes use of $F$ wishes to modify the implementation of this processing element so that uses instrument correction information available to her, locally. As we are dealing with a distributed programming platform, introducing this change under the specific PE, would cause the execution of $F$ to behave differently before and after the modification, universally within the workflow platform. While this would be acceptable to the user who introduced the modification, it would be unacceptable to other users. The modified version of the PE, say `eu.verce.seismo.InstrumentCorrection`$'$, would now contain the locally desired functionality, while corresponding to the same PE signature as before.

From an execution perspective, when the platform is requested to enact and deploy $F$, during PE resolution[2], and unless a priority mechanisms has been introduced, the registry would not be able to resolve the PE in question deterministically, as there would be two versions under the same fully-qualified name (id) for same entity.

Based only on the Dispel's packaging system, this problem could be addressed by a combination of explicit version requests, alongside suitable default behaviours (e.g. "if a version has not been explicitly provided, resolve to the most recent version"). However, while this approach would take care of this problem from an enactment perspective, it would make interacting with the platform a tedious task. Assuming that such platform would be used by a potentially large number of scientists with individual and research-group agendas, default behaviours within a global scope would be of little value. On the other hand, if we were to implement local-scope default behaviours, exchanging information and collaborating with other scientists, outside of one's immediate research environment, would not be encouraged. To disallow ad-hoc modifications completely would also be an option – it would be a direct equivalent to the standard library of a traditional programming language, i.e. the open public cannot modify the library, unless the modifications are fed back to the developing community or company through official channels. This is an approach, however, that would be of little value to collaborating researchers of different, and frequently changing, requirements and it would incur a higher cost of maintenance to the platform. In order to avoid such problems, we choose to design for collaboration inside the registry itself, by providing native support for *workspaces*.

In the following two sections, we provide further discussion of the platform's building blocks which appear both in the registry as well as the workflow specification language, namely of packages, workspaces and their relationship.

### A. Packages

Dispel employs a package mechanism similar to the one found in the Java programming language. As Dispel provides a means to specify types, processing elements and other workflow components, i.e. the core elements of any workflow-based system, it is clear that the registry should also be able to reflect the language's structures, including packages. As the use of packages in programming is well understood, we don't expand further on Dispel packages in this paper.

[2]In this work, we consider the resolution mechanism as being part of the registry component due to its central role within the architecture and the coupling of its function with entities described in the registry.

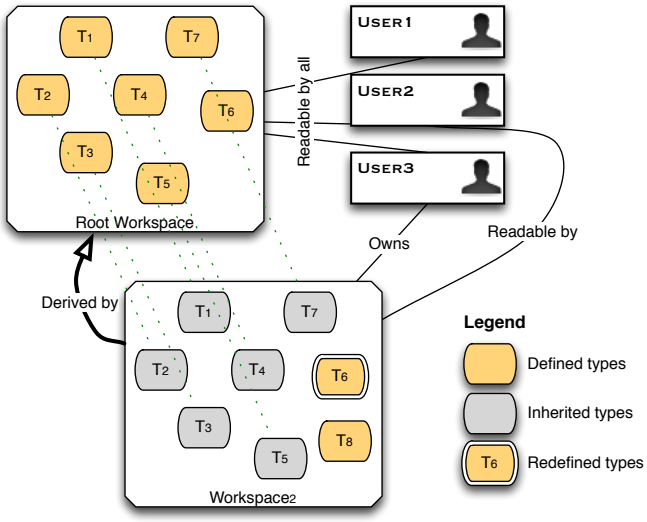



Fig. 2: An example workspace hierarchy and its relationship with users. Instead of types we could have any named, and therefore registrable, Dispel element.

### B. Workspaces

A workspace represents a snapshot of the whole ecosystem of library and other (e.g. external, domain-specific, etc.) components, as these are created, refined or being put to use within a specific, user-defined context. Workspaces are organised as a hierarchy, with each workspace being potentially an extension of another one. Each workspace contains new, modified or implicit pointers to packages of parent workspaces and, consequently, digital objects of interest defined within its parent workspace.

A workspace is typically associated with a user or a user group or with applications or experiments sharing similar goals. The exact use of a package is left to the user and to the characteristics and constraints of the application at hand. Each workspace represents a certain Dispel-based environment, which can be invoked by users and utilised for look up, resolution and execution of workflows by the enactment component.

Workspaces, as an abstraction, are orthogonal to packages, *i.e.* a workspace spans the packages of the types and other objects it contains.

On a conceptual level, each workflow component can be seen as bearing some semantic significance to the purpose or intention of the library. Often such semantics are encoded in package- and class-names, for instance, while in other cases additional information may also be employed, such as semantics-specific types (*e.g.* Dispel's d-types), relationships between components, text or other descriptions, etc. Let us

denote a conceptual workflow component as $T \leftarrow \mathbb{T}$, where $\mathbb{T}$ is the set of all conceptual components defined in the distributed workflow environment.

Users of the registry should be able to reference and introduce conceptual components, and also modify and specify/implement them. Let us denote a specification of a component as $t$, and its association to a conceptual/abstract component $T$, as $t : T$. In our model, each workflow component specification, $t$, can only be associated with a single abstract component, $T$, within each workspace. Components associated with the same conceptual/abstract entity are said to be *siblings*, denoted by $\widetilde{t_i, t_j}^T$, where $T$ is, optionally[3], the "parent" abstract type. For the purposes of this work, no assumption is asserted on the structure siblings may be organised in – *i.e.* all siblings may be on the same level, or arranged hierarchically, or otherwise, under a parent type.

*d) Workspace Definition:* A workspace then can be defined as a set of logical workflow components, such that no two components can share the same conceptual type in the same workspace:

$$W = \{t | \nexists t' \in W \ . \ \widetilde{t, t'}\}$$

In other words, conceptual types can be used as an in-workspace identifiers for logical workflow components. Given that the scope of conceptual types is the whole of the distributed workflow system, it follows (and is indeed desirable) that there will exist siblings of the same conceptual types in different workspaces, and therefore, that $|\mathbb{T}_W \cap \mathbb{T}_{W'}| \geq 0$, where $\mathbb{T}_i$ denotes the set of conceptual types for which there exist specifications in workspace $i$, for any two arbitrary workspaces $W$ and $W'$.

*e) Cascading Workspaces:* Each workspace is designed to be an independent, yet parent-complete representation of the run-time ecosystem available to the scientist or to the developer at any given time, derived either from the "standard library" of workflow components on offer, or by another customised workspace. This model of workspace definition leads to hierarchies of workspaces, much like package hierarchies, only orthogonal to them.

In terms of component inheritance, each new workspace will inherit all the components of its parent workspace *by-reference* (changes to the components in the parent workspace will be reflected on the children), while modifications to components will take place in a *by-value* fashion (*i.e.* modifications to components in the child will not be propagated up to the parent), as depicted in Figure 2. This, effectively, creates cascading hierarchies of workspaces, where member components get inherited, until modified.

*f) Single Vs. Multiple Inheritance:* The cascading workspace model presented above can be materialised either by single or my multiple inheritance, with the latter being a more general case. In the case of multiple inheritance, as in traditional programming languages, there is the need for a precedence operator, denoted as $t_1 \succ t_2$, for two workflow component specifications, $t_1$ and $t_2$. Choosing between single and multiple inheritance would depend on the application at hand and on the way workspaces would be employed during the lifetime of a system.

[3]To specify the conceptual type of the siblings is sometimes optional, as each component can have exactly one "parent", and so there can only be one parent under which two components can be siblings.

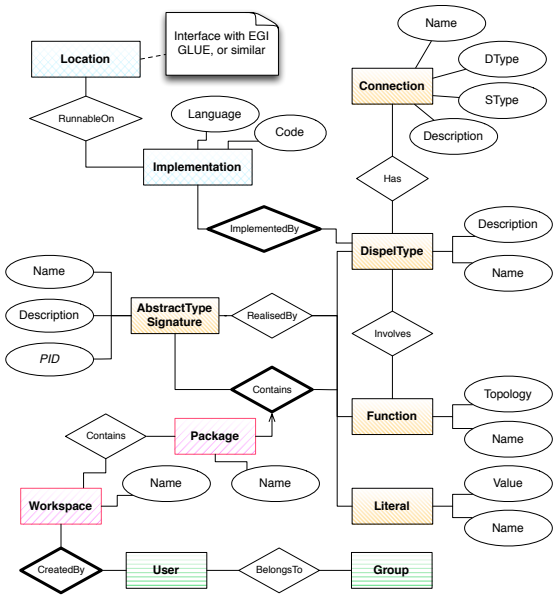



Fig. 3: An Entities-Relationships (ER) conceptual model for the e-Science registry, depicting the most important entities and attributes. Entities are colour- and pattern-coded according to the logical group relevant to their function, *e.g. Connection* and *DispelType* are part of the core Dispel registry, while *Workspace* is related to functionality pertaining to the organisation chosen within a specific application. The attributes presented are a subset of the actual attributes implemented, and many of them are instead realised as additional entities. Relationships depicted in bold are interfacing relationships between different logical parts of the registry.

If multiple inheritance were to be adopted, the precedence operator, and therefore type resolution, where not defined explicitly in Dispel, could take place over (1) the type placement in the Dispel script, (2) provenance information, such as recency of the specification of the clashing workflow component, (3) user-specified preference, etc. For the purposes of this study, we assume a single-inheritance model, leaving the exploration of alternative strategies for type resolution under multiple inheritance as future work.

*g) Relationship to Packages:* Workflow components are rarely conceived in isolation, instead being grouped with similar components or with dependencies. Related components can therefore be expected to be packaged together when accessed or modified inside or moved between workspaces. In the case of multiple inheritance, briefly discussed above, precedence could be expressed at package rather than at component level.

### C. Logical Schema

In order for an information registry component to be usable within a complex distributed system, it needs to be able to store data and interpret queries outside its core, workflow

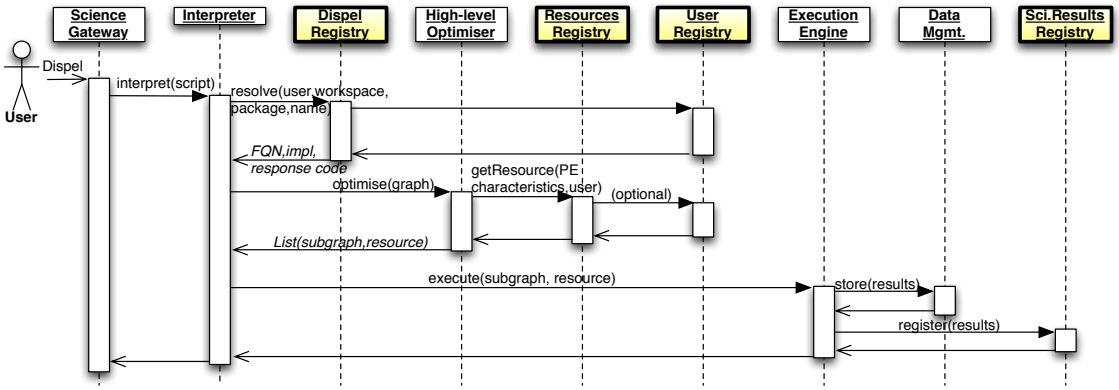



Fig. 4: A sequence diagram depicting a typical interaction between a number of components of the e-Science solution. Components which logically belong to the Information Registry appear with bold outline. Calls and responses are indicative of the intended functionality, and they should be implementable according to the registry schema of Figure 3.

specification function. Figure 3 shows the conceptual model for such registry, focusing on Dispel-specific components. Here, four distinct registry areas are shown and colour-coded: the core language-related represented by the entities *AbstractTypeSignature*, *DispelType*, *Function*, *Connection* and *Literal*; the implementation area, which pertains to specific implementations which are enactable at computing facilities with certain characteristics, and with example entities being *Location* and *Implementation*; the organisational area, which might implement different policies depending on the application at hand, and with representative entities *Package* and *Workspace*; and the user-management area, with representative entities *User* and *Group*.

For reasons of interoperability and sustainability, certain logical areas of the registry need to be made compatible with known and well-accepted standards, so as to take advantage of useful third-party catalogues and registries. For instance, the implementation logical area of the registry, which is designed to interact with the enactment and optimisation services of the workflow platform, could retain compatibility with schemas such as EGI GLUE[4], primarily in the European context, or with constituent Dublin Core[5] schemas for resource description. Similarly, the user-management area could be made compatible with relevant Dublin Core schemas, such as FOAF[6]. In maintaining interoperability and consistency between such workflow information registry and external catalogues, the registration of appropriate prefixes through initiatives such as the European Persistent Identifier Consortium – EPIC[7] is essential.

### D. Language Implications

As the reason behind introducing the workspaces construct is to aid collaboration and ease the resolution of workflow components, it needs to be communicated to the interpretation and execution components of the e-Science system either via injects in the Dispel language or via the user-interface. For demonstration purposes, here we assume that a set of Dispel directives has been implemented. Such a design decision, in combination with sensible domain-specific default behaviour, aids collaboration and adds value to the overall platform without jeopardising the flexibility and generality of the Dispel language.

Examples of using such language directive, extended to take into account potential user roles, from within Dispel are the following:

```
// Default reference to a workspace:
@using workspace MyWorkspace
// Extending a single workspace:
@using workspace MyWorkspace(MyParentWorkspace)
// Multiple workspace inheritance - conflicts may be
// resolved by right-to-left precedence:
@using workspace MyWorkspace(MyParent1, MyParent2)
```

In the following section we show how the collaboration model described above may be used in a seismology context.

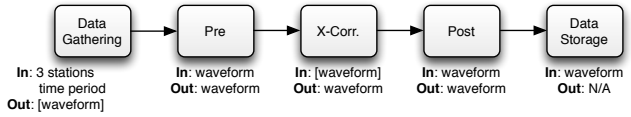
## V. Application Scenario: Ambient Noise Cross-Correlation in Seismology

The Green's function of the medium between two seismographic stations can be deduced from the inter-correlation of the seismic noise recorded at these stations [2] During recent years, rapid practical implementation of these innovative statistical methods, both in seismology and acoustics, have lead to breakthroughs in high-resolution imaging – in seismology and exploration geophysics – and acoustic communications [6].

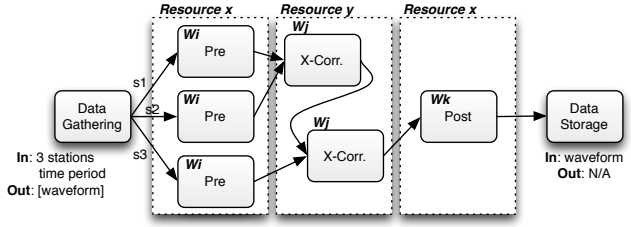
The calculation of cross-correlation is performed on waveforms retrieved from arbitrary numbers of stations, on regional or global levels and over varying time periods. This entails gathering, managing and processing large volumes of data from a number of arbitrary sources, potentially annotating and storing byproducts, before cross-correlating the traces;

[4]http://go.egi.eu/glue2-standard
[5]http://dublincore.org
[6]http://xmlns.com/foaf/spec/
[7]http://www.pidconsortium.eu

therefore making this a data-intensive scientific task. Cross-correlation is a parallelisable procedure, which can be executed either on a local cluster or at an external facility. Post-processing, visualisation and interactive steering typically follow the cross-correlation calculation.




(a) Abstract workflow for ambient noise cross-correlation.




(b) Workflow for ambient noise cross-correlation after high-level optimisation.

Fig. 5: Workflow stages for ambient noise cross-correlation before and after initial, high-level optimisation.

Focusing on the interactions of the Information Registry and the user as well as the other components, and without loss of generality, a cross-correlation experiment can be seen as having three broad phases: (1) pre-processing of multiple waveforms; (2) cross-correlation and (3) post-processing (which may include visualisation). Before processing can commence, data needs to be gathered from relevant sources, while at the end of the workflow result data will typically need to be stored through some appropriate data-management service. In typical cases, intermediate results correspond to hundreds of thousands of files. Let us assume that the abstract workflow consists of one pre-processing, one cross-correlation and one post-processing component, as in Figure 5(a).

The Dispel information registry provides information to enable the optimiser to extract appropriate PE implementations. Such information includes user-specified workspaces (explicit or derived), *e.g.* $W_i, W_j, W_k$ of Figure 5(b). The parallelisation strategy may be derived by information such as the types of the input and output streams of PEs (*e.g.* whether they expect single or multiple input streams), by observable indicators such as output/input ratio as well as by other properties. By consulting the registry about resources, the optimiser can also make an informed decision about which resource (*e.g.* $x, y$ of Figure 5(b)) should be responsible for each workflow segment. Furthermore, the presence of such a registry component ensures that PEs and workflows, such as the ambient noise cross-correlation presented here, can be exchanged between users and still be resolvable and executable consistently throughout the e-Science platform. By storing provenance and other metadata, such as workspace names of PEs involved, the choice of resources and the workflow segmentation, the system through the shared information registry, can support the controlled re-enactment of scientific workflows and the consistent replication of past experiments.

## VI. Conclusions and Future Work

In this paper we presented the role of an information registry within a Dispel-powered e-Science framework. Further, we provided a suitable design for such a registry component taking into account the distributed nature of the target system and of the way scientists conduct research as well as their collaboration needs. We demonstrated how a mechanism of workspaces can help retain consistency across what is essentially a distributed programming platform, while being meaningful to users.

However, the design of such a wide-ranging e-Science workflow platform is far from complete. Pointers for future work include to research and specify appropriate APIs which would allow interoperability with other e-Science platforms and registries, the integration of external stores and catalogues in an extensible and domain-unaware fashion as well as, crucially, to kick-off an iterative design process with increasing numbers of members of interested scientific communities.

## References


[1] Malcolm Atkinson, Rob Baxter, Peter Brezany, Oscar Corcho, Michelle Galea, Jano van Hemert, Mark Parsons, and David Snelling. *THE DATA BONANZA: Improving Knowledge Discovery for Science, Engineering and Business*. John Wiley & Sons Ltd., April 2013.

[2] Michel Campillo and A Paul. Long-range correlations in the diffuse seismic coda. *Science*, 299(5606), 2003.

[3] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[4] Bartosz Dobrzelecki, Amrey Krause, Alastair Hume, Andrew Grant, Mario Antonioletti, Tilaye Alemu, Malcolm P. Atkinson, Michael Jackson, and Elias Theocharopoulos. Integrating Distributed Data Sources with OGSA-DAI DQP and Views. *Philisophical Transactions of the Royal Society A*, 368(1926):4133–4145, 2010.

[5] William H. Dutton and Paul W. Jeffreys. *World Wide Research: Reshaping the Sciences and Humanities*. MIT Press, 2010.

[6] Erica Galetti and Andrew Curtis. Generalised receiver functions and seismic interferometry. *Tectonophysics*, 532535(0):1 – 26, 2012.

[7] Anthony J. G. Hey, Stewart Tansley, and Kristin Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[8] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole A. Goble, Matthew R. Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34:729–732, 2006.

[9] Edward A. Lee and Stephen Neuendorffer. MoML – A Modeling Markup Language in XML – Version 0.4. Technical report, University of California at Berkeley, March 2000.

[10] Chee Sun Liew. *Optimisation of the enactment of fine-grained distributed data-intensive workflows*. PhD thesis, School of Informatics, University of Edinburgh, 2012.

[11] Xavier Llorá, Bernie Ács, Loretta S. Auvil, Boris Capitanu, Michael E. Welge, and David E. Goldberg. Meandre: Semantic-Driven Data-Intensive Flows in the Clouds. In *IEEE Fourth International Conference on eScience*, pages 238–245. IEEE Press, 2008.

[12] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *SIGMOD Conference'08*, pages 1099–1110, 2008.

[13] Toby Segaran and Jeff Hammerbacher. *Beautiful Data: The Stories Behind Elegant Data Solutions*. O'Reilly, 2009.

[14] Arie Shoshani and Doron Rotem. *Scientific Data Management: Challenges, Technology and Deployment*. Computational Science Series. Chapman and Hall/CRC, 2010.