Title: **Configuration Tool Implementation**
Work Package 6

Authors: Omer F. Rana, Arnaud Contes and Vikas Deora

Editor:

Reviewers:

Type: Deliverable (D6.3.1)

Version: *Rev* : 198 (svn)

Date: August 20, 2006

Status: Final

Class: Confidential

Summary

The purpose of this document is to define the configuration process and subsequently demonstrate its implementation within a provenance system. Particular focus is placed on the configuration process required for the tool suite, such as the configuration necessary for the analysis engine and the user portal. The relationship between the setup protocol and the configuration process is also described. Performance analysis of the analysis engine and the user portal is reported based on p-assertions from OTM/EHCR and DLR applications.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

Members of the PROVENANCE Consortium

IBM United Kingdom Limited	United Kingdom
University of Southampton	United Kingdom
University of Wales, Cardiff	United Kingdom
Deutsches Zentrum für Luft- und Raumfahrt e.V.	Germany
Universitat Politecnica de Catalunya	Spain
Magyar Tudomanyos Akademia Szamitastechnikai es Automatizalasi Kutato Intezet	Hungary

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

Foreword

This document describes:

- Configuration management issues within the provenance system and the tool suite.
- Configuration support provided for different user roles in the tool suite – such as a System Administrator, an Application Administrator, an End User, etc. These roles have also been defined in the document.
- User interface support provided for configuration management in the Portal.

The primary audience of this document include: administrators of the provenance system, application managers making use of the tool suite, and application users (particularly those making use of a workflow engine) employing tools to better understand the behaviour of their applications (based on previously submitted p-assertions).

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

Contents

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

List of Figures

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

List of Acronyms

- HTTP: HyperText Transfer Protocol.
- MIME: Multimedia Internet Mail Extension.
- URI: Uniform Resource Identifier.
- VFS: Virtual File System.
- OTM: Organ Transplant Management.
- EHCR: Electronic HealthCare Records.
- GUI: Graphical User Interface.
- PS: Provenance Store.
- CIFS: Common Internet File System.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

1 Introduction

The operations necessary to support configuration management, the types of configuration options that are made available within a particular provenance system, and mechanisms for the use of these configuration options within an application, are described in this document. Configuration management can be supported for various components that make up a provenance system. These range from:

1. Configuring the overall architecture – i.e. identifying the components involved within a provenance system, such as the location and number of Provenance Stores, the recording actors, the location of the tool suite, etc.
2. Configuration of individual components that make up a provenance system, such as the Client Side Library, the tool suite, the Provenance Store, etc.
3. Configuration of the security mechanism being employed within the provenance system – such as role-based access control, use of digital certificates, username/password-based access management, etc.
4. Configuration of components within the tool suite – such as the Analysis and Navigation tools, etc.
5. Configuration of individual portlets within the Navigation tool – for instance, allowing users to modify the "view" on particular p-assertions that have been retrieved from one or more Provenance Stores.
6. Configuration of the setup protocol – identifying the number and types of stages allowed for a particular provenance system (as described in Deliverable D6.2.1 from WP6).

A given type of configuration may only be undertaken by a particular category of user. For instance, application end users may not be allowed to modify the security mechanism being used – whereas system administrators may be allowed to modify any of the above configurations. The focus within this document is primarily on configurations 4, 5, and 6 listed above. The following types of users may be involved in configuration management:

- *Provenance System Administrators*: These individuals would be responsible for managing the overall configuration process. They are responsible for acting as an overall coordinator for the configuration management process.
- *Application Administrators*: These individuals are responsible for configuring the interaction between the application, the tool suite and the Provenance Store. Such users must therefore provide configuration parameters required within the setup process – in collaboration with application end

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

users. The outcome of the setup process is the creation of a configuration file for use by application administrators to deploy on one or more application instances.

- *Tool Suite Administrators*: These individuals are responsible for ensuring that all the tools (navigation, analysis, etc) are ready and available for use. They may be the same as application administrators – or may be part of the overall Provenance system administration team. Any errors produced during the setup process must be logged by the tool suite administrators. The error messages may subsequently be analyzed by application administrators.

- *Application End Users*: These individuals make use of the portal and the rule engine to interact with a particular application. The portal may have been pre-configured by the *Application Administrators*, but such users would still be able to modify these configurations based on their preferences. Such modifications are generally supported via a menu-based interface.

This user classification is based on the description provided in Deliverable D6.1.1. A new role has been identified by one of the application end users in the Provenance project (DLR), which is that of a *Project Manager*. A project manager is responsible for configuring the types of interactions that can take place between an application user who belongs to the project, the tool suite and the Provenance Store.

1.1 Purpose of the Document

The purpose of this document is to explain the configuration management process (with particular focus on the tool suite) and its relationship to other components in the Provenance system. A distinction is made between setup and configuration:

- Setup involves identifying the operations that are necessary to enable an actor to record or to query a Provenance Store. Such operations will make use of functionality provided within a client application, the tool suite and the Provenance Store.

- Configuration involves identifying the parameters that may be associated with a setup process. Essentially, the setup protocol focuses on the set of activities (or processes), whereas configuration focuses on specific instances of these activities, with support for generating particular values for the parameters associated with each activity in the protocol.

1.2 Document Overview

This document is structured as follows:

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

- Section 2 describes the overall configuration management process – identifying various components of a Provenance system that need to be configured, and the possible period over which configuration changes are possible.
- Section 3 contains a description of configuration management for the Tool Suite. An overview of the types of possible configurations being supported is provided in this section, followed by a detailed discussions in sections 5 and 4. The relationship between the setup protocol (described in deliverable D6.2.1) and configuration management is explained.
- Section 6 identifies how the configuration management process can be applied in a non-provenance aware application. This section demonstrates how the configuration process can be adapted for an application that does not make direct use of a Client Side Library to submit p-assertions.
- Conclusion and possible further work is presented in section 8.

1.3 Links to other Provenance Documents

This document makes use of content in the following existing Provenance project documents:

- WP2: Requirements identified in D2.2.1. In particular, the focus is on requirements that impact the setup of the Provenance system – such as SR-1-10, SR-1-11, SR-1-17, SR-1-18, SR-6-3, and SR-7-2. The last of these is particularly significant, as it states that the Provenance system should be loosely coupled to the application that makes use of it.
- WP3: The frozen architecture document D3.1.1. Configuration management is discussed with reference to the architectural components discussed in section 3.3 of "An Architecture for Provenance Systems" – such as the relation between the client-side libraries and the Provenance Store.
- WP6: Tools Deliverables D6.1.1 and D6.2.1 – such as the operations necessary to support the navigation and analysis tools. Configuration management is related to the setup protocol, identifying how parameters needed to configure various stages of this protocol are specified. Configuration of the Portal, discussed in deliverable D6.2.1, is also described in this document, along with examples of use in the context of the OTM/EHCR and DLR application scenarios.

2 Configuration Management

In this document, configuration management refers to the ability of a user to specify values to parameters associated with the Provenance system, using either a Graphical User Interface (GUI) or a command-line based tool. The values

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

associated with parameters may be entered manually or they may be read from an existing file. Where multiple possible values can be associated with a given parameter, such values may be chosen using a pre-defined menu (or another equivalent user interface mechanism). Constraints may be associated with such values, to limit them to a pre-defined range or to a pre-defined type (such as numeric, string, etc). Such constraints need to be defined by a System administrator at setup time. The following aspects of configuration management need to be considered:

- User role: this aspect determines the types of configuration options that should be made available to a particular user role. Such roles have already been identified in deliverables D6.1.1 and D6.2.1 (and provided in section 1 of this document). The application administrator is provided with greater configuration privileges than the application end user, for instance, when making use of the tool suite.
- Time period: this aspect determines the time at which a particular type of configuration should be allowed. For instance, it may be necessary to identify the location of a Provenance Store at setup time only. Alternatively, the types of views available to an application end user on the recorded provenance data may be configured by the end user involved.
- Type: this aspect identifies the "types" of configurations which are allowed – i.e. what aspects of the provenance system may be configured, and what components in the provenance system are potentially configurable. A provenance system administrator may restrict the configuration of some components in the system from being modified.
- Mechanism: this aspect identifies how configuration changes should be supported, essentially identifying the method used to achieve a change in configuration for a particular provenance system component, or for the system as a whole. The mechanism used may range from uploading a pre-defined text file, to making configuration changes using a pull down menu.

It is assumed that an overall *Provenance System Administrator* is responsible for the above aspects of a provenance system. Such individuals would also be involved in setting up the system in the first instance (as discussed in deliverable D6.2.1), and subsequently defining particular user roles within the system.

3 Tool Suite Configuration

3.1 Tool Suite Overview

The tool suite from WP6 is composed of different types of components as shown in figure 1 – these include:

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

- The *processing tools*: these tools provide features accessible through an Application Programming Interface (API). They are designed to be used within larger applications. These tools include the following:
  - The Analysis Engine provides reasoning capabilities over a set of p-assertions,
  - The Comparator Tool may be used to compare p-assertions that have been submitted by an application,
  - the Query Tool makes use of the Client Side Library to query one or more Provenance Store(s).
- The *visualisation tools*: these tools provide Graphical User Interfaces (GUI) for visualizing p-assertions that have been submitted by an application (potentially making use of a workflow).
- User interaction is supported through the eXo portal to provide a standard way to invoke the Setup protocol and the Configuration Tool.
- Various portlets have also been developed to provide different views to the recorded p-assertions. Portlets can exchange data with each other via the eXo portal container. New portlets may also be added – and are made visible by publishing the portlets within a registry (that is part of the portal framework).

3.2 Configuration Types

In order to provide a greater degree of flexibility in configuring the tool suite, each tool is developed to be stateless. The use of this approach prevents the need to bind the tool suite to a particular configuration architecture, and prevents the introduction of new dependencies between the tools.

Nevertheless, as a demonstration of how such a configuration architecture could be used, WP6 provides an example of a configuration architecture. This architecture has been developed as a set of portlets, and can be used alongside other portlets that are publicly available over the Internet or from commercial vendors that support the JSR168 and WSRP portal standards (as discussed in Deliverable D6.1.1).

The configuration portlet relies on a database to provide the configuration settings requested by the tools. We describe several mechanisms of configuration that could be implemented.

- a static configuration approach is by far the easiest, and involves specifying the configuration within the source code of the program. Unfortunately, it is also the least flexible and easy to update approach. Thus, this type of configuration should only be limited for development/debugging purposes.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

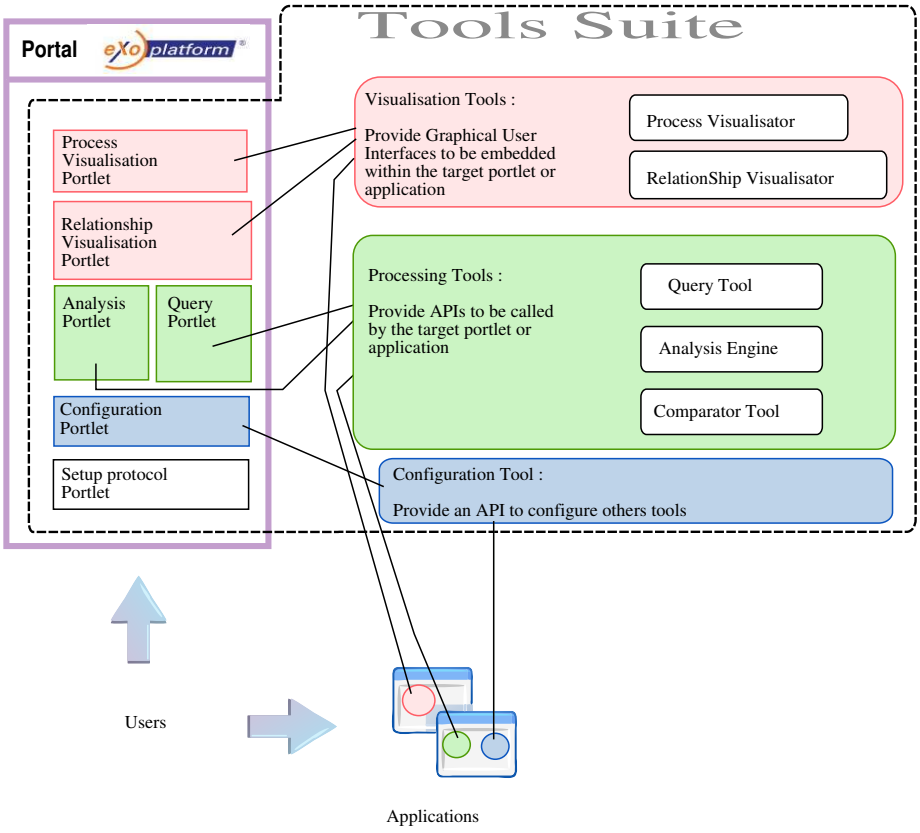

Figure 1: *Components included within the Tool Suite*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

- An alternative approach is to write the configuration into a file and pass it to the tool suite at start-up. The storage format that could be used can vary, and depends essentially on the application context. It could be a simple set of key/value pairs, or may be a structured XML document, or may be in a binary file format only readable by the application.
- An extended version of the previous approach would be to allow the file to be remotely accessible, thereby allowing the sharing of the configuration between the tools. This approach remains a read-only solution.
- The use of a database can be described as the recommended type of configuration. A database is recommended when the configuration could be updated during the execution of an application or when several parties have access to a commonly shared configuration. The database approach also provides a number of features such as concurrent access support, immediate propagation of an update or deletion, etc.

3.3 Relation to Setup Protocol

The setup protocol document (D6.2.1) contains a description of the steps necessary to make an application provenance aware. We now discuss the configuration provided by an application manager for the tool suite at step S1 described in D6.2.1.

In order to enable an application to use a provenance system, several packages of the Provenance Architecture have to be already installed:

- one or more Provenance Store(s) must be installed, see section 3.3.1;
- several instances of the Client Side Library, one per actor.
- any requested Documentation Style Transformations, the configuration is done per actor, see section 3.3.3;
- the security sub-system, see 3.3.5;
- The portal server.

Note that the configuration of the portal and the tool suite are developed latter in the document (sections 4 and 5).

3.3.1 Configuring the Provenance Store

The Provenance Store represents a key component of a provenance system, as it is responsible for storing p-assertions submitted by an application. Thus, the location of at least one Provenance Store must be included in any configuration of a provenance-aware application. The location of a Provenance Store is represented using the Uniform Resource Location (URL) format.

The installation of a Provenance Store from WP9 includes the installation of a number of dependent software packages:

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

- the eXist 1.0 XML database,
- the Globus Toolkit version 4.0.2 (Java WS Core).

The configuration of a Provenance Store is performed by individuals belonging to the Provenance System Administrator role. It includes the creation, modification and deletion of a Provenance Store, and the broadcasting of these events to the Application Administrators. The notification can be achieved in a number of different ways – from a manual process: sending an email to all Application Administrators involved, to a more complex process that would involve an update to the database of Provenance Stores shared by the Provenance System Administrators (with read/write/update access) and by the Application Administrators (with read-only access).

3.3.2 Configuring the Client Side Library

Each actor in a provenance system makes use of the Client Side Library (CSL) to communicate with a Provenance Store. This communication could be either a query or a submission of provenance-related data – such as a description of an interaction, a snapshot of the actor state at a particular moment of time during application execution, etc.

In order to be able to contact the Provenance Store, the CSL needs to know the location of, at least, one Provenance Store. This is a mandatory parameter. The list of available Provenance Stores is gathered by the Application Administrator from the Provenance System Administrators. It is worth mentioning that an application could be publishing to, or retrieving data from, several Provenance systems that make use of several Provenance Stores.

3.3.3 Configuring Documentation Style Transformation

When an actor documents an interaction, it constructs an interaction p-assertion which states the content of a message received or sent by the asserting actor. This message needs to be qualified by the asserting actor so that querying actors can understand the nature of the transformation that was applied to an application message. This is the purpose of the Documentation Style. More details about Documentation Style can be found in sections 6.5 and 8.5 of the Logical Architecture document (D3.1.1).

The Documentation Style framework provides a set of basic transformations for documenting a p-assertion. Each transformation can be configured by defining a transformation definition document. This is an XML file that provides information on how a specific transformation is to be performed. An instance of a transformation definition document is shown in listing 2 – explanations of this document are given in section 3.3.4. Each transformation has a set of parameters associated with it; the transformation is performed using the transformation definition document as input to supply values for these parameters. The transformed input is put into a p-assertion along with a URI pointing to

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

the transformation definition document, which should now be made available at a publicly accessible site.

As describes within the PStruct schema, each p-assertion contains an element named `DocumentationStyle` that describes the transformation used on this p-assertion. The default value is called `Verbatim`, which means that the content of the p-assertion contains the original message as it has been sent by the actor. By default, a Documentation Style Transformation provides two operations:

- The *forward* transformation occurs when an actor submits a p-assertion through the Client Side Library. This transformation modifies the content of the p-assertion submitted by the client by applying the configured Documentation Style Transformation. This transformation is undertaken within the Client Side Library and without any further interaction with the actor. A field within the structure of the p-assertion is updated to describe the transformation used. Then the *transformed* p-assertion is sent to the Provenance Store.
- The *reverse* transformation is used to reverse the effect of a Documentation Style Transformation on the content of the p-assertion. It is usually used on a p-assertion extracted from a Provenance Store. When a reverse transformation is applied, the field describing the Documentation Style Transformation used on the p-assertion is set to `Verbatim` and the content of the p-assertion contains the message originally sent by the actor.

The transformation document has to be deployed by the Application Administrators on every actor involved in a particular application.

3.3.4 Configuring the VFS Documentation Style Plugin

The TENT application from DLR (aerospace engineering, WP7) does not send contents during method calls, but rather stores the data to be transferred onto an external storage location (a WebDAV server [6]) and only sends the location of such data as part of a p-assertion. This behaviour is equivalent to the use of a Documentation Style plugin that would upload the content of the interaction to a remote server.

In order to allow the retrieval of the information stored on the remote server, a Virtual File System (VFS) Documentation Style plugin has been developed by the WP6 team. This plugin makes use of a Virtual File System architecture to access the data stored outside the Provenance Store. The VFS architecture used in this plugin comes from the Jakarta Commons [12] "Common Virtual File System" [11] project. The Commons Project is dedicated to creating and maintaining reusable Java components, enabling the collaboration and sharing of code between developers from the Jakarta community.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

Commons developers make an effort to ensure that their components have minimal dependencies on other libraries, so that these components can be deployed easily. In addition, Commons components will keep their interfaces as stable as possible, so that Jakarta users (including other Jakarta sub-projects) can implement these components without having to worry about changes in the future. This VFS has also been chosen due to the high numbers of file systems supported within it – thereby ensuring that formats other than WebDAV can also be accommodated. The following formats are supported:

- CIFS: provides access to the files on a CIFS server, such as a Samba server, or a Windows share.
- Local Files: provides access to the files on the local physical file system.
- HTTP and HTTPS: provides access to files on a (non-)secure HTTP server.
- FTP and SFTP: provides access to the files on a (non-)secure FTP server.
- WebDAV: provides access to files on a WebDAV server.
- Zip, Jar and Tar (uncompressed, tgz or tbz2), gzip and bzip2: provides read-only access to the contents of Zip, Jar, Tar, gzip and bzip2 files.

All filenames are treated as URIs, i.e. the reference to a file located on a WebDAV server would be `webdav://somehost:8080/file.xml`

This plugin is provided as an add-on that could be used with any Client Side Library, the only requirement is to include the provided jar (that contains the plugin classes) and the jar from the Jakarta VFS project into the `classpath` of the actor's Java Virtual Machine. The plugin is used to reverse the *virtual* forward transformation required by the TENT application. The forward transformation is called *virtual* because it is not undertaken by the plugin or the TENT application itself. The TENT application uploads the content directly to the WebDAV server (external to the provenance system). Thus the forward method does not alter the content of the p-assertions. The reverse transformation downloads the content from the remote server (in this case the WebDAV server), and then replaces the location reference within the p-assertion with the actual content from the server.

In order to facilitate the use of this transformation, an XML schema has been defined to specify how to encode the location of the remote data and its type. The schema is shown in figure 2 and an example defining a text content location at the URL would be `webdav://localhost/davfiles/text.txt` on a WebDAV server.

The type of the remote content can be specified by the application user. According to the type used, the transformation plugin will import the content differently. Possible values for the type attribute are:

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

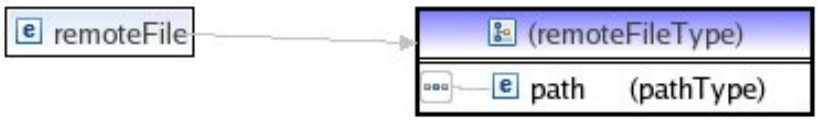

Figure 2: *Virtual File System schema*

- `text` or `xml`: if this type is specified, the content of the *transformed* p-assertion is replaced by the content of the remote file.
- `binary`: this type specifies that the remote content is a binary file (PDF, image, MS-Excel or MS-Word document, etc). In this case, as an XML document cannot contain binary data, the content is first encoded using a base64 algorithm and then inserted into the p-assertion instead of the *transformed* content.

```
1 <vfs:remoteFile xmlns:vfs="http://www.gridprovenance.org/schemas/tools/vfs.xsd">
2     <vfs:path type="text">webdav://localhost/davfiles/text.txt</vfs:path>
3 </vfs:remoteFile>
```

Listing 1: VFS example : text content

As any Documentation Style transformation, the VFS transformation requires a transformation definition document. An instance of this definition document is shown in listing 2. Lines 7 and 8 define respectively the input format, i.e. the document that is going to be transformed (XML in this case) and the output format, i.e. the result of the transformation. Line 10 specifies the Documentation Style Transformation to use, each transformation is defined by a unique identifier. Lines 13 and 14 specify the namespace of the document before and after the transformation. The key parameter in this definition document is the `<Accessor>` element (line 16) that specifies how to find the nodes that contain a reference to the remote content. In this example, the location is encoded using the schema previously introduced. However, it would also be possible for an Application Administrator to define their own schema and to use it as long as the specified node contains the URL of the remote file, and an attribute `type` specifying the type of the file.

```
1 <transformDefinition
2   xmlns="http://www.gridprovenance.org/documentationstyle/transformdefinition.xsd"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:ps="http://www.gridprovenance.org/documentationstyle/transformdefinition.xsd"
5   xsi:schemaLocation="http://www.gridprovenance.org/documentationstyle/transformdefinition.
        xsd">
6
7  <inputTechnology>http://www.gridprovenance.org/documentationstyle/dataTypes/XML_TYPE
       </inputTechnology>
8  <outputTechnology>http://www.gridprovenance.org/documentationstyle/dataTypes/
       XML_TYPE</outputTechnology>
9     <transformOperation>
```


The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

```
10        <transformType>http://www.gridprovenance.org/documentationstyle/transformTypes/
              webdav</transformType>
11        <webdav>
12          <nameSpaceMapping>
13            <ps:originalNameSpace>http://www.gridprovenance.org/schemas/tools/vfs.xsd</
                  ps:originalNameSpace>
14            <ps:transformedNameSpace>http://www.gridprovenance.org/schemas/tools/vfs.xsd<
                  /ps:transformedNameSpace>
15          </nameSpaceMapping>
16          <Accessor>//*[ namespace-uri() = "http://www.gridprovenance.org/schemas/tools/vfs
                .xsd" and local-name() = "remoteFile"]</Accessor>
17        </webdav>
18    </transformOperation>
19  </transformDefinition>
```

Listing 2: VFS Transformation Definition Document

3.3.5 Configuring the Security Architecture

The Security architecture makes use of a Role-Based Access Control (RBAC) [9] mechanism, as stated in section 4 of the Security Requirement deliverable document D4.1.1.

RBAC is an approach for restricting system access to authorized users that take on particular "roles" within a system. Hence, it makes use of roles that have been created for various job functions within an organization. The permission to perform certain operations ('permissions') are assigned to specific roles. Members of staff (or other system users) are assigned particular roles, and through those role assignments acquire permissions to perform particular system functions.

Since users are not assigned permissions directly, but only acquire them through their role (or roles), management of individual user rights becomes a matter of simply assigning the appropriate roles to the user, which simplifies common operations such as adding a user, or changing a user's department.

RBAC differs from access control lists (ACL's) used in traditional discretionary access control systems, as it assigns permissions to specific operations that have meaning in the context of a particular organization, rather than to low level data objects. For example, an access control list could be used to grant or deny write access to a particular system file, but it would not say in what ways that file could be changed. In an RBAC-based system an operation might be to create a 'credit account' transaction in a financial application or to populate a 'blood sugar level test' record in a medical application. The assignment of permission to perform a particular operation is meaningful, because the operations are fine grained and themselves have meaning within the application.

Thus, in order to configure the Security architecture for an application, the Application Administrators have to:

- create certificates for every actor of the application. By actors, we imply both software components and users that make use of the application,
- define the security roles according to the application,

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

- bind all actors to their designated roles,
- define the authorization rules for each role defined before.

If several security domains are involved, the Application Administrator must propagate this configuration to all security domains. In the case of a single security framework shared by all the components involved (Provenance Stores, eXo Portal, Client Side Library, etc) in the application, the configuration only needs to be undertaken once. However, if the application crosses multiple security boundaries, such as in the Organ Transplant Management application (WP9), the configuration must be propagated to all security subsystems involved in the application. The way to propagate this configuration could be by editing manually the configuration of each subsystem, or by making use of tools that can maintain coherency between all the components. This aspect of security (i.e. dealing with multiple types of security mechanisms) is not being considered in the project, as we assume that a single security subsystem is in place.

At a systems configuration level, inter-domain authentication is achieved through the GT4 security framework. Each independent domain has to maintain its own root certificate (issued by some valid Certificate Authority); hence incoming requests from clients in a different domain cannot be authenticated properly as their certificates were signed or issued by a different Certification Authority. In order to support inter-domain authentication at the level of the GT4 message level security framework, it is possible to either install the required root certificates of other domains into the container environment of a given domain or use "cross signed" certificates. Details about these two options are provided within the `CrossDomainCerts.txt` file located under the `docs` directory of the Provenance Store release.

4 Configuring the Query and the Analysis Tools

The Query engine is a stateless tool, hence its behaviour does not depend on previous query executions — but only on the configuration used when executed. This section describes the requested configuration data that such tools expect, outlining when and how the configuration of these tools needs to be undertaken.

4.1 The Query Engine

The Query engine is dedicated to retrieve information (p-assertions) located within one or more Provenance Stores. It provides an API to query multiple Provenance Stores with only one request.

Querying multiple Provenance Stores leads to the retrieval of several partial results. Before delivering these results to the application, the Query engine merges these into a single result. The merge process ensures that:

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

- all possible duplicated p-assertions are removed. Duplicated p-assertions could occur when querying two Provenance Stores where the first one is a backup of the second. The detection of duplicated p-assertions is done using the Comparator Tool also provided by WP6.
- the version of the PStruct schema is the same for all the p-assertions retrieved. This restriction is imposed by the definition of PStruct which can only contain p-assertions using the same version of PStruct, whereas querying multiple Provenance Stores could return several p-assertions using different schema versions.
- merge all partial results into one single PStruct, as if there was only one Provenance Store.

The final result can be manipulated later by other tools or external applications. In order to perform a query, the engine requires two configurable inputs: (1) a query, and (2) a list of Provenance Stores to which the query should be submitted. The list to be used can be specified by:

- *Provenance System Administrators*: as these individuals are responsible for managing the configuration process, they could provide the list of all Provenance Stores that are going to be available in a particular Provenance system.
- *Application Administrators*: these individuals are responsible of an application, that is why they have full access to the list of Provenance Stores used by this application in order to be able to create, modify or delete the locations of Provenance Stores from this list.
- *Application End Users*: by default, these individuals can only use the pre-defined list of Provenance Stores. However, the Application Administrator is able to authorize them to add new Provenance Stores. Authorizing an end user to add new Provenance Stores is achieved by adding this user to the *Personalized Provenance Store* security role. Note that the end user has full access to their own list of Provenance Stores, but they cannot modify the list specified by the Application Administrator.

Similarly, the set of available queries can be specified by different types of users:

- *Application Administrators*: as these individuals are responsible for an application, they have full access to the Query Engine for this application in order to be able to create, modify or delete queries from a list.
- *Application End Users*: by default, these individuals can only execute queries that are contained in a pre-defined list. However, the Application Administrator is able to authorize users to insert their own set of queries. Authorizing an end user to add new queries is achieved by adding this user to the *Personalized Query* security role. Note that the end user has full access to their own rules, but they cannot modify the rules specified by the Application Administrator.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

Template Queries

The "template queries" mechanism allows the creation of a query in which some parameters can be specified when the query is going to be executed. Thus, it is necessary to only define a template query once, and to use it several times in different application contexts — simply by modifying its parameters. Configuring the parameters of a template query is part of the configuration process.

This configuration can happen either during the application configuration (i.e when Application Administrators define the available query) or at runtime when using the query portlet within the eXo portal. Hence, if a template query is selected to be submitted, and the requested parameters have not been defined, the query portlet enters a special mode and waits for the definition of the query parameters by the user executing the query. The content of the query itself can be expressed using two formats: an XPath expression or an XQuery. An example of a template query is shown in example 1

Example 1 Example of queries for retrieving the content of a Provenance Store

```
(1) XQuery format :
       "<result>{for $n in ps:pstruct return $n}</result>"
(2) XPath format :
       "/ps:pstruct"

(3) Template query:
       "$query"
  where the parameters $query could be
       <result>{for $n in ps:pstruct return $n}</result> as in (1)
    or /ps:pstruct as in (2)
```

The Query portlet gives access to the Query engine through the eXo portal. The Query engine is able to analyse a query, and report if this query is a template or a standard query. For a template query it can also return parameters that need to be filled, or values of parameters that have already been filled. Thus, when a query is selected from a pre-defined list and executed, the Query portlet makes use of the Query engine to check if the given query is a template query or not. If yes, before the execution of the query, the portlet displays a form allowing the user to fill the parameters. Where parameters values have already been set – a default value or value from a previous execution, for instance – are shown.

4.2 The Analysis Engine

The Analysis tool encompasses several components to provide a reasoning framework over a set of p-assertions. The relationship between the components is shown in figure 3. These components are:

- The Java Expert System Shell (JESS),

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

- an `XMLLoader` module, a JESS extension written in Java that is in charge of loading the p-assertions exposed as XML data into the JESS memory,
- XMLTemplates are JESS templates that specify the mapping between an XML element and a JESS fact,
- a set of Helper functions, written in JESS, that help developers write reasoning rules by providing functions such as recursive deletion of an XML node when exposed as JESS facts, conversion between an XML element ID and its JESS facts, search function by element name, attributes, values, etc.
- a Conflict Detection module, written in JESS that validates the p-assertions. This module makes use of the Comparator Tool (another tool provided by WP6 and written in Java) that performs XML document comparison. The Comparator Tool makes use of the Documentation Style framework in order to compare two p-assertions using different documentation styles if the transformation from one documentation style to the other has been documented.

In addition, the Analysis Tool makes use of the Query Engine (section 4.1 to query Provenance Stores and retrieve p-assertions.

The core of the analysis tool is based on the Java Expert System Shell (`JESS`), a Java-based version of the CLIPS system. `JESS` uses the Rule Based approached to implement a Knowledge-Based (Expert) System and is more correctly classified as a Production Rule System. It uses an enhanced version of the `Rete` algorithm to process rules. `Rete` is an efficient pattern matching algorithm for implementing rule-based ("expert") systems. It is also an efficient mechanism for solving the difficult many-to-many matching problem (see, for example, [3]).

A production system is a computer program which consists of a set of rules, or productions which are of the form "IF *conditions* THEN *actions*", a database, also called working memory, which maintains state data, and a rule interpreter. The rule interpreter executes a forward chaining algorithm for updating and producing data. The condition portion of each rule (left-hand side or LHS) is tested against the current state of the working memory. If conditions are true, the consequent actions (right-hand side or RHS) are executed — updating, removing or adding data to the working memory.

The system stops processing either when the user interrupts the forward chaining loop; when a given number of cycles have been performed; when a "halt" RHS is executed, or when no rules have true LHSs. A Production Rule System's Inference Engine is stateful and able to enforce truthfulness - called Truth Maintenance. A logical relationship can be declared by actions, which means the action's state depends on the inference remaining true; when it is no

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

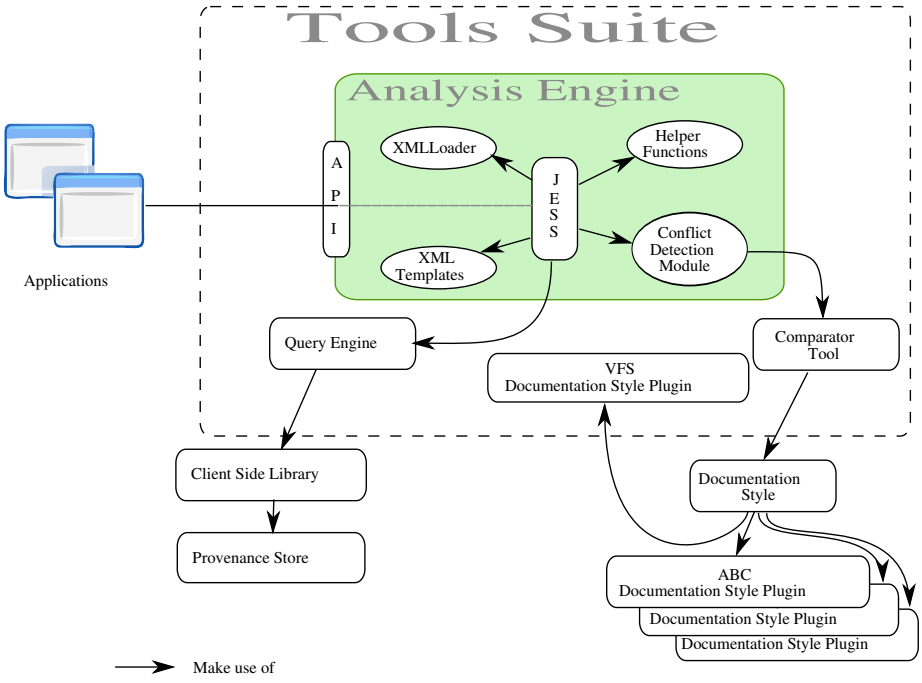

Figure 3: *Detailed view of the Analysis Engine*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

longer true the logical dependant action is undone. The "Honest Politician" is an illustrative example of Truth Maintenance, which always ensures that hope can only exist for a democracy while we have honest politicians. This example could be defined using the following rules:

1. when an honest Politician exists then logically assert Hope
2. when Hope exists then print "Hurrah!!! Democracy Lives"
3. when Hope does not exist then print "Democracy is Doomed"

The equivalent JESS rules are shown in listing 3.

```
(defrule rule1
   (honest ?Politician)
=>
   (assert Hope))

(defrule rule2
  (Hope ?)
=>
  (printout t "Hurrah!!! Democracy Lives"))

(defrule rule3
   (not (Hope ?))
=>
  (printout t "Democracy is Doomed"))
```

Listing 3: Jess example: Democracy

The rule interpreter, or inference engine, cycles through two steps: matching production rules against the database, followed by selecting which of the matched rules to apply and execute the selected actions.

Production systems may vary on the expressive power of conditions in production rules. Accordingly, the pattern matching algorithm which collects production rules with matched conditions may range from the naive – trying all rules in sequence, stopping at the first match – to the optimized, in which rules are "compiled" into a network of inter-related conditions. The latter is illustrated by the Rete algorithm.

To sum up, the Analysis Engine expects two different types of input: (1) a set of *rules* which represent the logic of the computation (also called *production rules*) and (2) a set of facts which represent the data to be analysed (also called *working memory*). As the rule engine is based on the JESS assertion engine, the reasoning rule must be provided either under the default Lisp-based encoding or under an equivalent XML-based format such as JESSML [15].

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

The data produced by the execution of a provenance-aware workflow is composed of a set of *p-assertions*. Such a set of p-assertions provides the description of the physical workflow. A p-assertion can be used to record one of the following events:

- an interaction between two actors,
- the state of an actor with respect to an interaction at a particular moment,
- a relation between two events.

These p-assertions are stored within a Provenance Store. The set of p-assertions to be analyzed is loaded into the analysis engine's memory using XML (the exposed format of the p-assertions) to facts (the format the analysis engine is able to cope with) bridge. This bridge has been developed in WP6 and can be used in a number of other contexts and JESS applications. In order to evaluate the performance of `XML` loading into the rule engine, some benchmarks have been performed to assess the overhead resulting from this rule loading process. They consist in measuring the time spent by the bridge to convert the XML data into facts. Obviously, the number of generated facts ties in with the length of the `XML` document to load. The computer used for these benchmarks is a laptop with Pentium M operating at 2.13GHz, 1GB of memory, a Linux fedora core 5 with kernel version 2.6.17. The hard disk is a Hitachi 5400RPM, ATA/100, with an access time of 12ms. The data we have used have been provided by the applications from WP7 and WP8.

The "loading time" curve in figure 4 presents the average duration of the populating process depending on the number of generated facts. The average time is computed after 100 consecutive loadings. The "file size" curve presents the size of the `XML` document depending on the number of generated facts. From the performance evaluation performed, we can observe that:

- The loading process has a linear complexity $O(n)$.
- The average loading rate is about 103,000 facts per second, or average accepted throughput is about 7.2MB per second.

This indicates that the rule loading process is scalable, and may be used for *reasonable* sized documents – the 103,000 facts (for instance) provides a representable value within the two applications being considered in this project.

A rule may be specified by different types of users:

- *Application Administrators*: as these individuals are responsible for the application, they can also specify the rules that can be run and that are meaningful in the application context.
- *Application End Users*: by default, these individuals can only execute rules available on the list of application rules set by the Application Administrator. However, the Application Administrator is able to authorize a user to insert their own set of rules by adding this user into the *Personalized Rules* security role.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

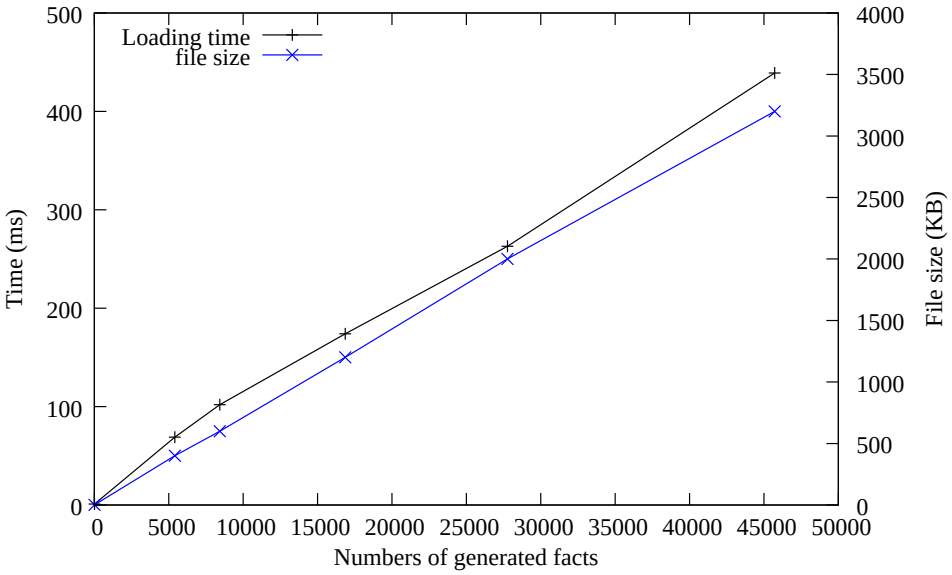

Figure 4: *Populating the engine memory*

4.2.1 Template Rules

The "template rule" mechanism provided by the rule engine allows individuals to create generic reasoning rules where some of the constraints can be specified externally at runtime. The definition of these constraint parameters is part of the configuration process.

Listing 4 shows how it is possible to change a standard rule into a rule making use of the template rule mechanism. The rule "normalRule" and "templateRule" only differ by one line – line 5 has been modified into line 13. In line 13, the use of the command `fetch` means that the value of this element is no longer fixed but varies according to the content of the variable "patientName".

```
1  (defrule normalRule
2    "Rule looking for a patient name,
3       name is specified in the rule"
4    (Element (ElementID ?elementID) (LocalName "Patient")
5          (Text "Durand"))
6    =>
7    ; ... some actions )
8
9  (defrule templateRule
10   "Rule looking for a patient name,
11    the name is contained within a variable called patientName"
12   (Element (ElementID ?elementID) (LocalName "Patient")
13     (Text (fetch "patientName")))
14   =>
```


The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

```
15      ; some actions )
```

Listing 4: Using template rules

The following Java example shows how to initiate a JESS engine instance (line 2), load a file containing the JESS rules defined in listing 4 (line 5), load a set of p-assertions into the JESS engine (lines 8-12), set the value of the variable "patientName to "Durand" (line 14) and launch the computation (line 16), then the value of "patientName" is set to "Dupont" and the computation is launched again.

```
1  // Jess Engine creation
2  JessAssertionEngine jess = new JessAssertionEngine();
3
4  // Loading of the rules (including the template one)
5  jess.batch(new File("/path/to/a/file.clp");
6
7  // Set of Passertions
8  Document passertions = ...
9
10 // Loading of the p-assertions
11 jess.loadXML(passertions);
12
13 // Filling the template parameters
14 jess.store("patientName", new Value("Durand",RU.STRING));
15
16 jess.executeCommand("(run)");
17
18 // Modifying the patient name only requires to set the new
19 // value, there is no need to write another rule.
20 jess.store("patientName", new Value("Dupont",RU.STRING));
21
22 jess.executeCommand("(run)");
```

4.2.2 P-assertion Conflict Detection

In addition to providing a reasoning framework within the project, the analysis tool is also used to detect possible conflicts in the p-assertions recorded. The nature of detected conflicts can vary, from detecting a difference between the data submitted by the sender and by the receiver for the same interaction (for example in the OTM application, if the result of a blood test as reported by the laboratory and the result available at a particular hospital differ) or the detection of unexpected behaviour during the execution of a workflow (for instance, an abnormally long duration between the extraction of an organ and the scheduled transplant operation).

Some benchmarks have been performed in order to evaluate the performance of the conflict detection module. The hardware and software environment is the

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

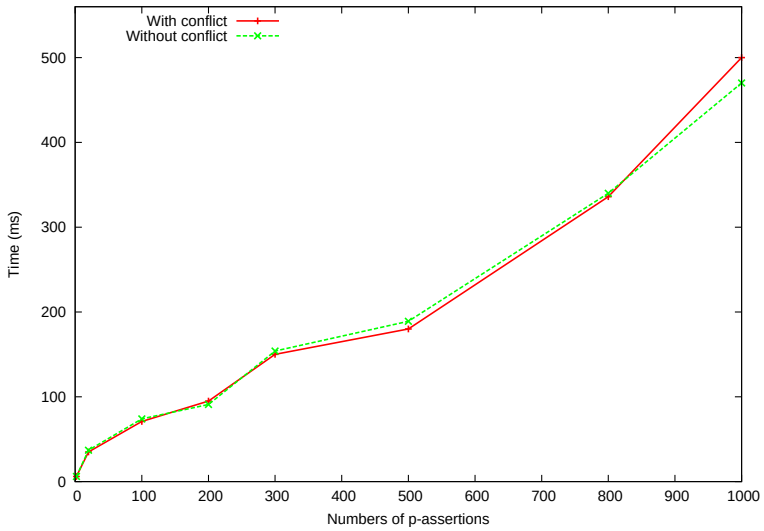

Figure 5: *Conflict Detection : Computation Time*

same as before. The results are shown in figures 5 and 6. Each value represents the average of 20 successive tests.

The performance of the conflict detection module has been tested on a set of p-assertions without conflict, and also by introducing one conflict in a randomly selected p-assertion. The introduction of the conflict has consisted in modifying one of the integer values exchanged by two actors in the sender view of the p-assertion. The results are shown in figure 5.

The memory usage has also been monitored and figure 6 shows the usage for a set of p-assertions when stored on the file system and the amount of memory used by its equivalent representation when loaded inside the analysis tool. Note that the amount of memory shown represents the amount of Heap memory used by the Java Virtual Machine executing the conflict detection code. The values have been collected through the Java Management eXtension (JMX) [14]. The maximum heap size of the JVM has been set to 512MB.

Figure 5 shows that the time spent on detecting a conflict is similar — with or without conflict in the input data. There is a small difference in the computation time when the amount of p-assertions increase. We consider this difference insignificant. As a possible explanation, it could be caused by the large amount of memory used, and the use of Java's garbage collection mechanism – which can be launched at irregular intervals for the Java Virtual Machine. The fact that the computer was not running only the performance code but also additional programs (contributing to background workfload) could also contribute to this value.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

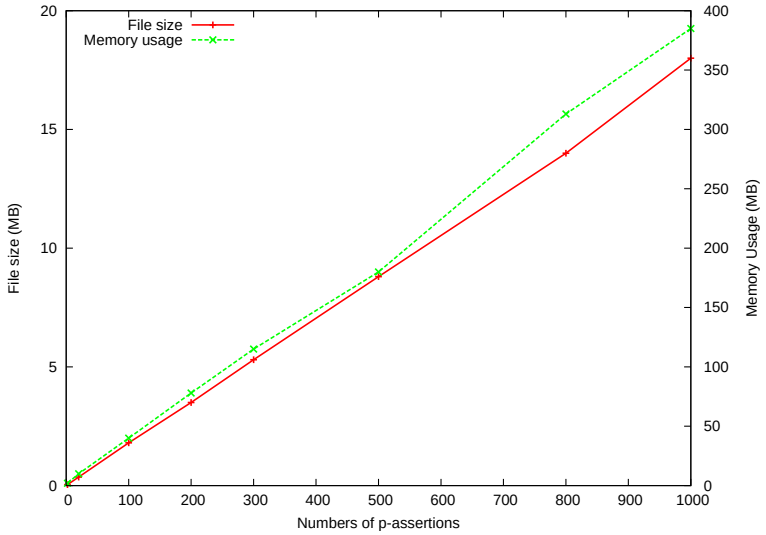

Figure 6: *Conflict Detection : Memory Usage*

Figure 5 shows that the memory used to represent a p-assertion remains constant as the two curves increase in a similar manner. From the data, it is possible to calculate the amount of memory used to represent a p-assertion. Given that a set of 1000 p-assertions use 18MB on the filesystem and its representation in memory uses 380MB, we can deduce that, on average, a p-assertion use 18KB on the hard disk and its representation in memory is 380KB. The ratio of memory representation divided by the file system space used is 21. This ratio is explained by the fact that each XML element is mapped into a Java object and that the data contained within one element rarely exceeds the size of 2 characters. In [8], the author demonstrates that a plain Java `Object` takes 8 bytes, an `int` uses 16-byte result, an empty `String` takes 40 bytes – enough memory to fit 20 Java characters. So an empty XML element with a 4 characters-long name uses: 2*4 characters + 2 * `"<"` + 2 * `">"` = 12 octets whereas its representation in memory uses at least: `Object` + 4 * `String`(element name, prefix, namespace, content) = 168 bytes. The ratio is here already 168/12 = 14.

Both of the figures present linear curves that indicates a complexity of $O(n)$.

4.3 Extended Queries

This mechanism was not initially planned to be provided inside the tool suite. Its inclusion results from a request of the OTM team (WP8) to be able to perform a verification (query of a Provenance Store + analysis of the result) as one single action to check on a particular process during an organ transplantation.

In order to provide an easy and complete integration of the tools within applications like Organ Transplant Management (OTM), the two separate compo-

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

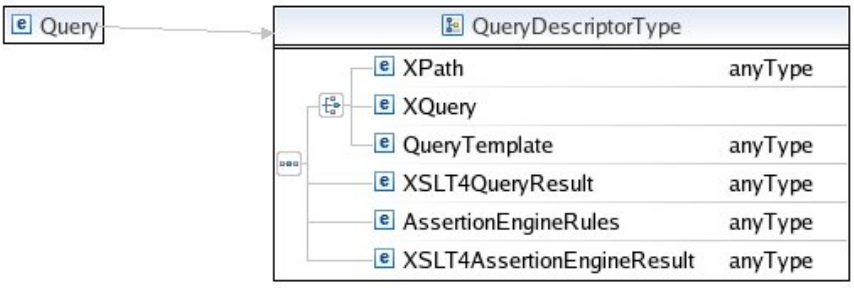

Figure 7: *Extended query*

nents presented above (the query tool and the analysis engine) can be combined in a all-in-one submission featuring also an XSLT based filtering and conversion mechanism between the query engine and the rule engine. XSLT, the Extensible Stylesheet Language Transformations, [4] is a language to transform the format of XML data into other XML documents or into data of other formats, on the basis of a set of well-defined rules. For example, XML files can be transformed into HTML pages, or into WAP Mark-up Language (WML) for display on Web-enabled mobile phones or in JPEG images.

An extended query is composed of four parts, as shown in figure 7:

1. a query that needs to be submitted to the Provenance Store. The query could be defined using XPath, XQuery or a template query
2. an XSLT document to filter/render the results of the query. The result of this rendering can be used to display the data retrieved from the Provenance Store,
3. a set of rules compatible with the analysis engine (JESS),
4. another XSLT document to filter/render the results of the analysis engine. This XSLT is used to render the results returned by the analysis engine into the format wanted. This could be another XML document, an HTML document, a PDF file, an image, etc.

Executing an extended query is only granted to those individuals who are allowed to access both the query engine and the analysis tool. This means that they have to belong, from the security point of view, either to the Query group and to the Analysis group or to any group which has access rights to both of these tools. In order to create an extended query, an individual must belong to the *Personalized Query* and *Personalized Rule* security role. Only the Application Administrator can grant these roles to a user.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

5 Configuring the Portal

5.1 User Interaction Types

In this section, a description of configuration to support user interaction with the portal server, various portlets, and configuration of groups and community-based access to portlets is provided. Each of the above is discussed below:

- User Registration: The aim of this operation is to configure access for new users to the eXo portal server. This would then allow new users to successfully authenticate with the portal using the given username/password pair. The list of new users to be added can be specified by:
  - *Provenance System Administrators*: as these individuals are responsible for managing the overall configuration process, they provide the list of all users who are going to take the role of Tool Suite Administrators and Application Administrators.
  - *Tool Suite Administrators*: as these individuals are responsible for managing the tools, they provide the list of all users that are going to be involved in maintenance, configuration and deployment of tools (e.g. navigation, analysis, etc).
  - *Application Administrators*: these individuals are responsible of specifying the list of application end users (for example doctors), allowing application end users to make use of the portal.
  - *Application End Users*: by default, these individuals have no rights to add new users.

  Figure 8 displays an abstract user `sergioalvarez` being added to the portal by a Provenance System Administrator, who belongs to an abstract organ transplant management(OTM) group and Application Administrators group. Discussion on group configuration is covered next. In figure 8 the banner and logo can be customized for each application.

- Group Membership: The aim of this configuration is to allow creation of groups and membership. This process allows the allocation of users to groups and giving them a membership type. This configuration restricts only certain users in a group to perform any edit on page content. For example, a user (doctor) in one of the OTM group can also be the head of a department and can thus be in a *department head group*. Department head group in this instance is an abstract group representing a hierarchial status in an organization (OTM application). The *"edit"* function on the page can be configured to restrict only members of department head group to perform any modification on the content relevant to this OTM group. This provides secure and efficient means to manage a portal configuration and layout.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

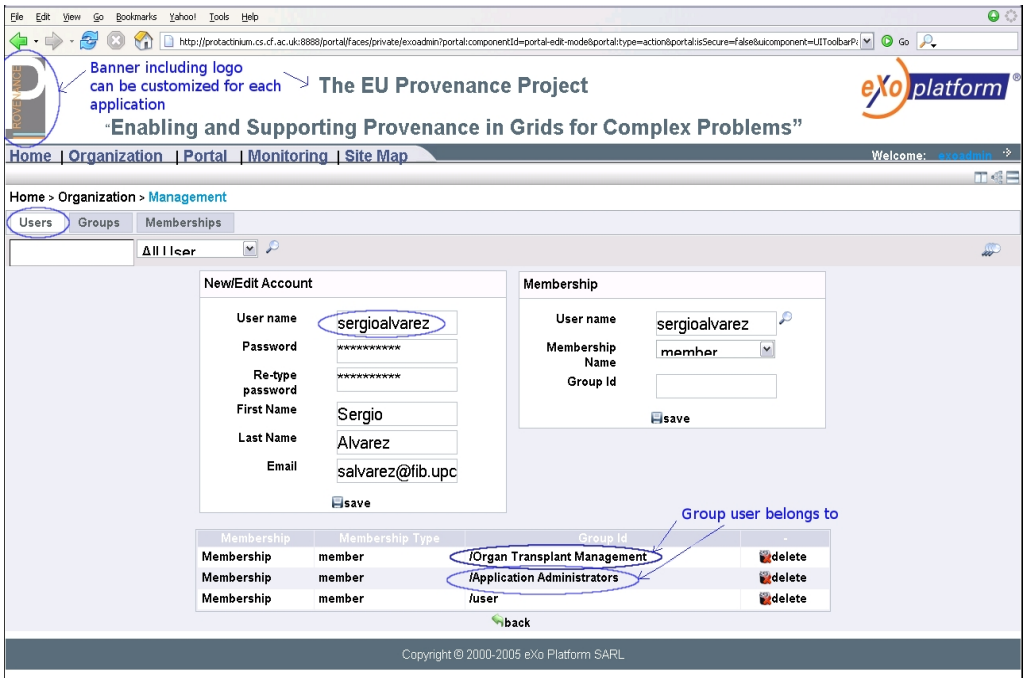

Figure 8: *Add new user*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

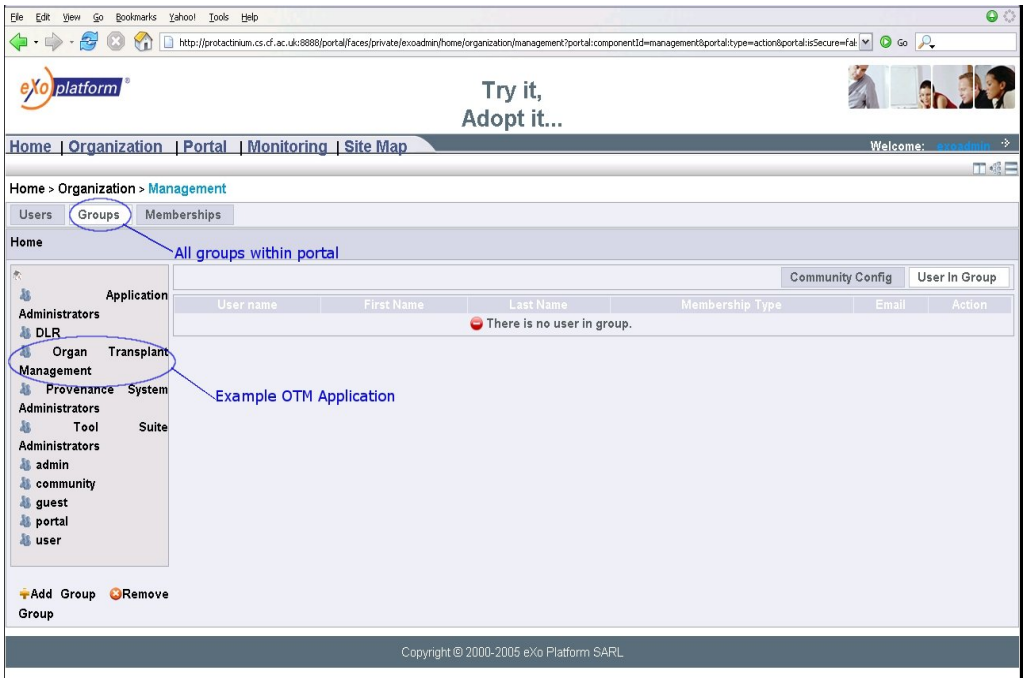

Figure 9: *Group Explorer*

This form of configuration eases the definition of layouts for all users that require similar content by grouping them together. Group configuration page is shown in figure 9. Groups can be specified by different types of users:

- *Provenance System Administrators*: as these individuals are responsible for managing the overall configuration process, they would be responsible for creation of Tool Suite Administrator and Application Administrator groups and assigning appropriate users to the respective groups.
- *Application Administrators*: these individuals are responsible for specifying application specific groups. For example, an abstract *department head group* for OTM application or *project manager* as identified by DLR application Section 1.
- *Application End Users*: by default, these individuals have no rights to add groups.

- Community Group: The aim of this configuration is to allow a group to inherit the portal and navigation properties from the community with which it has been associated. For example, users in department head group (OTM application) and users in Application Administrators group

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

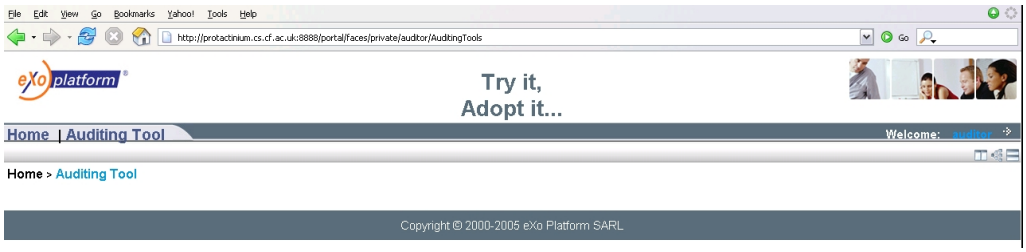

Figure 10: *Auditor Page*

requiring auditing on recorded p-assertions to ascertain if any anomalies exist. This can be achieved by either individual configuration of each users' portal page to include the navigation links, developing a portal page and required portlets, or by mapping the two groups into an *auditing community*. In this case an auditing community would consist of a portal page containing the analysis tool configured with auditing rules. By mapping to the auditing community, all users in the group will inherit the auditing pages and navigation links in their portal. The steps involved in this process are:

1. A template page is required for auditors. Groups can inherit pages from this template. Figure 10 displays the portal page for owner *auditor* which contains the template *"Auditor Tool page"*.
2. We select a single user (sergioalvarez) from OTM group, for demonstration. This group would be configured to inherit the *"Auditor Tool page"* and navigation links.
3. By default, community configuration for OTM group would be empty. Figure 11 displays the community configuration for OTM group without any reference to any community.
4. figure 12 displays the portal page for user (sergioalvarez) from OTM group. As seen from figure 12 the portal only contains the home page. By default, all users in portal have the most basic portal template, with only their home page content and navigation link made visible. Application Administrators or Application End Users can later configure the pages to better suit their individual needs.
5. In the next step, the community configuration for OTM group is set to include auditor community as seen from figure 13.
6. On revisiting the user (sergioalvarez) portal page (figure 14), it can be noticed that in addition to the home page the *"Auditor Tool page"* is also added.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

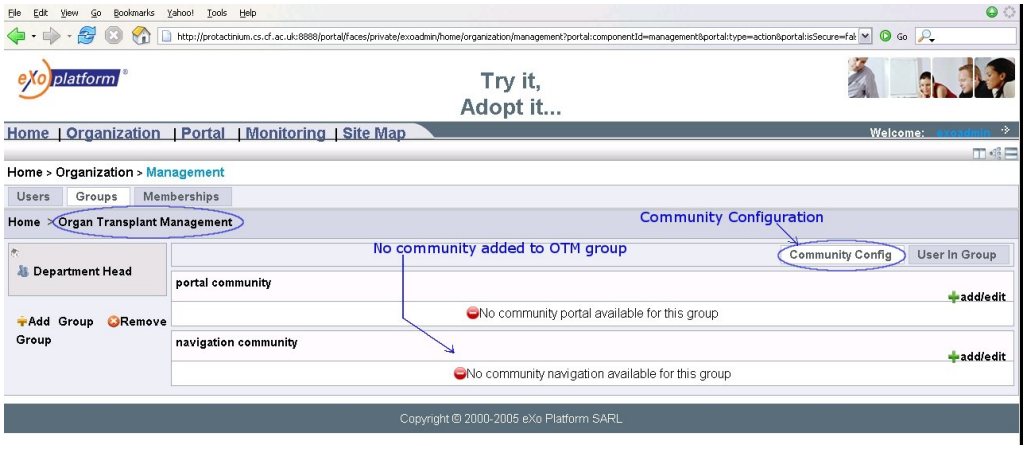

Figure 11: *Community Page without configuration*

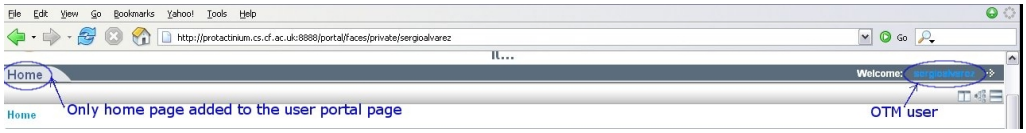

Figure 12: *OTM group user page without community configuration*

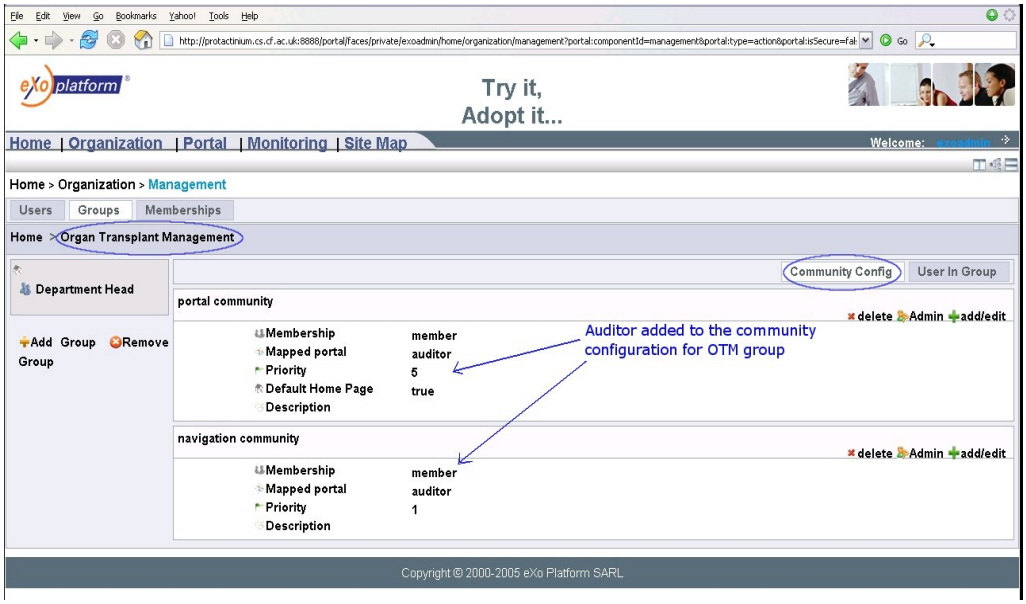

Figure 13: *Community Page with configuration*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

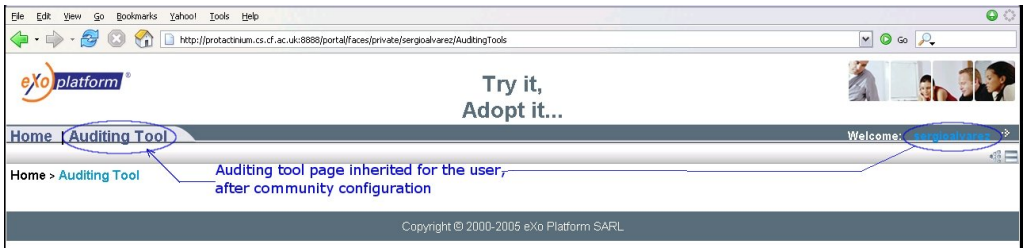

Figure 14: *OTM group user page with auditor community configuration*

5.2 Portal Management

Portal management involves identifying the operations that are necessary to configure Application End Users portal pages and perform backup of these configuration. Each of these is discussed below:

- Application End Users' portal page management: Application Administrators are often required to make changes to user and group portal pages. This would normally require a user to login first, perform necessary changes and logout. The same procedure would be carried out for each user. However the portal management configuration allows an administrator to login remotely to all user pages, perform the changes and come back to the configuration page. This provides a quick and efficient way to manage changes to portal pages. Figure 15 displays the portal management configuration page – as seen from figure 15 an *"edit"* icon is displayed next to each owner of the portal pages. On selecting the *"edit"* icon (for example, auditor) a "Welcome: auditor" page is displayed as seen from figure 15 making the auditor workspace now accessible for editing. Once the necessary edit is completed, pressing the back button next to the welcome message as seen in figure 15, will redirect the administrator to the original portal management page.

- Configuration file import/export: Important data such as user account information and portal customization configuration are stored as a compressed file in the portal. A compressed file is created for each user in the portal. Figure 16 displays the portal page to import/download compressed user data. Figure 16 also displays a snapshot of content present inside the configuration file of a user(vikasdeora). The import function allows a backup of the user data on the portal server, while the download allows the user data to be archived to a local store. The recovery of user data can be either performed using the portal, or by uploading a local copy of the user data as shown in figure 17.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

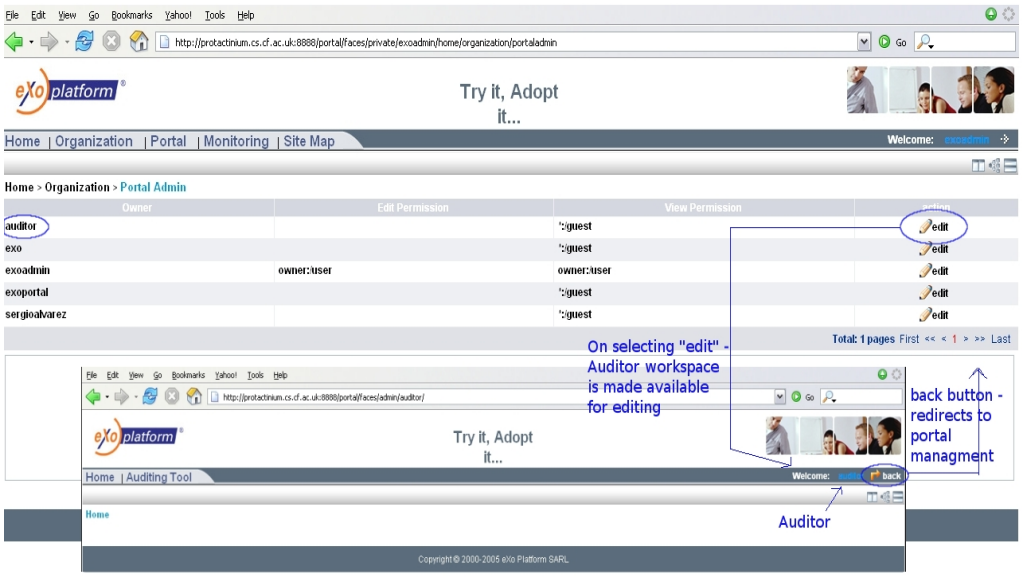

Figure 15: *Portal management configuration*

5.3 Views Supported

Various views are used in order to enable customization of portal pages for Application Administrators and Application End Users. This process also includes identifying the configuration necessary to restrict certain views (text, visualization) based on role based access. Each of these is discussed below:

- Customization mode: as seen in figure 18, on the upper right hand side is the customization configuration starting with edit portal, edit navigation, and edit page. These configurations allow the customization of the portal environment. The edit portal configuration allows customization of the complete portal layout, the edit navigation configuration allows customization of navigation links, and the edit page configuration allows customization of the content to be displayed on the portal page.

  As seen from figure 19, edit portal allows configuration of the overall layout of the portal. Edit portal configuration allows customization of banners, footer, and portlets within the portal. Since the edit portal configuration affects the layout for all users of the portal, access to this is restricted only to Application Administrators.

  The navigation configuration as seen in figure 20 allows configuration of navigation menu that references portal pages. This configuration allows new pages to be added and referenced with navigation links. Navigation configuration also allows categorization of contents, so that high priority pages appear first in the navigation menu.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

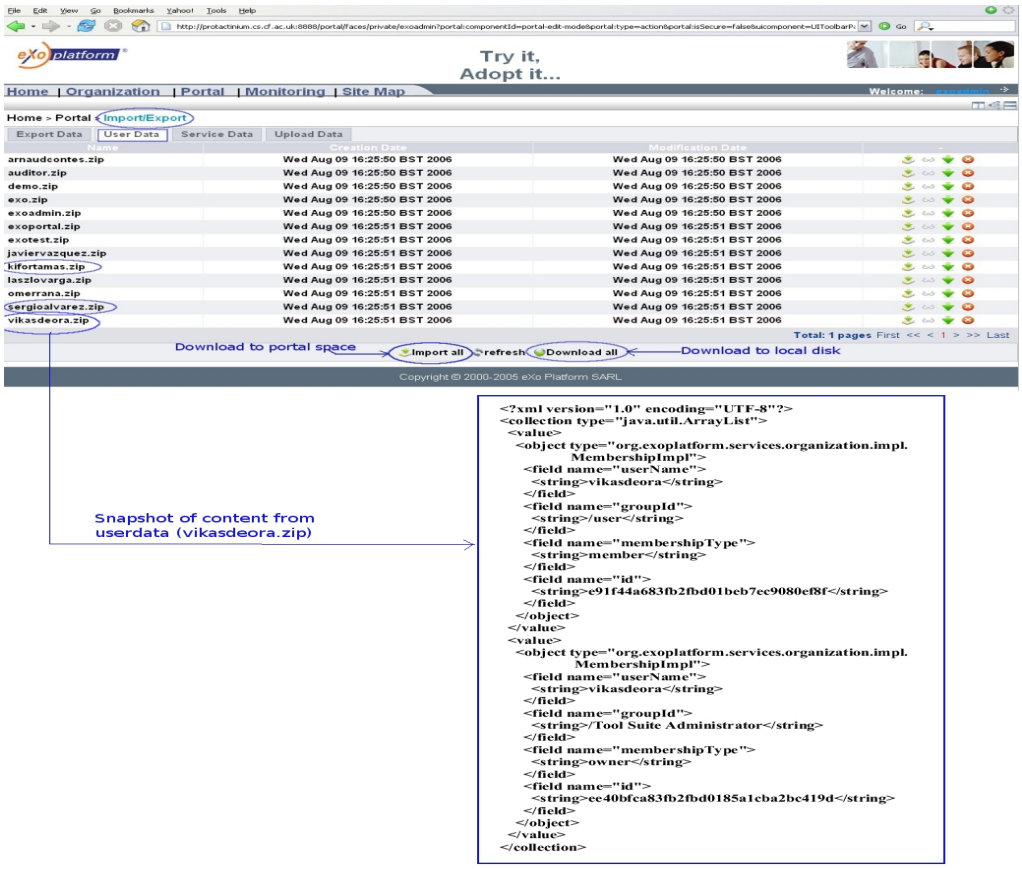

Figure 16: *Import User Data and sample content of User Data(vikasdeora)*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

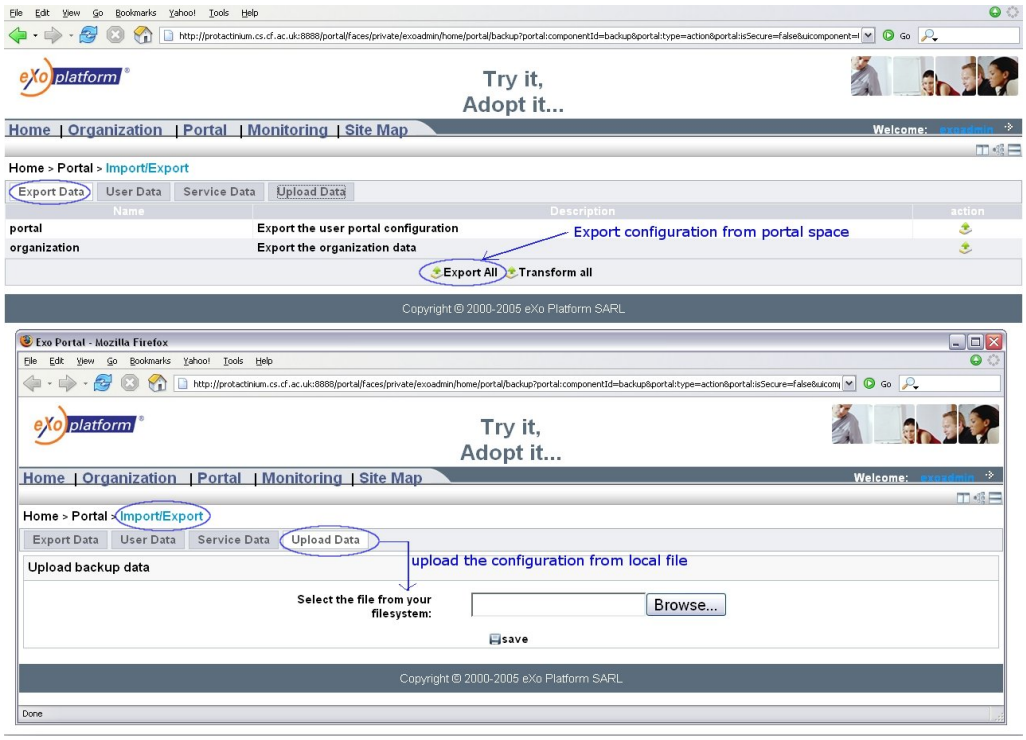

Figure 17: *Export User Data*

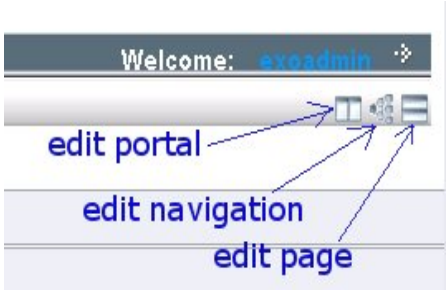

Figure 18: *Three configuration to allow customization of layout, navigation and content*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

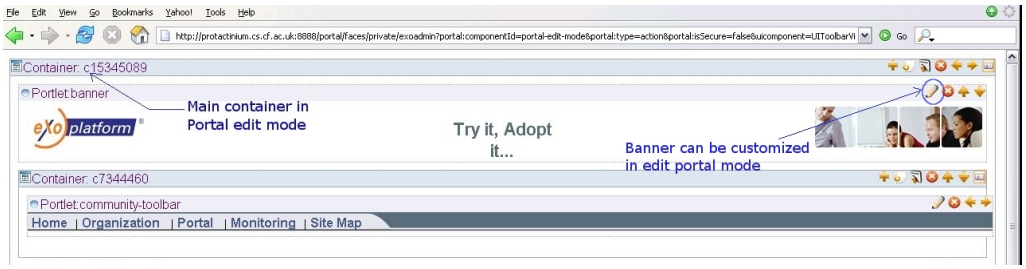

Figure 19: *Edit portal: Configuration of layout for main portal container*

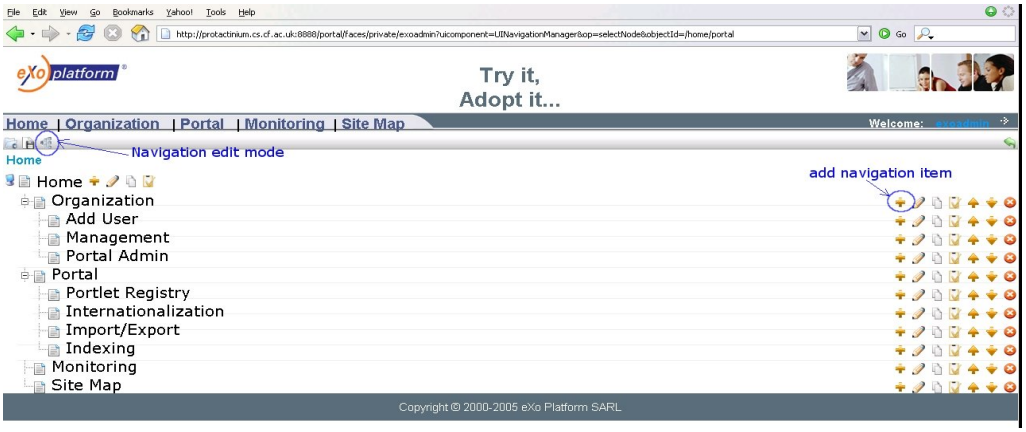

Figure 20: *Navigation Configuration*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

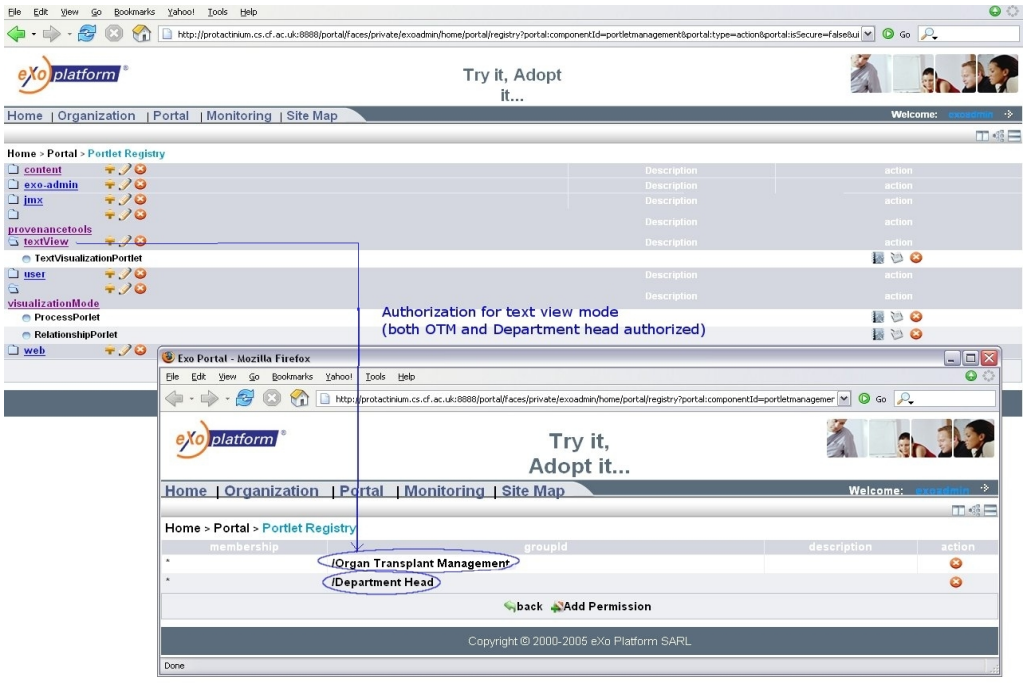

Figure 21: *Portal view configuration through portlet registry for text view portlet*

- Restricted view access: two main views are provided within the tool suite: the text navigation view and the visualization mode view (containing process view and the relationship view). Since Application End Users are allowed to manage their own content, they can add or edit content as required. However this needs to be restricted, so that only relevant content is displayed. In case of Application Administrators, Tool suite Administrators and Provenance System Administrators the portlet registry contains a list of all deployed portlets. This list is created by the portal automatically at startup time by adding portlets in "webapps folder".

  Application Administrators would be responsible for configuring the Portlet registry to allow role-based access to portlets. Figure 21 displays the text visualization portlet configured to be accessible by OTM and *department head* user and figure 22 displays access to the visualization portlet made available only to the *department head* user.

  Figure 23 displays a user (sergioalvarez) from OTM group adding a new portlet to his portal page. As seen from figure 23 only the text view portlet is exposed to this user, as visualization mode portlets were configured only to be accessible by the *department head*. Figure 24 displays a department head (javiervazquez) adding a new portlet, both text and visualization

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

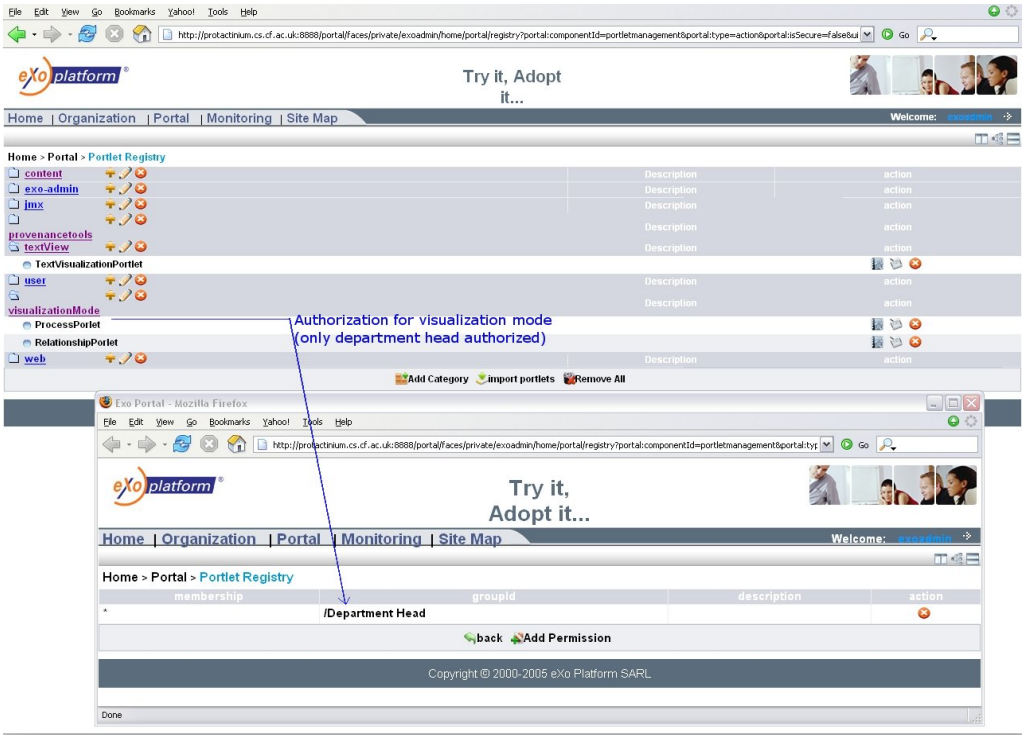

Figure 22: *Portal view configuration through portlet registry for visualization portlet*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

Figure 23: *Text only view*

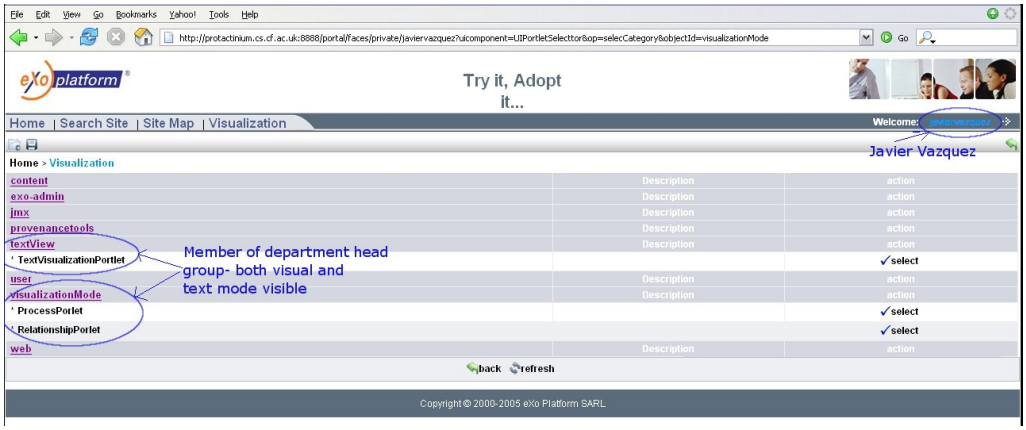

Figure 24: *visualization view*

mode portlets are available for this user to add.

5.4 Performance

The set of p-assertions to be visualized is first loaded into memory as a jgraph [7] model. The input p-assertions to the jgraph model are loaded as an XML (the native format of the p-assertion) document. The visualization component uses the jgraph model to render the visualization on screen. In order to evaluate the performance of rendering a visualization for any given set of p-assertions, some benchmarks have been performed — they consist of measuring the time spent by the visualization component to convert the p-assertions into the jgraph model and to subsequently complete the process of visualization/rendering on screen. The time taken to create and render the model ties in with the length of the `XML` document containing p-assertions. The computer used for these bench-

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

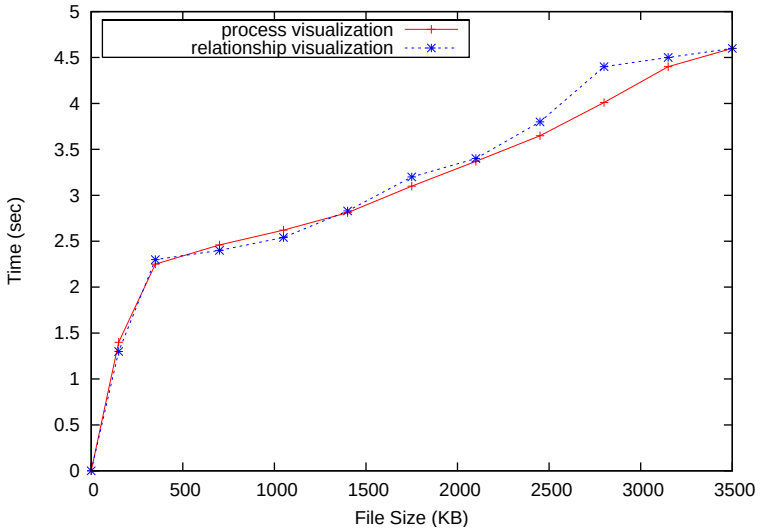

Figure 25: *Visualization Performance (File size)*

marks is a desktop with Pentium 4 operating at 2.23GHz, with 1GB of memory, running Windows XP with service pack 2. The hard disk is a Western Digital 7200RPM, ATA/100. The p-assertions we have used have been provided by the applications in WP7 and WP8.

The first curve (process visualization) in figure 25 presents the time taken by the process visualization portlet for rendering a process graph depending on the size of the `XML` document containing p-assertions. The second curve (relationship visualization) similarly presents the time taken by the relationship visualization portlet. It is important to note that the `XML` document containing p-assertions used for the experiment contains a random mixture of interaction and relationship p-assertions. Experiments to test performance in presence of only either of the p-assertions is displayed in figure 26.

The curves in figure 26 presents the time taken by the process and relationship visualization portlets for rendering a graph depending on the number of p-assertions. Note that only interaction p-assertions are used for the first curve (process visualization) and relationship p-assertions for second curve (relationship visualization).

From the performance evaluation performed in figure 25 and figure 26, we can observe that:

- Both of the figures present linear curves that indicates a complexity $O(n)$.
- The average visualization rate is about 1 MB of p-assertion data per second (figure 25).

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

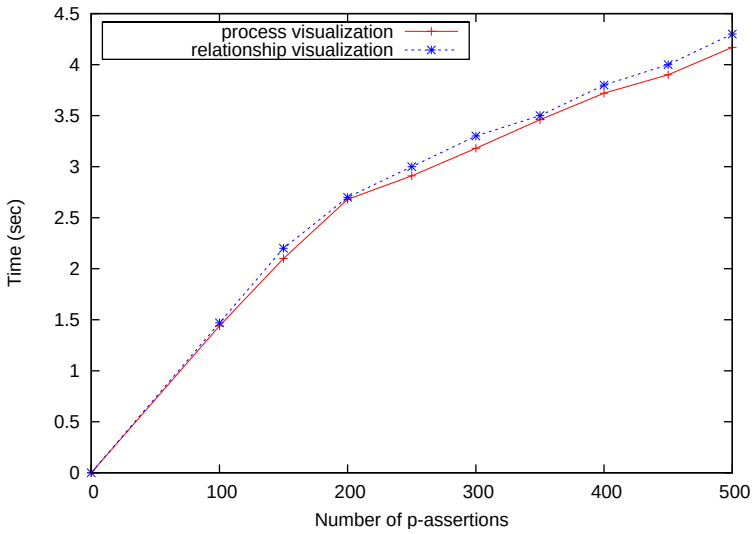

Figure 26: *Visualization Performance (Number of p-assertions)*

- The process and relationship visualization portlet have negligible difference in performance.

The curve in figure 27 presents the memory usage of the portal as new portlets are initialized. The memory usage represents the amount of Heap memory used by the Java Virtual Machine (JVM) initializing a portlet. The values have been collected through the Java Management eXtension (JMX) [14]. The maximum heap size of the JVM has been set to 512MB. Note that a higher than normal increase in memory usage was noticed during initialization of portlets for a very short period (one second), however this was ignored and only the stable figure was used in figure 27. One possible reason for this could be due to the fact that Java's garbage collector is triggered after initialization of portlets.

From the performance evaluation performed in figure 27, we can observe that:

- The memory usage has a linear complexity $O(n)$.
- The average memory used is about 4 MB per portlet.

5.5 Security Support

The Security framework [1, 2] needs to correctly interpret the role based access control (RBAC) configured using the portal. The configuration process involves synchronization of role information between the security framework and the portal, assuming a common semantics exist for the roles. The synchroniza-

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

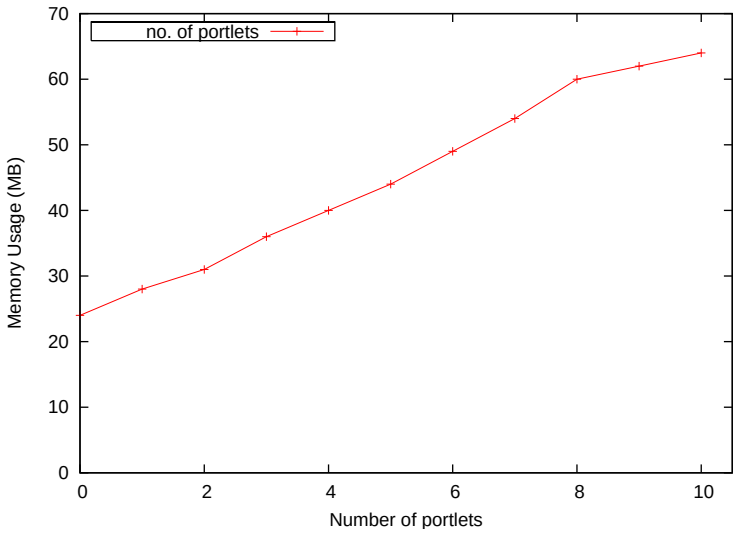

Figure 27: *Memory usage*

tion would be the responsibility of Application Administrators as the roles are application specific.

The process involves downloading the role information that needs to be synchronized as shown in figure 16 and transferring this information to the security framework. Its is assumed that the security framework can interpret the conveyed data and update the role information accordingly. It is also assumed that the security framework is active before the configuration process is initiated.

The aim of this configuration is to allow fine grained security support for query operations performed within the portlets. The security framework would be notified of the role information along with the certificate of a user performing any query on the Provenance Store. This configuration allows query results to be customized depending on the current role of the user.

6 Application Scenarios

Our visualization portlets allow p-assertions retrieved from a query, using the navigation tool, to be visualized as a graph. The visualization portlet displays two graphs: a process graph and a relationship graph that are based on interaction and relationship p-assertions respectively. Each of these, in the context of the EHCR, OTM and DLR application is discussed below:

- Process graph: By capturing all the interactions that take place between actors involved in the computation of some data, one can replay an execution, analyze it, verify its validity or compare it with another execution. A crucial element of an interaction p-assertion is information to identify

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

a message uniquely. Such information allows us to establish a flow of data between actors. Indeed, let us consider two interaction p-assertions: an actor A making an assertion α_A that it sent actor B a message with identity i, and actor B making an assertion α_B that it received from A a message with the same identity i. Such a pair of interaction p-assertions α_A, α_B is said to be "matching"; it identifies a flow of data from actor A to B.

The process of visualization is performed using the *process portlet* made available as part of the Navigation tool. Figure 28 displays a re-constructed process history using interaction p-assertions. In this case, the actors are represented as *boxes* and the *edges* represent the interactions between the actors. Multiple edges between two actors represent multiple interactions. Figure 29 displays the recreated process for EHCR application [10], figure 30 displays the reconstructed process for the OTM application [13] and figure 31 displays the recreated process for DLR application [5]. We describe the created graph based on EHCR application scenario. As can be seen from Figure 29, six actors are involved in the process. The "OTM:CollectPatientData", "OTMA3", "ehcrauth", "EHCR1", "EHCR2" and "EHCR3". Figure 29 displays all interaction between actors as part of collecting patient data. In EHCR graph (Figure 29) the data from the p-assertions that lead to the creation of the *boxes* and the *edges* is shown next to each actors. This data has been directly extracted from the submitted p-assertions.

The visualization can be configured using "widgets" made available on the top right hand corner (figure 28). "Widgets" are small containers which contain a collection of common set of operations to be performed on the visualization. Widgets allow Application Administrators to configure only certain operations to be active, in order to better suit an application context. For example, for an application that does not involve iterative processing of data, a widget to "handle iteration" might not be useful. In this case Application Administrators can disable this widget. The disabling of particular widgets is undertaken to reduce the complexity of the screen layout.

- Relationship graph: While matching interaction p-assertions denote a flow of data between actors, relationships explain how data flows inside actors. Relationship p-assertions are directional since they explain how some data was computed from other data. The relationship visualization is performed using the *relationship portlet* made available as part of the Navigation tool. Figure 32 and figure 33 displays the interaction relationship visualization for EHCR and OTM applications respectively. Figure 32 displays the relationship graph that illustrates the relationship between the interactions that took place as part of a process (figure 29). In this case, an interaction is represented as *boxes* and the *edges* represents the relationship between the interactions. The relationship graph helps a user

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

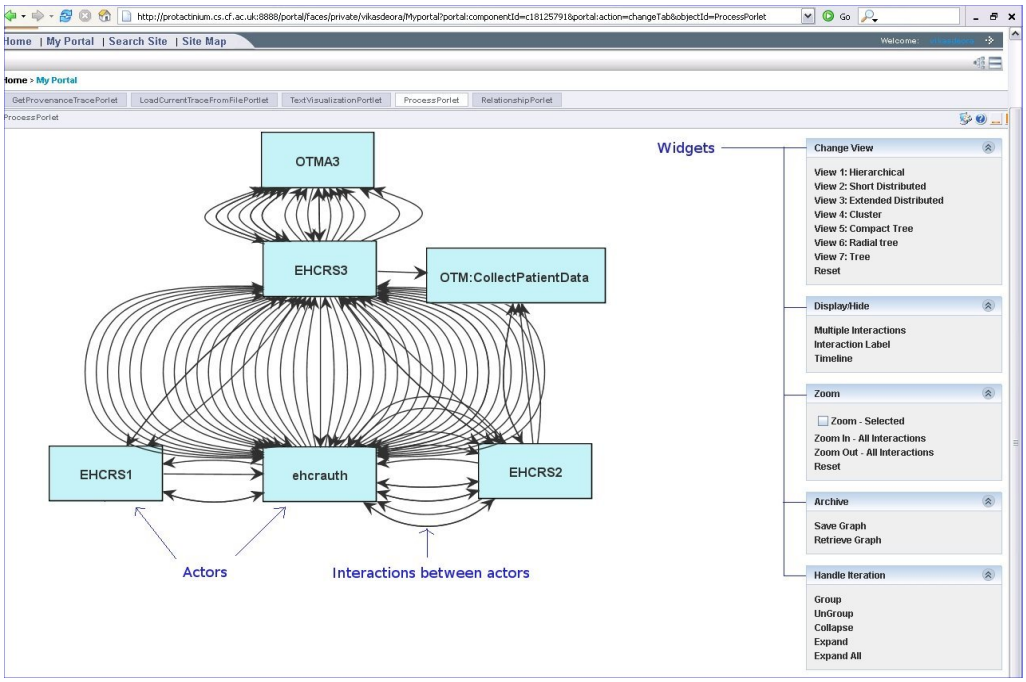

Figure 28: *EHCR Process with "widget" expanded*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

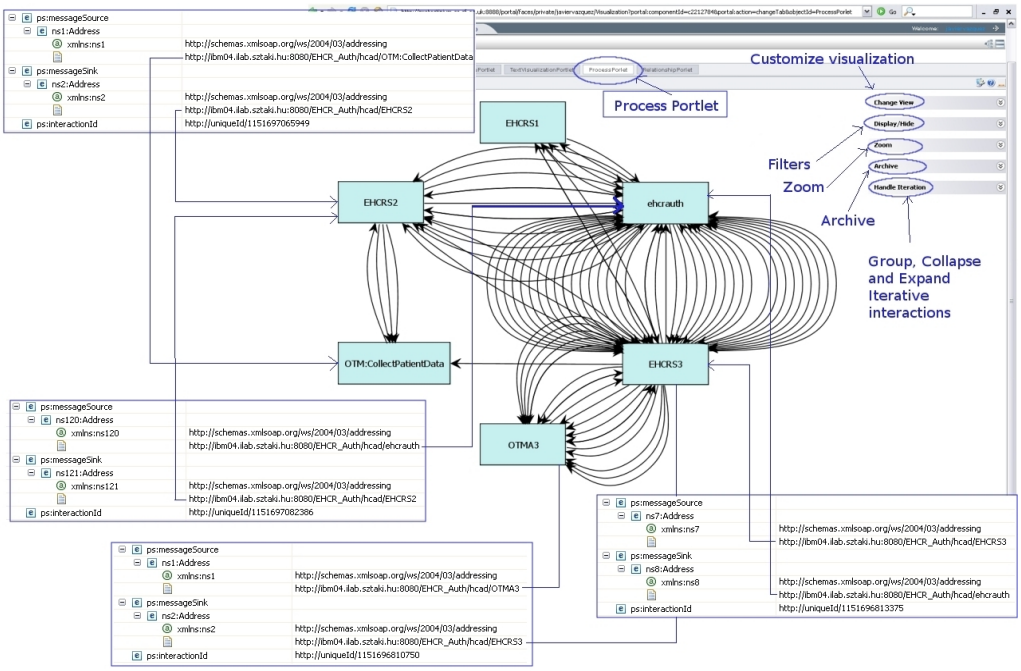

Figure 29: *EHCR Process*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

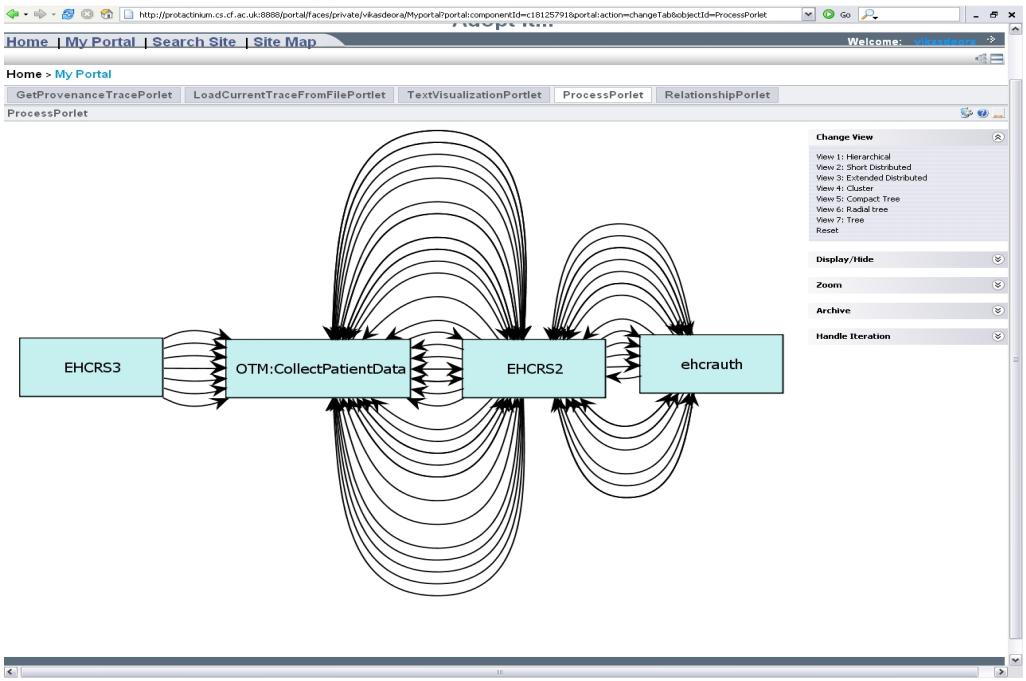

Figure 30: *OTM Process*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

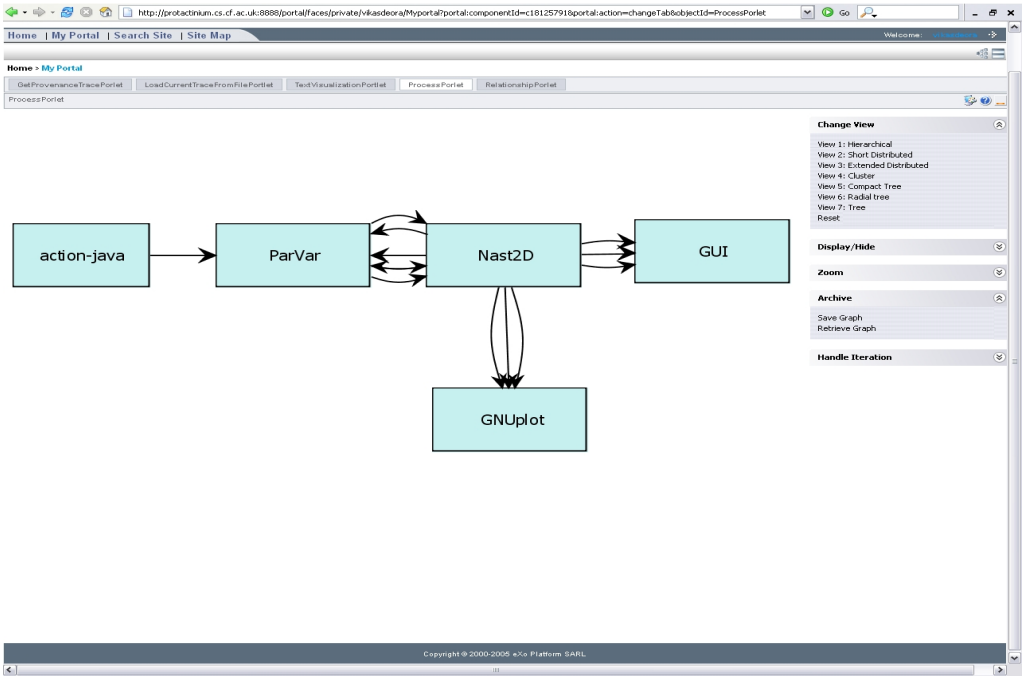

Figure 31: *DLR Process*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

understand *"Why"* and *"How"* an interaction happened. Relationship view also allows a user to analyze critical interactions.

For example, it can be observed from Figure 32 that some interactions are critical within the process, as they are the reason that many other interaction is caused by. This is important to observe, as a problem within such interaction can falsely lead to a chain of interaction that should not have been present. For example (in the OTM scenario - upon receipt of the "serology test", many other interactions are initiated to either start the operation and complete the organ transplant process or reject the donor, a mistake in this test, can cause a wrong set of interactions to be fired). In Figure 32, interactions 8347 and 8349 (local p-assertion id) are caused by 8345 (local p-assertion id). Making 8345 a critical interaction. A further investigation on such interactions can now be performed to trace how a treatment decision was made. Such analysis also helps in auditing; for example, to verify if all necessary interactions were performed before a decision was made. Other critical interactions include "caused by" which is based on many other interactions happening. For example, in OTM scenario, a decision request is caused by several interactions happening such as test, lab result, brain death, etc., visualization of the "caused by" relationship allows users to, for example, trace back how import decisions were made.

The visualization can again be configured using the "widgets" made available on the top right hand corner of the portal page. Figure 32 and figure 33 displays two different configuration for visualization of relationship. In figure 32 the user can visualize all interaction relationships as small thumbnails laid on top of the visualization panel. This allows a user to see the complete picture, while at the same time investigate critical interaction by pulling them to the empty region and zooming into subsets.

7 Non-Provenance Aware Application

In order to use provenance tools with applications that are not provenance-aware (i.e. do not make direct use of the Client Side Library), there are two possible approaches: to modify the application to be provenance aware, or to provide a set of tools which enable provenance data about applications to be stored. The former approach may be appropriate when the source code of the application is available and when the application runs on resources which are able to contact a Provenance Store. The latter approach is necessary if the application cannot be modified, and if it is unable to contact the Provenance Store.

The application considered here is the simulation of the collision of two black holes using a code implemented in the Cactus Framework [17, 18, 19]. The Cactus Framework is an open source, modular, highly portable, programming

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

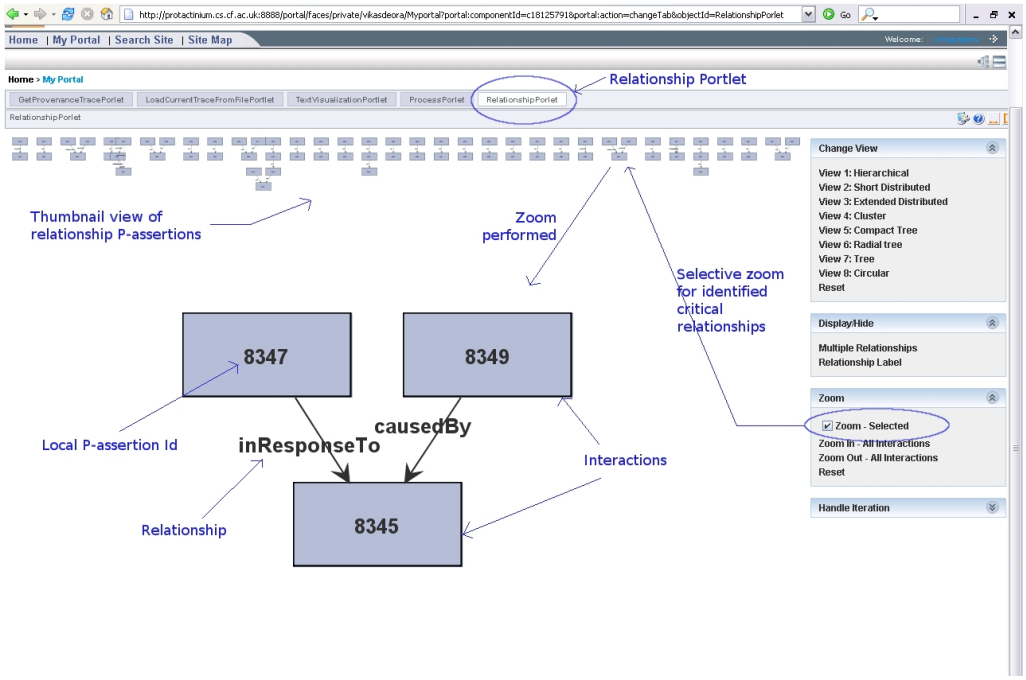

Figure 32: *EHCR Interaction Relationship configured with Thumbnail view*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

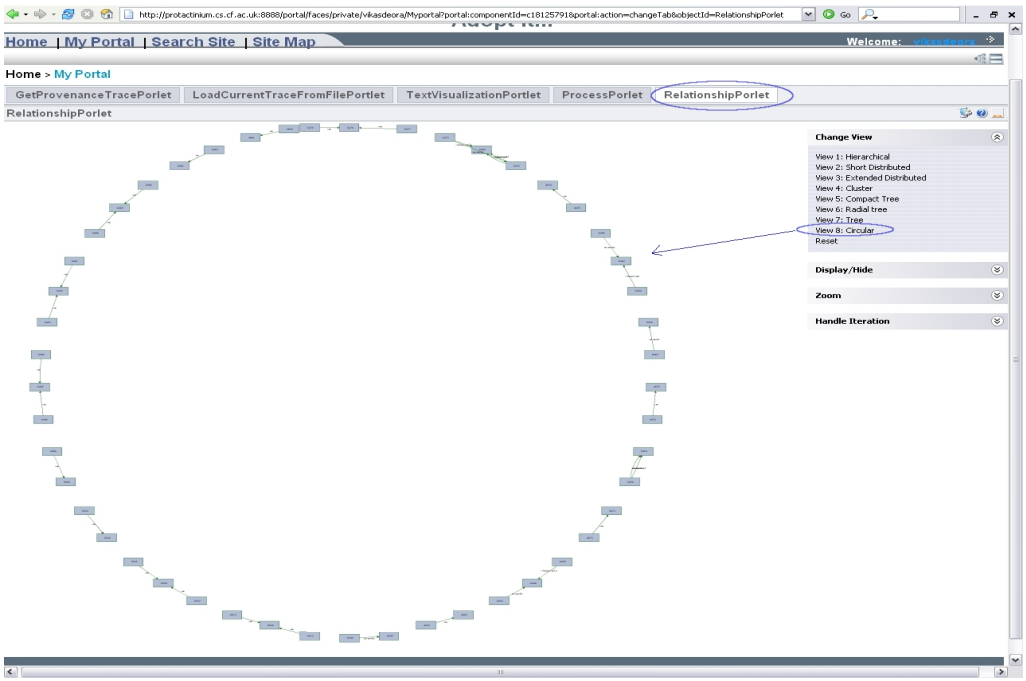

Figure 33: *OTM Interaction Relationship configured with thumbnail and circular view*

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

environment for collaborative HPC computing, now primarily developed at the Center for Computation & Technology at Louisiana State University. Cactus has a generic parallel computational toolkit with modules providing e.g. parallel drivers, coordinates, boundary conditions, elliptic solvers, interpolators, reduction operators, and efficient I/O in different data formats. Generic interfaces are used, (e.g. an abstract elliptic solver API) making it possible to develop improved modules which are immediately available to the broad user community. Cactus is used by numerous application communities internationally, including Numerical Relativity e.g. [20, 21], Climate Modelling [22], Astrophysics [23], Biological Computing [24] and Chemical Engineering [25]. It is a driving framework for a number of computing infrastructure projects, particularly in Grid Computing, including GrADS [26], GridLab [27], MicroGrid [28], GriKSL [29], and the ASC [23, 30].

The standard Cactus distribution contains modules for many common tasks in computational science, such as coordinate systems, physical and symmetry boundary conditions, parallelism and parallel I/O, along with standard toolkits for numerical relativity and a sample toolkit illustrating its use for the standard wave equation. Additionally, work is underway to develop toolkits for such fields as computational fluid dynamics, climate modeling and earthquake modeling. Cactus has many advanced features; e.g. parallel, platform-independent checkpoint/restart, dynamic parameter steering, portability across a plethora of architectures, an HTTP interface to allow remote users to monitor and steer their simulation using any web-browser, output in several widely deployed binary formats such as HDF5, and performance monitoring. Additional modules are available allowing the simulation meshes to be refined in selected regions to allow greater accuracy or resolution where required, i.e., adaptive mesh refinement.

Cactus users generally run large jobs, on a number of different machines — some users have accounts on upwards of seventeen different computer systems. Each of these jobs may generate its own initial data or use initial data generated by previous Cactus runs or by other codes, or from instrument readings. Tracking such runs is an increasingly difficult problem, both in locating the data and in recording the details of each run, so that it may be reproduced if necessary, and also to make manifest the processes leading to results published in papers or data passed to other applications. The Provenance tools created within this project provide a way to achieve these goals.

The specific application considered, the simulation of the collision of two black holes, is a standard test case which exemplifies the use of the code and the features of the code that are relevant to provenance. There are three possible approaches to using provenance with cactus:

1. As Cactus is a modular system, it would be possible to write a module which uses the Client Side Library (CSL) to store provenance information about the run. This approach was not taken as the CSL is a Java library and Cactus is often run on machines where Java is not available.

2. A module could be written which uses the Web Service interfaces to the

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

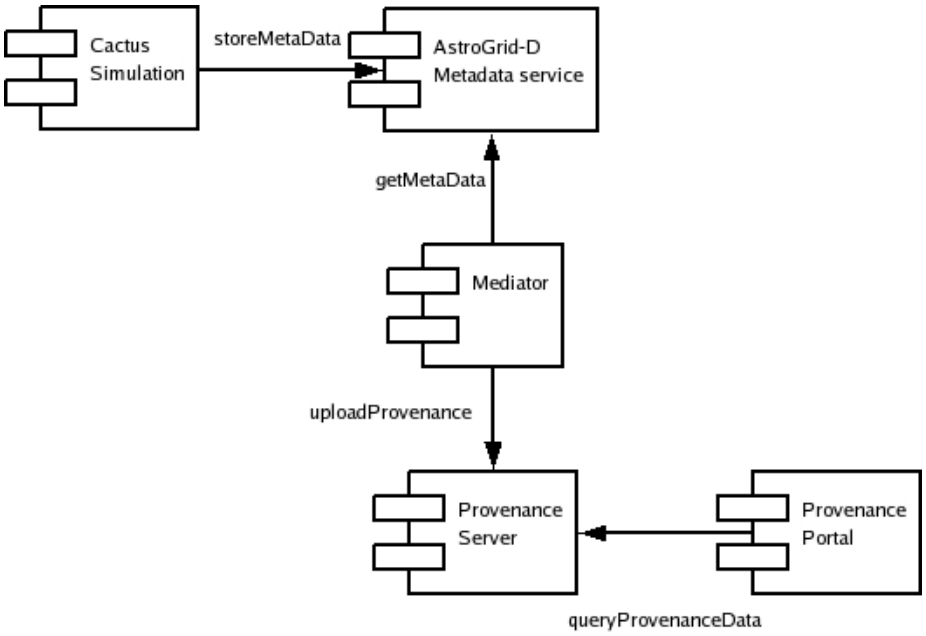

Figure 34: *Cactus, AstroGrid-D RDF store and Provenance architecture.*

Provenance Store to store the provenance information. This approach was not taken as many machines used for such computations are behind firewalls or NATs which prevent outgoing communication. Moreover, if the provenance store has security enabled, the lack of GSI on many such machines would prevent the use of the module.

3. The final approach is to develop tools using the CSL which can read in data provided by the simulation and any associated tools and upload these to the provenance server from the command line or from scripts. This was the approach taken. It also provides a general solution for non-provenance aware applications.

The most readily available data for a simulation is the configuration state of the simulation itself — i.e. the parameters with which it was configured and the location of input and output files. These can be encoded as actor state p-assertions and can be searched by the appropriate tools. For astrophysical simulations such data is readily available through the work of the German AstroGrid project [31] which has written a module to upload data about Cactus runs to an RDF store. This data is queried and processed into a set of p-assertions which are uploaded to the provenance store. This is illustrated in figure 34.

This is a very primitive use of the provenance infrastructure and does not

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

allow the full power of the provenance tools to be used. A more sophisticated approach is to record the interactions between different components in the simulation as separate p-assertions in the Provenance Store. An end user seeking to determine the provenance of a result obtained from the simulation then has a clear representation of the flow of data between different components and can use the provenance tools described earlier to analyze the data flow.

In order to facilitate this a tool has been written which takes the schedule from a cactus run and converts it into a set of p-assertions which can be uploaded by the standalone tool mentioned above. The combination of these tools allows a user to store the complete workflow of the application: data source, internal cactus workflow and state, and location of resulting data.

Using these tools it is possible to determine which Cactus runs were started using a particular data set, and the chain of previous runs which lead to a final data set.

8 Conclusion

This document has discussed the types configurations necessary within a general provenance system, and subsequently within the tool suite. Configuration management issues are discussed – such as the role of a user undertaking the configuration, the period over which the configuration is undertaken, the type of configuration being undertaken, and the mechanism used to support the configuration.

Some configuration needs to be supported during system setup time – such as the location of the Provenance Store or the tool suite etc – whereas other forms can progress during p-assertion submission or access. The relationship between the setup protocol and configuration management has also been discussed in section 3.

The configuration mechanisms employed within the analysis engine and the navigation tool are discussed in particular. The analysis engine involves:

- A description of the configuration of queries that can be made to retrieve p-assertions from a Provenance Store.
- The configuration of the rule engine, and subsequently the rules required to undertake analysis on the p-assertions that have been retrieved.
- Configuration support for integrating the rule engine and the query mechanism for supporting “extended queries”.

Performance analysis for various types of configuration options within the analysis engine have also been presented. The navigation tool configuration involves:

- Identifying user registration and group access to the portal server.
- Identifying the types of portlets a particular user should be allowed to access.

The *PROVENANCE* Project receives research funding from the European Commission’s Sixth Framework Programme

- Identifying the types of "views" that should be supported on the retrieved p-assertions (from a user query). This involves identifying particular options that an application end user has for modifying how the returned p-assertions are displayed.

Performance analysis for various types of configuration options within the navigation tool have also been presented.

Both the analysis engine and the navigation tool configuration has then been demonstrated in the context of the OTM/EHCR and DLR applications – based on p-assertions obtained from them. Finally, the use of configuration management approaches within a non-provenance aware application (i.e. one that does not make direct use of a Client Side Library) is also presented. This demonstrates how the tool suite can be used alongside applications that require use of provenance tools, but which have not been modified to work with the Client Side Library. The limitations of this approach have also been discussed in section 7.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

Note on References: Some of the references are restricted documents and only accessible to participants in the EU Provenance project.

References

[1] Liming Chen, Paul Groth, Simon Miles, Victor Tan, Fenglian Xu, Luc Moreau, "Security Architecture Strawman", from EU-Provenance project – WP4, May 2005.

[2] John Ibbotson, Victor Tan, "Towards implementing a Security Architecture for Provenance Systems", from EU-Provenance project – WP4, Feb 2006.

[3] C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem". *Artificial Intelligence*, 19:17–37, 1982.

[4] James Clark, "XSL Transformations (XSLT) Version 1.0", Available at: `http://www.w3.org/TR/xslt`, 1999.

[5] Frank Dannemann, "Outline Aerospace Scenario", from EU-Provenance project – WP7, January 2005.

[6] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. "HTTP extensions for distributed authoring – WEBDAV", 1999.

[7] JGraph. Available at: `http://www.jgraph.com/`, January 2006.

[8] Vladimir Roubtsov, "Do you know your data size?". Available at: `http://www.javaworld.com/javaworld/javatips/jw-javatip130.html`.

[9] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. "Role-Based Access Control Models". *Computer*, 29(2):38–47, February 1996.

[10] Kifor Tams, "EHCR store", from EU-Provenance project – WP8, July 2005.

[11] The Apache Software Foundation, "Commons Virtual File System". Available at: `http://jakarta.apache.org/commons/vfs/`, August 2006.

[12] The Apache Software Foundation, "Jakarta Commons". Available at: http://jakarta.apache.org/commons/, August 2006.

[13] Javier Vazquez, " OTM application system work/data flow", from EU-Provenance project – WP8, June 2005.

[14] Sun Microsystems, "Java Memory eXtensions (JMX)". Available at: `http://java.sun.com/products/JavaManagement/`, August 2006.

[15] JESSML. Available at: `http://www.jessrules.com/jess/docs/70/xml.html`, August 2006.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

[16] M. Ripeanu, A. Iamnitchi, and I. Foster. Performance predictions for a numerical relativity package in Grid environments. *International Journal of High Performance Computing Applications*, 15(4):375–387, November 2001. http://people.cs.uchicago.edu/~matei/PAPERS/ijsa.ps.

[17] Cactus computational toolkit. http://www.cactuscode.org.

[18] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf. The Cactus framework and toolkit: Design and applications. In *Vector and Parallel Processing - VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer. http://www.cactuscode.org/Papers/VecPar_2002.pdf.

[19] G. Allen, T. Goodale, G. Lanfermann, T. Radke, D. Rideout, and J. Thornburg. *Cactus Users Guide*, 2004. http://www.cactuscode.org/Guides/Stable/UsersGuide/UsersGuideStable.pdf.

[20] Apples with apples: Numerical relativity comparisons and tests. http://www.AppleswithApples.org.

[21] EU Astrophysics Network home page. http://www.eu-network.org/.

[22] B. Talbot, S. Zhou, and G. Higgins. Review of the Cactus framework: Software engineering support of the third round of scientific grand challenge investigations, task 4 report - earth system modeling framework survey. http://sdcd.gsfc.nasa.gov/ESS/esmf_tasc/Files/Cactus_b.html.

[23] Astrophysics Simulation Collaboratory (ASC) home page. http://www.ascportal.org.

[24] Illinois BioGrid. http://www.illinoisbiogrid.org/.

[25] K. Camarda, Y. He, and K. A. Bishop. A parallel chemical reactor simulation using Cactus. In *Proceedings of Linux Clusters: The HPC Revolution, NCSA*, 2001. http://www.cactuscode.org/Papers/Camarda01.doc.

[26] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf. The Cactus Worm: Experiments with dynamic resource discovery and allocation in a grid environment. *Int. J. of High Performance Computing Applications*, 15(4), 2001. http://www.cactuscode.org/Papers/IJSA_2001.pdf.

[27] GridLab: A Grid application toolkit and testbed project home page: http://www.gridlab.org.

[28] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. The MicroGrid: a scientific tool for modeling computational grids. In *Proceedings of SC2000*, Dallas, TX, 2000. http://www.sc2000.org/techpapr/papers/pap.pap286.pdf.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme

[29] DFN-Verein project *"Development of grid based simulation and visualization techniques"* (GRIKSL) home page. `http://www.griksl.org`.

[30] R. Bondarescu, G. Allen, G. Daues, I. Kelley, M. Russell, E. Seidel, J. Shalf, and M. Tobias. The Astrophysics Simulation Collaboratory portal: a framework for effective distributed research. *Future Generation Computer Systems*, 2003. Accepted. `http://zeus.ncsa.uiuc.edu/~ruxandra/asc.pdf`.

[31] AstroGrid-D project `http://www.gac-grid.de`.

The *PROVENANCE* Project receives research funding from the European Commission's Sixth Framework Programme