SemSorGrid4Env

FP7-223913

Deliverable

1.2v2

Deployment of Technological Infrastructure

Juanjo Aparicio, Jesús Gabaldón

December 31st, 2009

<Status: Final>

<Scheduled Delivery Date: December 31st, 2009>

Executive Summary

This document provides a deployment strategy for the architecture described in D1.3v1, "SemSorGrid4Env Architecture – Phase I". Here we summarize the selection and deployment strategy of the technological infrastructure that the consortium maintains as development, deployment and testing environment for SemSorGrid4Env project. We also detail the status of the deployment of the technological infrastructure as it is at M16 into the project.

The major achievements are the identification of the technologies to support the development of the SemSorGrid4Env software; the deployment of these technologies in the server ssg4env.techideas.net, comprising two Tomcat servers, the Axis2 Web Services container and the OGSA-DAI Web Service; and the deployment of a continuous integration server (Apache Continuum), and a Maven repository (Apache Archiva).

Note on Sources and Original Contributions

The SemSorGrid4Env consortium is an inter-disciplinary team, and in order to make deliverables self-contained and comprehensible to all partners, some deliverables thus necessarily include state-of-the-art surveys and associated critical assessment. Where there is no advantage to recreating such materials from first principles, partners follow standard scientific practice and occassionaly make use of their own pre-existing intellectual property in such sections. In the interests of transparency, we here identify the main sources of such pre-existing materials in this deliverable:

- Sections 1 and 2 contain material from D1.3v1 “SemsorGrid4Env Architecture – Phase 1”
- Section 2 contains material from D5.1 “Specification of high-level application programming interfaces”

Document Information

Contract Number	FP7-223913	**Acronym**	SemSorGrid4Env
Full title	SemSorGrid4Env: Semantic Sensor Grids for Rapid Application Development for Environmental Management		
Project URL	www.semsorgrid4env.eu		
Document URL	http://www.semsorgrid4env.eu/home.jsp?content=/sew/viewTerm&content=instance.jsp&_sew_var_name=instance&_sew_instance=D1.2+v2&_sew_instance_set=SemSorGrid4Env&_origin=%2Fhome.jsp		
EU Project officer	Antonios Barbas		

Deliverable	**Number**	1.2	**Name**	Deployment of technological infrastructure		
Task	**Number**	1.1	**Name**	Analyse, select and deploy sensor network and Semantic Grid technological infrastructure		
Task	**Number**	1.2	**Name**	Track sensor network and semantic grid standards and practice		
Work package	**Number**	1		Software Architecture and Middlware for Semantic Sensor Grids		
Date of delivery	**Contractual**		31/12/09	**Actual**	31/12/09	
Code name				**Status** draft ☐	final ☑	
Nature	Prototype ☑ Report ☐ Specification ☐ Tool ☐ Other ☐					
Distribution Type	Public ☑ Restricted ☐ Consortium ☐					
Authoring Partner	TI					
QA Partner	UNIMAN					
Contact Person	Jesús Gabaldón					
	Email	jesus.gabaldon@tecnalia.es	**Phone**	+34 902 109 443	**Fax**	+34 932 917 727
Abstract (for dissemination)	This document provides a deployment strategy for the architecture described in D1.3v1, "SemSorGrid4Env Architecture – Phase I". Here we summarize the selection and deployment strategy of the technological infrastructure that the consortium maintains as development, deployment and testing environment for SemSorGrid4Env project. We also detail the status of the deployment of the technological infrastructure as it is at M16 into the project.					
Keywords	Infrastructure sensor network semantic grid					

Version log/Date	**Change**	**Author**
0.1	**First version**	**Miguel Vidal**
0.2	**Updated from comments on version 0.1**	**Juanjo Aparicio**
0.3	**Update with status at month 16**	**Juanjo Aparicio**
0.4	**First QA recommendations**	**Juanjo Aparicio, Ixent Galpin**
0.5	**Second round of QA recommendation**	**Juanjo Aparicio, Ixent Galpin, Jesús Gabaldón**
1	**Final Version**	**Juanjo Aparicio, Ixent Galpin, Jesús Gabaldón**

Project Information

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number FP7-223913. The Beneficiaries in this project are:

Partner	Acronym	Contact
Universidad Politécnica de Madrid (Coordinator)	UPM	Prof. Dr. Asunción Gómez-Pérez Facultad de Informática Departamento de Inteligencia Artificial Campus de Montegancedo, sn Boadilla del Monte 28660 Spain #e asun@fi.upm.es #t +34-91 336-7439, #f +34-91 352-4819
The University of Manchester	UNIMAN	Prof. Carole Goble Department of Computer Science The University of Manchester Oxford Road Manchester, M13 9PL, United Kingdom #e carole@cs.man.ac.uk #t +44-161-275 61 95, #f +44-161-275 62 04
National and Kapodistrian University of Athens	NKUA	Prof. Manolis Koubarakis University Campus, Ilissia Athina GR-15784 Greece #@ koubarak@di.uoa.gr #t +30 210 7275213, #f +30 210 7275214
University of Southampton	SOTON	Prof. David De Roure University Road Southampton SO17 1BJ United Kingdom #@ dder@ecs.soton.ac.uk #t +44 23 80592418, #f +44 23 80595499
Deimos Space, S.L.U.	DMS	Mr. Agustín Izquierdo Ronda de Poniente 19, Edif. Fiteni VI, P 2, 2º Tres Cantos, Madrid – 28760 Spain #@agustin.izquierdo@deimos-space.com #t +34-91-8063450, #f +34-91-806-34-51
EMU Limited	EMU	Dr. Bruce Tomlinson Mill Court, The Sawmills, Durley number 1 Southampton, SO32 2EJ – United Kingdom #@ bruce.tomlinson@emulimited.com #t +44 1489 860050, #f +44 1489 860051
TechIdeas Asesores Tecnológicos, S.L.	TI	Dr. Jesús E. Gabaldón C/ Marie Curie 8-14 08042 Barcelona, Spain #@ jesus.gabaldon@techideas.es #t +34.93.291.77.27, #f ++34.93.291.76.00

Contents

List of Figures

1.Introduction

1.1.Scope

This document summarizes the set of infrastructural tools that the consortium has set up as development and deployment environment for SemSorGrid4Env project. Version 2 of the document, the current one, updates the first version with the changes introduced in the technological infrastructure up to month 16 in the project (December 2009).

1.2.Overview and Context within SemSorGrid4Env

WP1 is concerned with the definition and validation of architecture of the SemSorGrid4Env, plus the setup of the technological infrastructure and tools necessary for the development of the project software artifacts.

[D1.3v1] specifies the architecture of the SemSorGrid4Env software platform, aimed at supporting "the rapid development of applications over data sources, including both sensor networks and traditional database, in which significant value is derived by the use of semantic techniques for service discovery and data integration".

Figure 1 provides a diagram of the interactions between the components of the platform. There are three tiers, namely the application, middleware and data tiers. The Application tier provides services to the applications, allowing them to collect information from a variety of data sources and present them in an integrated manner.

The Middleware tier provides the means for data sources and services to be registered and later discovered and accessed. The Data tier provides access to data sources such as sensor networks and legacy databases, by means of the uniform interface provided by the Data Source, implemented by the Connectivity Bridge. For a given concrete resource the connectivity bridge may (eg JDBC) or may not exist.

The Data Sources are registered, with the proper semantic annotations, in the Semantic Registry. The Semantic Integrator queries the Semantic Registry in order to discover Data Sources.

The Application Services make use of the Semantic Registry to discover these Data Sources, and use the Semantic Integrator to obtain information from different sources in an integrated manner, instead of only juxtaposed (section 2.2.1 of [D1.3v1]).

As can be seen in the figure, any service is able to call any other service. This includes applications and application services getting data from the Data Sources once the relevant ones have been discovered with the help of the Middleware tier services.

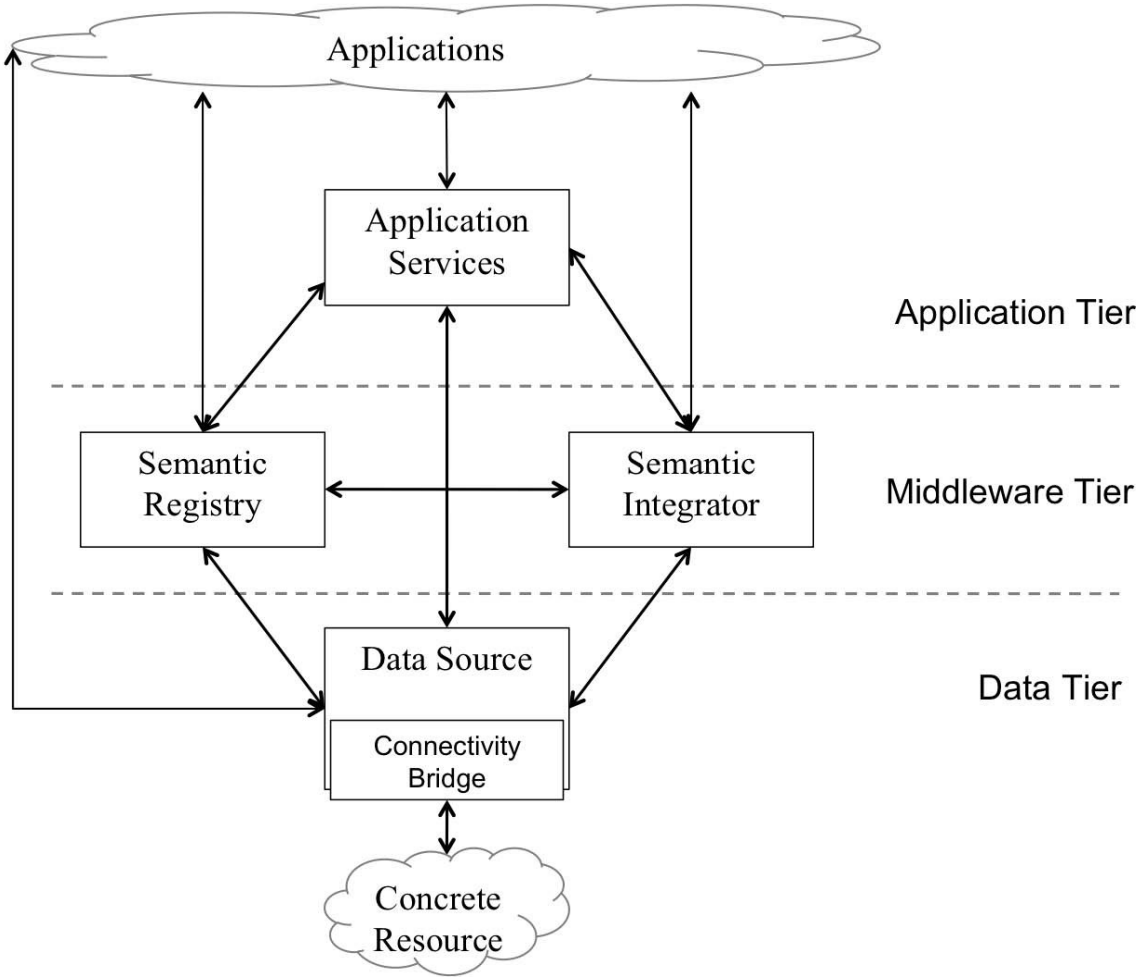

Figure 1: High-level overview(section 2.1 of [D1.3v1])

1.3.Objectives

This document provides the assessment and choice of implementations that support the realization of the SemSorGrid4Env architecture, and documents the current status of the deployment of these implementations for the purposes of development, testing and integration.

The architecture document specifies that the architecture "comprises a set of Web Services". We have to choose the suitable supporting technology for the development of Web Services.

In the architecture document, the interfaces for the Web Services are specified. Some of them are based on WS-ResourceFramework. Although it would be enough to identify a framework that supports the creation of Web Services and use it to develop the service providers and consumers that conform to the WS-RF specifications, it is convenient to find a library or framework which provides support for the WS-RF set of specifications, in order to reuse existing work and decrease the implementation effort.

1.4.Document Structure

This document is organized as follows: section 2 provides an overview of the requirements for the technological infrastructure needed to support the development of the project software components. Section 3 provides the selection of components needed to fulfill the specified requirements and the status of its deployment in the integration and testing environment. Section 4 describes the tools that have been set up to support the distributed collaborative development effort. Section 5 concludes.

2. Requirements Arising from the SemSorGrid4Env Architecture

The focus of this deliverable is on the selection and deployment of a technological infrastructure for the support of the middleware tier between the data sources and the mashup applications. The middleware will support the interactions between the various components, in terms of the syntax and the semantics of their interfaces.

2.1. SOA and Interfaces

The SemsorGrid4Env architecture document [D1.3v1] states that a Service Oriented Architecture will be used for the interaction between components in the project. A SOA emphasizes loose coupling between components, thereby allowing a greater flexibility in the choice of implementation details for each of the individual components. This is needed to allow for the arbitrary orchestration of software components, which is intended to support the development of on-the-fly mashups.

In order to guarantee the interoperability between components of the system, and allow their parallel evolution, it is important to identify the relationships between them, and to define these interactions, in terms of the syntax and the meaning of the operations.

The architecture document specifies what are the interfaces of the services involved in the middleware. These interfaces use the WS-ResourceFramework [WSRF] and WS-Data Access and Integration [WS-DAI] standards.

2.2. Application Tier

The application tier interface, specified in [D5.1], provides “mechanisms […] through which applications can access sensor network components and associated data for presentation and interaction with end-users”.

The SemsorGrid4Env architecture combines the SOA and REST architectural approaches for Web Services. The REST style “will be exposed to support application development with minimal a priori design such as Web mash-ups ” ([D1.3v1]).

While there is nothing in the REST architectural style [REST] that makes it specific to HTTP, the static Web, that is, restricted to static pages, is an implementation of a RESTful architecture. “The key principle of REST is the use of *resources*”, and “these resources [...] are then accessed through the URI, usually using HTTP” ([D5.1]).

The technological infrastructure must therefore provide the means to allow software components to interact with the middleware by using raw HTTP.

2.3.Middleware Tier

The middleware tier must support both synchronous and asynchronous modes of interaction. Synchronous modes include the use of SOAP to perform remote procedure calls (RPC) and the manipulation of resources via HTTP in a RESTful manner. Asynchronous modes include publish-subscribe and queue-based mechanisms.

2.4.Data Tier

The data tier must provide virtualizations of concrete resources via the appropriate Connectivity Bridge. This bridge must conform to the interfaces specified in section 4.3 of [D1.3v1]. These include the Service Interface (section 3.1 of [D1.3v1]) and the Registration Interface (section 3.2 of [D1.3v1]), which are based on [WSRF]. It is therefore desirable to provide a framework which supports WSRF instead of just being able to generate clients and services for the WSRF interfaces. This would allow the development effort to progress more quickly.

The data tier services must also provide implementations of the Query Interface (section 3.5 of [D1.3v1]), which is that of [WS-DAI]. This includes further specializations of that interface, such as WS-DAIR, WS-DAIX, WS-DAI-RDF(S) and WS-DAI-Streaming.

3.Selection of Software Infrastructure

It is necessary to identify the software components that can fulfill the specified requirements. In broad terms, we will need a web application container for the web application libraries, a web services container and framework, and implementations of the standards specified in [D1.3v1]. The selection of the tools and frameworks to be installed is based on suitability and availability.

The following sections provide a more concrete description of the needs raised by the requirements, and a description the software components that fulfill those needs, keeping in mind the needs and selection of components for the other requirements.

3.1.Application Tier

For the application tier it is necessary to provide the means that will allow the implementation of components that may interact via raw HTTP. A servlet container allows the deployment of components written in the Java language to send and receive requests and responses that use the HTTP protocol.

Additional technologies an libraries will be added to the integration and development environment as the needs arise.

3.2.Middleware Tier

The middleware tier must support the deployment of components that interact via SOAP and raw HTTP in a RESTful manner. The Axis2 ([Axis2]) Web Services container provides facilities to allow services to be invoked in a RESTful manner as well as via SOAP messages. The Axis2 Web Services container needs a servlet container to be run.

The WS-ResourceFramework specification encompasses other specifications, in particular WS-ResourceProperties and WS-Notification. While it is possible to use a generic WS framework to generate the skeletons for services that comply with the WS-ResourceFramework set of specifications, it requires the programmer to concentrate on implementing remote procedure

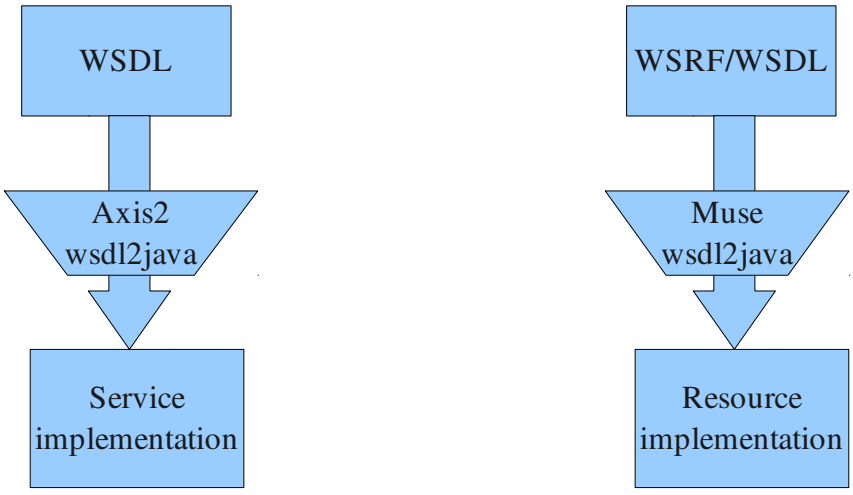

Figure 2: Generic WS framework vs Muse

calls. The WS-ResourceFramework is centered around an abstraction, the resource, and defines a

limited set of operations that manipulate the resource. For example, for very simple resource properties, Muse generates the implementations of the methods that set and get their values, and requires the programmer to set an initial value. Of course, programmers may override the implementations of the methods specified in [WSRF], and provide alternate implementations, eg when the values of properties are not stored as variables in the resource instance or must be acquired from remote sources.

The Apache Muse framework provides an implementation of WS-ResourceFramework that provides to developers classes that represent resources instead of just procedure calls.

Muse services require only an Axis2 web services container to be run. The tools provided with the Muse framework produce as output a web application, which can be deployed in a web application container like Tomcat.

In addition to the above WS standards, secure and reliable communications between components are needed. The Apache Axis2 modules Rampart and Sandesha provide implementations of the WS-Security and WS-ReliableMessaging standards, respectively. Rampart provides developers with the means to request, provide and check access credentials. Sandesha provides developers with the means to detect whether communications errors have taken place, and the means to attempt to remedy the situation. An extensive description of these modules is out of the scope of this document.

3.2.1.RDF Engine

The registry metadata is to be expressed in RDF, as specified in section 6.1 of the architecture document. There are various implementations of engines that provide access to metadata in RDF. The main alternatives for Java-based implementations are Atlas [Atlas], Jena [Jena] and Sesame [Sesame].

An option would be to use the Atlas P2P storage [Atlas], part of OntoGrid, which is in turn reliant on the Bamboo DHT. This would allow for a distributed storage of RDF and RDFS data. Atlas allows the execution of RQL queries on the metadata.

Jena is a framework for building semantic web applications, which includes, but is not limited to, RDF APIs and a SPARQL [SPARQL] query engine.

Sesame is a framework for the storage, query and reasoning with RDF and RDF Schema [RDFSchema].

Jena seems to be a good tool for the implementation of the applications, given its focus on the development of semantic web applications. Sesame seems to be a simpler approach if all is needed is a front-end to a set of RDF statements, and the distribution of this information and the distributed access to it is provided by the components that interact with Sesame. Only actual usage will tell what is the best choice for the project needs, and we are therefore willing to change the recommended RDF back-end technology.

3.2.1.1.Month 16 Update

[D1.3v1] specifies in section 6.1.1 that the Atlas system will be used for the implementation of the registry service. An Atlas node has been set up and is available at ssg4env.techideas.net, port

8898. The Atlas node has a web interface that allows to check the configuration, it is available at http://ssg4env.techideas.net:8889/.

3.3.Data Tier

The architecture document [D1.3v1] identifies three main classes of data sources: repositories containing documents normally created, read, updated and deleted by REST web services, repositories of structured data according to a data model (for example, a relational database), and sensor networks. Section 2.2.3 of the architecture document specifies that the first class of data sources is virtualized by default; the second and third classes of data sources are to be virtualized by means of query processing technologies. For the second class, structured data, we will use "existing approaches, such as WS-DAI in which requests are ultimately specified in terms of SQL, XPath/XQuery, or SPARQL." For the third class, WS-DAI will be used for data for which is stored or for which the temporal aspect is not relevant. For data streams, a continuous query language such as [SNEEql] will be used.

OGSA-DAI is a mature implementation of WS-DAI, based on Axis1 [Axis1].

For the deployment of the above implementations it is necessary to identify a core container that is able to provide their runtime environment. The selection is explained in section 3.6.

3.4.Data Access

For the access to data by using WS-DAI, OGSA-DAI [OGSA-DAI] 3.1 has been chosen. It requires either the Globus Toolkit [GlobusToolkit] or Apache Axis for Java 1.4 [Axis2]. We have opted for the latter due to its lighter weight. We haven't found so far any need for the Globus Toolkit to be present in the development and testing environment. The Globus Toolkit Java WS Core can be obtained separately from the Globus Toolkit. It can be run from within an Apache Tomcat installation.[1]

An Apache Tomcat 5.5 server will be the container for the various middleware components that will be created. The latest stable version is 6.0.18 is also provided, for the components that do not need the older version.

We have chosen to include the relevant libraries in the server classpath, for simplicity. If classpath-related problems arise during the development of the various components, it may be needed to switch to an installation with a clean classpath, and where each component includes its own version of the needed libraries.

3.4.1.Month 16 Update

The OGSA-DAI web application has been installed and is available at the Tomcat5 server. It can be reached through the URL http://ssg4env.techideas.net:8080/dai/.

Providing the OGSA-DAI web application in the Tomcat6 server is an ongoing effort. The status of this activity will be reported in the next version of this document.

[1]http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#javawscore-admin-tomcat-deploying

3.5.WS-Notification / WS-ResourceProperties

The Apache Muse project provides an implementation of the WS-ResourceFramework collection of specifications, including WS-Notification and WS-ResourceProperties.

However, Muse is provided as a toolkit for the generation of Muse-based web services. Muse-generated web applications include an instance of the Axis2 1.1 web application, allowing programmers to develop their Muse services and deploy them onto a standard servlet container like the Tomcat instance provided to the members of the consortium.

3.5.1.Month 16 Update

Muse uses and embeds version 1.1 of the Axis2 web application. This situation precludes the usage of Axis2 modules created for later versions, such as Rampart (for WS-Security) and Sandesha (for WS-Reliable Messaging). Work is underway to provide a solution to this problem, by allowing Muse to be used with higher versions of the Axis2 web application. One possibility is to patch the muse distribution. The tests performed so far show that by carefully changing a single directory in the Muse distribution it is possible to generate services that embed another version of Axis2. More through tests are needed to ensure that this is applicable as a general method.

3.6.Core Server

The available implementations of WS-DAI and WS-RF described above are deployable as web applications. We need a web application container to support the deployment of the chosen frameworks and of the services developed with those frameworks.

We have chosen Apache Tomcat [Tomcat], as it is a mature implementation of a web application container, popular and well supported.

The various libraries for the particular technologies should be made available to both the developers and the runtime environment provided by the servlet engine.

The usage of a server like Apache Tomcat makes it also easy to develop the components that deal with a pure HTTP interface. These components are to be used by the Web applications and mashups that are to consume the data provided by layers deeper in the architecture. These applications provide the (graphical) end-user interface to the system.

3.6.1.Month 16 Update

Two Tomcat servers have been set up in our development and testing environment. Tomcat 5.5 can be reached through http://ssg4env.techideas.net:8080, while a Tomcat 6.0 server is available at http://ssg4env.techideas.net:8180.

The reason for having two different versions of the same web application container running in parallel is to minimize the impact of incompatibilities between versions of a particular library or framework. By allowing older versions to coexist with newer ones, development of functionality can continue while solutions for the incompatibility problems are sought after and applied.

3.7.Java 5 EOL

The version 5 of the Java Developers Kit has reached its end-of-life period in October 2009. That means that Java5 is no longer available to the general public from the http://java.sun.com website, although an archived version is available from http://java.sun.com/products/archive/, requiring registration in the web site. The current version of the JDK with general availability is Java6.

This has an impact on the choice of frameworks and libraries deployed so far. The implications are described in the following sections.

3.7.1.Axis2

The latest version of Axis2 is 1.5. The previous version is 1.4.1. Both of these versions use a library for the parsing of XML documents, called StAX, short for Streaming API for XML.

Java6 includes the StAX API, version 1.0. However, Axis2 packs, and seems to rely on, version 1.0.1. This causes the execution of Axis2 1.4.1 to fail if run on Java6.

One commonly used workaround is to "patch" the JDK by putting the proper StAX API jar file in the endorsed directory in the JRE. In that way, the classes in the StAX API jar file take precedence over the classes in the StAX API in the JRE itself, thereby allowing Axis2 (and other libraries which depend on the syntax and semantics of modern versions of the StAX API) to work with Java6.

The latest StAX API jar file can be obtained from http://stax.codehaus.org/Download. The patching operation consists on copying this file in the directory jre/lib within the JDK directory.

The tests performed so far with version 1.2 (final) of the stax library show that Axis2 can be run this way in Java6. However, additional tests must be carried on to ensure this is a proper solution.

3.7.2.Muse

The Muse framework inherits the limitations of the Axis2 web services container that it reuses. It is therefore expected that solving the problems with Axis2 will solve the problems with Muse, as these stem from the same circumstances, namely the APIs for XML parsing. This is a work in progress that will be completed during the next period.

3.7.3.OGSA-DAI

Once the problem with Axis2 has been sorted out (see section 3.7.1 for information on the current status) OGSA-DAI, which is based in [Axis1], has to be tested to see whether the same patching technique allows it to run properly. This is a pending effort for the next period.

3.8.Summary of the Deployed Infrastructure

Figure 1 depicts the deployed technical infrastructure as it is in month 16. Dashed boxes indicate components that are being developed or will be developed. Solid boxes indicate components that are already finished or that belong to third parties, and that provide support for the development activities.

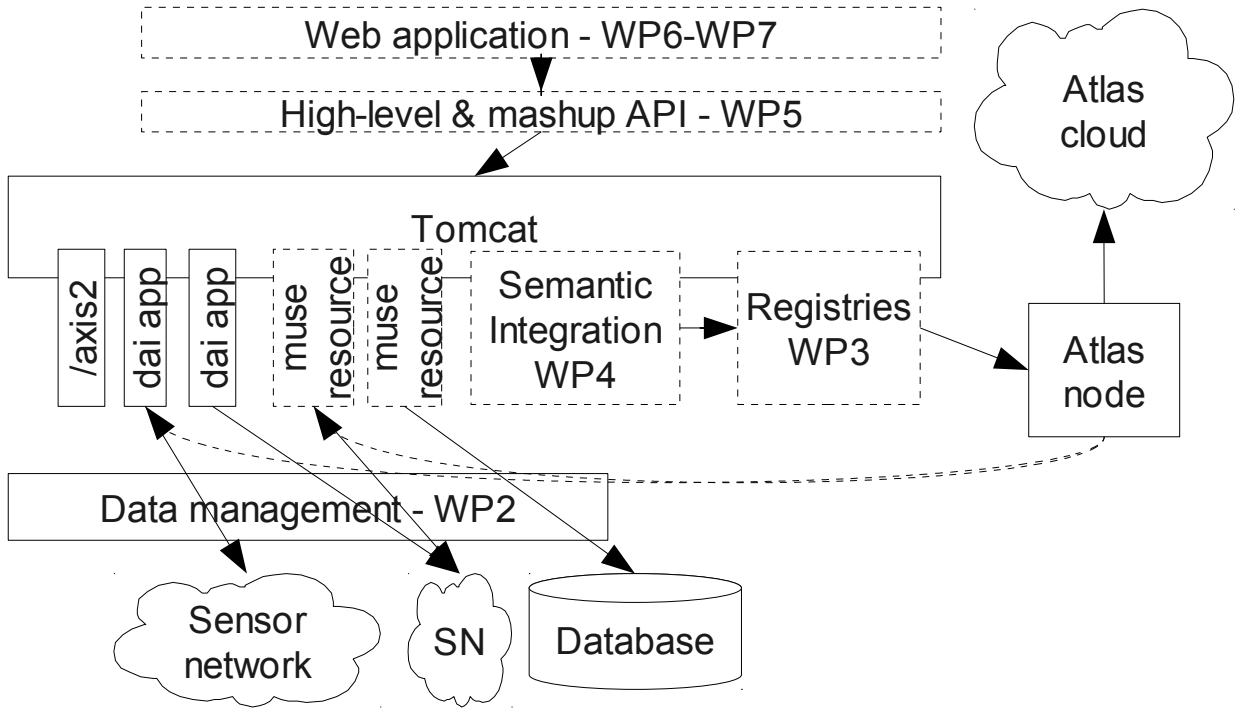

Figure 3: Deployment infrastructure

4. Software Development Tools

The software to be produced by this project is to be built by a team of developers that reside in separate locations in the world, spanning various time zones. It is necessary to provide the members of the development team with the tools that will help ensure adequate communication, collaborative writing of source code, and means to integrate and test the individual components, subsystems and the system as a whole. This section details the tools and mechanisms that are being provided to developers within the project to help provide the needed software artifacts (libraries, services, etc).

4.1. Source Code Repository

Whenever a pice of code is developed by more than one programmer it becomes necessary to use some kind of source code management tool. This need becomes greater as the number of developers grows, and is especially valuable when these developers are a long way from each other and the opportunities for side-by-side work are limited.

While there are several options offering the same basic functionality, we have chosen Subversion as the source code management tool, due to its maturity, cheap branching abilities, and the fact that it can be used through HTTP or HTTPS, which allows most firewalls to be traversed, thereby allowing the most developers to work with it.

We have deployed a Subversion server in the project development server, which is available at https://ssg4env.techideas.net/repos. Access is restricted to consortium members.

4.2. Integration Process

In a project where development is performed by various teams it is important to try to avoid divergences about the different teams assumptions, goals and contributions to the project. Integration meetings are a way to ensure proper alignment of the different parties, but the cost of such integration meetings grows with the time passed between meetings. By diminishing the time between these integrations their cost also diminishes, but if integration meetings break the normal pace of development their impact on the project will be negative.

4.2.1. Continuous Integration [ContinuousIntegration]

A way to deal with the divergent forces is to include integration in the development cycles from the very beginning. By defining early on how the different pieces are to be integrated, and by providing the tools and mechanisms to perform the integration in an easy and cost-effective way it is possible to reap the benefits of early integration with a moderate cost.

There are automatic tools for the integration of Java code from different parties, such as CruiseControl and Apache Continuum. The latter works best when combined with Maven, a project management and build process. For additional information on Maven, see page 20.

We have set up an Apache Continuum server for continuous integration. It is available at the URL http://ssg4env.techideas.net:8180/continuum.

4.3.Build Process

By defining a common build process it is possible for every partner that performs development to build and test the code of other partners. The build process has to make clear the following points:

- The dependencies of the artifact to be generated, and how to obtain them
- The way in which the artifact is generated
- The way in which the artifact is tested and used

There are several widespread alternative tools that can be used for project building. The following sections explore some of the possibilities.

4.3.1.Ant

Apache Ant was developed as a Java-only replacement for Make. It makes few assumptions about the way in which the project code is built, allowing the developer to customize its behaviour to suit the project needs. Conversely, its flexibility comes with a price: for non-trivial tasks a moderate amount of programming (in Ant's own scripting language) is needed.

For more information, see [Ant].

4.3.2.Maven

Apache Maven “is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information” [Maven].

Maven makes more assumptions about the project structure that Ant does, thereby limiting the freedom of developers who want to work with it. However, these assumptions are basic enough as to not to limit the usefulness of Maven.

On the plus side, as these assumptions are shared by all developers that work with the tool, it is easier for the parties involved in the development to build each other's code, to run tests, etc. This set of assumptions allow users of the Maven tool to stick to the do not Repeat Yourself (DRY) principle, concentrating on the business logic and devoting less time to the plumbing necessary to build the software artifacts.

The Maven style of development includes unit testing in form of Junit tests as part of the development cycle. If there are failed tests, the build fails as a whole. This helps ensure that no broken code is ever shipped. Of course, it is possible to fool the system by commenting out the tests. The point is to avoid accidental leakage of broken code into the production environment.

Maven provides means to track dependencies on libraries and to obtain these dependencies from library repositories. There is a central repository of dependencies, supplemented with a big set of

mirrors around the world. Maven also allows users to set up project- or group-specific dependency repositories.

The integration Maven provides with continuous integration (see page 19) tools such as Continuum makes it a better choice than alternatives like Ant.

An additional point in favor of Maven is that developers do not need to choose between the two. A project that is being built with Ant may be slowly converted to a project being managed by Maven. The descriptors that allow Ant and Maven to work do not conflict with each other, therefore allowing a developer to slowly move from an Ant-based build process to a Maven-based build process.

4.3.3.Conclusion

While Ant is better known and more popular than Maven, there are significant benefits to be drawn from the use of Maven as a build tool. Therefore, we will provide the means for developers to start using Maven.

4.4.Testing

Testing of the behaviour of components is usually performed as the last phase of the development cycle in a traditional waterfall approach. It is through testing that the implemented solution is validated against the specification.

However, by relegating testing to the last phase of the development cycle its value is diminished, because any implementation of design flaws found at this stage should be fed back to the proper development stage (implementation, design, even requirement analysis). That is why it is encouraged to do the "waterfall" at least twice [Waterfall]. Spiral or iterative software development methodologies tighten the development cycle and perform several iterations of the analysis-design-implement-test cycle.

4.4.1.Automated Testing

The ability to run and validate massive quantities of tests in an automatic way allows for developers to incorporate exhaustive test suites in their development cycle. Such tests provide guidelines for the code that is being developed, by allowing the developer to focus on specific aspects of the system under development.

Automatic tests also allow developers to check whether new code breaks the semantics of existing one, provided that there are automatic tests that check these semantics.

An important psychological aspect of automatic tests is an improved sense of security.

4.4.2.Unit Testing

In contrast with integration testing, unit testing attempts to validate a minimal portion of the implementation code. The boundaries of what is considered "unit" and "minimal" are not specified, and is basically left as a decision to the developer. A basic guideline is that a "unit" is the minimal part of a system that can be tested in isolation.

Unit testing helps by checking that the interfaces between the units are preserved in both their syntax and semantics.

4.4.3.Test Driven Development

The Test Driven Development [TDD] technique emphasizes the importance of unit tests and their value as guidelines for development by reversing the traditional approach to development, and putting the testing phase before the code is written.

Tests that reflect requirements are written before the code that fulfill those requirements. Then, the minimal amount of code necessary to pass the tests is written. The motivation for this approach is twofold. First, the tests provide an extremely focused perspective of the system. Second, by insisting in the minimalistic approach, very little code is written in each iteration. Less code also means less probability of bugs leaking into the code.

Even though not all developers may feel comfortable with the shift from code-test development to test-first programming, the use of an automated testing framework and of development tools that support it is a great aid during debugging phases of development. It is therefore important to have the proper tools to make automated testing unobtrusive.

The Test Driven style of development requires a unit testing framework to be easily integrated in the development process. The Junit framework is a mature implementation of a unit testing framework, and is already integrated in many development tools such as the Eclipse IDE and the Maven building tool.

4.5.Issue tracking

It is important in a distributed group of developers to have means to communicate about the implementation effort. While email, conference calls and instant messaging solutions provide means to easily communicate in electronic form, these mechanisms are mainly detached from the source code developers are working on, and require proper organization to ensure that all points raised are dealt with.

Issue tracking tools help solve this problem by providing a central point of communication between developers of software components, allowing for the organization of the issues discovered, their prioritization, and the linking with the relevant software components, source code lines and involved developers.

We have chosen to install and provide to the consortium developers the TRAC issue tracking tool, due to its stability, light weight, and the fact that it provides a view on the subversion repository, allowing to watch progress in an easy way, and a wiki, which can be used for development-centered discussions.

The TRAC server is available at http://ssg4env.techideas.net/trac/

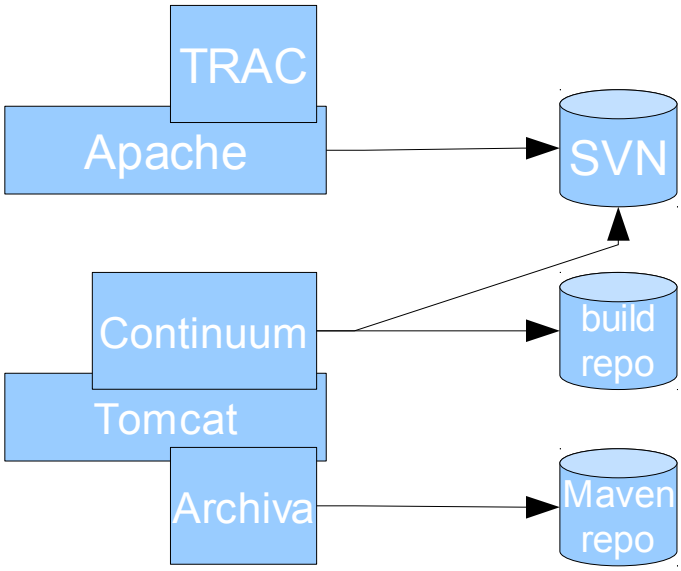

Figure 4: Development environment tools

Figure 4 shows how the subversion repository is made available through the proper Apache module, and how the TRAC server, deployed in Apache as well, provides a view of the SVN repository. Archiva and Continuum are web applications deployed in the Tomcat 6.0 server, and provide each access to a different repository: continuum to the continuous integration build repository, and archiva to the Maven artifact repository.

5. Conclusion

In this document we have considered the requirements specified in [D1.3v1], and have identified a set of frameworks, libraries and components that will allow developers to provide the SemSorGrid4Env platform implementation.

The process of search, assessment and selection of the recommended libraries and frameworks has required an iterative approach, testing the candidate implementations, due to the lack of a single framework that provides an implementation of all the required interfaces.

The recommended set of components includes the Tomcat servlet container, as a means to provide access to Web Services containers and as a container of web applications that will be used by the Applications, as can be seen in Figure 1; the Axis 2 Web Services container, which can be deployed in an instance of Tomcat and which provides the means to implement SOAP and REST Web Services; the Muse framework, which provides an implementation of the WS-ResourceFramework specification; OGSA-DAI as an implementation of WS-DAI, which was identified in [D1.3v1] as one of the primary means to virtualize data resources; and the Atlas server as the basis for the RDF(S)-based Semantic Registry.

We have provided a deployment and testing environment with the needed tools, for use of the consortium members, as a development, testing and integration environment.

We have also analyzed the set of tools that a distributed group of developers would need in order to perform the development activities for the elaboration of the realization of the architecture, deciding to use Subversion as the source code repository for its maturity, cheap branching abilities and its ability to be used via HTTP, allowing its use from behind a corporate firewall; we decided to include an issue tracking tool to help identify problems in the development process and concentrate the effort on the most important ones. Again, it was necessary to test various available solutions, until a set of simple, working and available tools was identified.

We have provided an infrastructure for the distributed development that includes these tools, including source code repository, issue tracking tool, dependency repository and continuous integration tool.

6.Appendices

This chapter includes additional information that doesn't fit the structure of the previous sections.

6.1.Glossary of Terms

RDF

Resource Description Framework. Metadata data model, based upon the idea of making statements about resources.

WS-DAI

Web Services Data Access and Integration. Family of specifications for the definition of web service interfaces to data resources, such as relational or XML databases.

WS-Notification

Web Services Notification. Family of related white papers and specifications that define a standard Web services approach to notification using a topic-based publish/subscribe pattern.

WS

Web Service. Defined by the W3C as “a software system designed to support interoperable machine-to-machine interaction over a network”.

Mashup

Web application that combines data and/or functionality from more than one source

SPARQL

RDF query language. W3C Recommendation since January 2008.

RQL

Query language for RDF and RDFSchema.

RDF Schema

Extensible knowledge representation language, providing basic elements for the description of ontologies. Intended to structure RDF resources.

OGSA-DAI

Middleware solution that allows resources, such as relational or XML databases, to be accessed via the grid.

Globus Alliance

Community of organizations and individuals developing fundamental technologies behind the "Grid," which lets people share computing power, databases, instruments, and other on-line tools securely across corporate, institutional, and geographic boundaries without sacrificing local autonomy.

Globus Toolkit

Open source software toolkit used for building Grid systems and applications. Developed by the Globus Alliance and others.

7. Bibliography

[D1.3v1]: Alasdair J G Gray, Ixent Galpin, Alvaro A Fernandes, Norman W. Paton, Kevin Page, Jason Sadler, Manolis Koubarakis, Víctor M. Díaz and Israel Liébana, "SemSorGrid Architecture - Phase I", 2009

[WSRF]: "Web Services Resource Framework", http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

[WS-DAI]: Mario Antonioletti, Amy Krause, Norman W. Paton, Andrew Eisenberg, Simon Laws, Susan Malaika, Jim Melton, Dave Pearson, "The WS-DAI family of specifications for web service data access and integration", 2006

[D5.1]: Kevin R. Page, David C. De Roure, Kirk Martinez, and Jason Sadler, "Specification of high-level application programming interfaces", 2009

[REST]: Roy T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", 2000

[Axis2]: "Axis2 User's Guide", http://ws.apache.org/axis2/1_4_1/userguide.html

[Atlas]: "Storing and Querying RDF Data in Atlas", http://atlas.di.uoa.gr/publications/FD24-Kaoudi.pdf

[Jena]: "Jena Semantic Web Framework", http://jena.sourceforge.net/

[Sesame]: "User Guide for Sesame 2", http://www.openrdf.org/doc/sesame2/users/

[SPARQL]: "SPARQL Query Language for RDF", http://www.w3.org/TR/rdf-sparql-query/

[RDFSchema]: "RDF Vocabulary Description Language 1.0: RDF Schema", http://www.w3.org/TR/rdf-schema/

[SNEEql]: Christian Y. A. Brenninkmeijer, Ixent Galpin, Alvaro A. A. Fernandes, and NormanW. Paton, "A semantics for a query language over sensors, streams and relations", Proceedings of 25th British National Conference on Databases (BNCOD25), July2008

[Axis1]: "Axis", http://ws.apache.org/axis/

[OGSA-DAI]: University of Edinburgh, "OGSA-DAI 3.1 Documentation", 2008

[GlobusToolkit]: "Globus Toolkit Version 4: Software for Service-Oriented Systems", http://www.globus.org/alliance/publications/papers/IFIP-2006.pdf

[Tomcat]: "Apache Tomcat", http://tomcat.apache.org/

[ContinuousIntegration]: "Continuous Integrationl", http://martinfowler.com/articles/continuousIntegration.html

[Ant]: "Apache Ant User Manual", http://ant.apache.org/manual/index.html

[Maven]: "What is Maven?", http://maven.apache.org/

[Waterfall]: "Managing the development of large software systems", http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

[TDD]: "Introduction to Test Driven Design", http://www.agiledata.org/essays/tdd.html