



*Title:*                *Tools Description Document*

*Author:*            *Work Package 6 (“Tools and Setup”)*

*Editor:*             *Omer F. Rana (UWC)*

*Reviewers:*        *Luc Moreau (SOTON), Simon Miles (SOTON), Steven Willmott (UPC)*

*Identifier:*         *D6.1.1*

*Type:*                *Deliverable*

*Version:*            *1.0*

*Date:*                *20<sup>th</sup> September 2005*

*Status:*              *Restricted*

### **Summary**

This document contains a description of different software libraries (“tools”) that may be used to access a provenance store. Such tools make up an integral part of a Provenance Architecture, and enable an application to make queries, submit data and manage one or more provenance stores. Relationships to the Software Requirements Document (D2.1.2) and the Logical Architecture are also provided.

**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 s.V.	Germany
Universitat Politècnica de Catalunya	Spain
Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutató Intézet	Hungary

## **Foreword**

This document has been compiled by Omer Rana (UWC). Other contributors to the document include:

- John Ibbotson and Alexis Biller:
- Luc Moreau, Paul Groth and Simon Miles:
- Arnaud Contes, Vikas Deora, Ian Wootten:

# Table of Contents

<b>Foreword.....</b>	<b>3</b>
<b>Table of Contents.....</b>	<b>4</b>
<b>Table of illustrations.....</b>	<b>5</b>
<b>List of acronyms.....</b>	<b>6</b>
<b>List of definitions.....</b>	<b>7</b>
<b>1 Introduction.....</b>	<b>8</b>
<i>1.1 Purpose of the document.....</i>	<i>8</i>
<i>1.3 Overview of the document.....</i>	<i>8</i>
<b>2 Tool Requirements .....</b>	<b>9</b>
<i>2.1 Relationship to Software Requirements Document .....</i>	<i>9</i>
<i>2.4 Relationship to Logical Architecture .....</i>	<i>11</i>
<i>2.6 Types of Tools and Scope.....</i>	<i>12</i>
<i>2.8 General Constraints and Assumptions.....</i>	<i>14</i>
<b>3 Tool Suite Description and Implementation.....</b>	<b>15</b>
<i>3.1 Navigation.....</i>	<i>15</i>
<i>3.4 Relationship Definitions.....</i>	<i>15</i>
<i>3.5 Analysis.....</i>	<i>17</i>
<i>3.4.1 p-assertionAM : Provenance Assertion Abstraction Mechanism.....</i>	<i>17</i>
<i>3.6.1 Assertion Engine .....</i>	<i>18</i>
<i>Bindings.....</i>	<i>18</i>
<i>3.9 Comparison .....</i>	<i>19</i>
<i>3.11 Conflict.....</i>	<i>19</i>
<i>3.13 Temporal .....</i>	<i>19</i>
<i>3.16 Graphical Interface and Portlets .....</i>	<i>19</i>
.....	<i>20</i>
<b>4 Tools and Provenance Store Interaction .....</b>	<b>22</b>
<b>5 Tools and Application Interaction.....</b>	<b>23</b>
<b>6 Examples of Usage.....</b>	<b>27</b>
<b>References.....</b>	<b>29</b>

## Table of illustrations

### Figures:

1. High level interaction between Tool Suite, provenance store and Application.
2. Comparison Plug-in Architecture.
3. Assertion combining User and Provenance defined data.
4. UML Class diagram of the Tool Suite.
5. UML Sequence diagram of the Tool Suite.
6. Tool Interaction.

## **List of acronyms**

## List of definitions

- *Actor*: An individual or an organisation that is involved in a data manipulation process.
- The *provenance of a piece of data* is the process that produced that data.
- A *p-assertion* is an assertion that is made by an actor and pertains to a process.
- *Process Documentation*: The documentation of a process consists of a set of p-assertions made by the actors involved in the process.
- *Workflow*: The process by which a series of tasks are executed in a specific sequence; including the specification of how outputs of tasks are routed to the inputs of other tasks or stored, whichever action is required.
- *Workflow enactment engine*: A software program that conducts the execution of a workflow in accordance with the specification of the workflow. In distributed computational environments the workflow enactment engine is usually a service that makes use of and coordinates other services in order to execute a given workflow submitted to the engine by a client.
- *Validation Assertion*: An atomic query that must evaluate to either “True” or “False” based on p-assertions within one or more provenance stores. Validation Assertions can be chained together using *AND* or *OR* operators.
- *Provenance Trace*: A collection of p-assertions that have been retrieved as a result of a query.

# 1 Introduction

WP6 is aiming to produce tools for navigating, accessing and reasoning over process documentation placed in the provenance store(s). Such tools are intended to be “generic”, i.e. application independent, and would interact with the provenance store using a Management, Query and Submission interface provided by the provenance store. This functionality provided by the tool suite provides extended capabilities to the application user, allowing ease of query and analysis over process documentation which is currently not possible with the use of provenance store Application Programming Interface (API). The provenance store API (in the Logical Architecture [WP3]) may be too low level for applications to make use of conveniently. It is therefore intended that the functionality provided by the tool suite provides a more useful, higher level set of capabilities.

A key deliverable from this WP will be the “analysis” tool, and the assertion checking mechanism that it supports. The remaining tools will make use of this mechanism. An “interface” in this instance is assumed to be an API, that may be made use of by an external program. As a simple definition, it is assumed that a provenance store contains a collection of p-assertions, that follow some pre-defined schema. The recording of these records and their physical structure in the provenance store is not made public through the management or query interfaces, however the scheme of each p-assertion is made public via the Query or Management API. A schema contains a set of elements that represent the structure of the p-assertion, and each of the element contains one or more data items associated with it.

## 1.1 *Purpose of the document*

The purpose of this document is to describe various components and the communication between the components constituting the WP6 package. A UML class and sequence diagram is also provided to illustrate the above.

Further sections provide details about each of the tools forming part of the WP6 tool suite.

## 1.3 *Overview of the document*

This document provides a description of the tools that access data in the provenance store. Tools in this case constitute a set of “helper classes” that can be accessed via an application program. A minimal graphical interface is also discussed which makes use of these helper classes to enable a user to view the contents of the provenance store. Section 2 describes relationships between the tools, the types of tools that are being supported, and assumptions being made about the architecture within which the tools are being deployed. Section 3 then describes the tool suite in more detail, with specific information about each tool. Interaction between the tools and the provenance store are described in Section 4, and between the tools and the application in Section 5. Some simple examples of use are provided in Section 6. It would be useful to note that Sections 1 and 2 describe the design of the tool suite, whereas Section 3 is focused on implementation aspects of it.

## 2 Tool Requirements

Three types of tools are envisioned as an outcome of WP6:

1. Tools for accessing and analysing the contents of the provenance store. Such tools are expected to provide functionality beyond that available within the Management and Query interface made available at the provenance store. In the simplest case, such functionality may aggregate capability available within the Management and Query interfaces.
2. Tools for submitting information to the provenance store. Such tools are expected to possess capability beyond that available within the Submission interface made available at the provenance store. In the simplest case, such functionality may aggregate capability available within the Submission interface.
3. Tools for setting up a Provenance recording session. Such tools may be used by a client for specifying: (1) the collection of components/services involved in the recording of process documentation, (2) the type of security protocol to be used (in collaboration with WP4), (3) the location of the provenance store, and (4) the type of data that must be recorded for each component (in collaboration with WP7 and WP8).

The capability of tools available within each of these categories are described in more detail below. Reference is also made to the Software Requirements Document (SRD) and the Logical Architecture. Access to the tool suite is made available only via the Application Programming Interface (API). A Graphical User Interface (GUI) and other components which are part of the tool suite make use of these API's to perform the required operation.

### 2.1 *Relationship to Software Requirements Document*

As described in [D2.2.1, section 2.5] of the SRD the provenance system should provide various services to store manage and query process documentation. The relationship of tool suite (WP6) in accordance to this comes mainly by providing query and analysis capabilities on the stored process documentation.

Some of the examples of the capabilities offered by the tools that would be used by the application services would include:

1. Capability to interact with the provenance system to analyse the stored process documentation.
2. Query capabilities that allow navigation on process documentation. It is however, worth mentioning that this capability should be independent of the query language used by the application and should not be limited to the capabilities provided by an individual provenance store on which the query would be performed.

The WP6 tool suite provides an Application Programming Interface (API) to all the services offered as referenced in SRD [D2.2.1, section 2.5] to insulate from the details of how the provenance system is implemented and also to provide a standardized set of operations.

Some of the operational capabilities that are described in this document relates to the following requirements captured in SRD.

1. Allow the retrieval of a provenance trace from the provenance store. Either a complete trace or a subset may be retrieved [D2.2.1, section 3.4.1]. This capability is provided by

the navigation tool in conjunction with the graph tool of the WP6 tool suite. The details of these tools are explained further in Section 3.1 and 3.8 of this document.

2. Allow the back-up of a Store to be taken. This will generally include an archiving facility that reads all the data from a provenance store, and records this as a disk file [D2.2.1, section 3.4.2]. This capability is provided by the navigation tool, which includes a trigger that is either initialized automatically over particular time interval or manually triggered by the application. The trigger will read the entire contents of the provenance store and would write it to a remote location.
3. Allow comparison to be undertaken with reference to a particular element (or groups of elements) within each p-assertion [D2.2.1, section 3.4.3]. This capability is provided by the comparison tool that performs the “similarity” check and the conflict tool that helps in detection of any conflict between p-assertions. A detail description on both these tools is provided in Section 3.5 and 3.6 of this document.
4. Allow the results of a query to the provenance store to be recorded into a file. A query in this context must be specified with reference to the XML Schema used to specify the contents of a provenance store. This Schema may be different from the particular recording format used internally within the provenance store. Such a Schema may be a combination of a general purpose schema (application independent) and an application specific schema. A query must use terms specified in the Schema, and may include conjunction or disjunction of terms (where this is appropriate) [D2.2.1, section 3.4.4]. This capability is provided by the navigation tool, which includes a logging function.
5. Allow a user to access a p-assertion based on the time and date at which the assertion was stored. This assumes that each p-assertion will have an associated time stamp, and an XML element that provides a suitable annotation for this. It is also assumed that the recording time corresponds to the clock at the provenance store, and not the time at which the provenance information was generated at the source [D2.2.1, section 3.4.5]. This capability is provided by the temporal tool of the WP6 tool suite. The details of this tool is explained further in Section 3.7 of this document.
6. Allow a user to specify a set of rules which can be verified with respect to the contents of a provenance store. Verification in this instance implies that the rule may be used to confirm whether the contents associated with a particular Schema element meets the constraints defined in the rule. A script may therefore be used to encode a “policy”. This capability is provided by the Analysis tool in conjunction with the Assertion tool of the WP6 tool suite. A detail description on both of these tools is provided in Section 3.3 and 3.4 of this document.
7. Allow a user to specify a time period in the future at which a provenance query may be submitted to a provenance store. A scheduler will be made available that allows queries to be stored to disk, and dispatched to the store in the future [D2.2.1, section 3.4.7]. This capability is provided by the navigation tool which includes a simple scheduler that accepts queries with time values, to allow the query to be run at a future time. This capability is based on the assumption that the results would be retrieved instantly on the execution of the query. This capability is provided by the Navigation tool.
8. Allow a service or user to specify the identity of the provenance store to which data must be recorded. Allow a service or user to choose the level of security they wish to be associated with the recording process. The level of security can range from no security, encrypted data, to more complex security mechanisms. Allow a user to specify the location of the archive to which data must be sent from a provenance store [D2.2.1, section 3.4.10-3.4.12]. These capabilities to allow p-assertion to be submitted to a Store, and for an application to initialize its recording process is provided by the set-up protocol [Omer05].

9. Access to the capabilities provided by the tools should be made available as an API. This is to allow such capabilities to be embedded within an existing application. This capability is fulfilled by all the tools that forms part of the WP6 tool suite.
10. Allow a user to visualize the contents of a provenance store. The visualization support may allow an easier way to access the XML-based content accessible from the Store. The graphical user interface is part of the navigation tool that allows application or a user to visually navigate and analyze the process documentation. A detailed description of the navigation tool is covered in section 3.1 of this document.

### **Schema Description**

We assume in this document that each p-assertion follows a P-Structure [WP3], a part of which contains elements that are derived from a specific application. We refer to this as the application schema.

## **2.4 Relationship to Logical Architecture**

The following requirements have been identified for the logical architecture:

1. A workflow description is provided by an application user. The representation should be in some XML format – which describes the set of activities and dependencies between the activities. It is assumed that each of these activities has corresponding p-assertions in the provenance store. The workflow description can then be verified by retrieving records from the provenance store. Work is underway with the applications (WP7 and WP8) to specify this XML workflow representation.
2. A workflow can be derived by analyzing p-assertions stored in the provenance store. This assumes that there is enough information within each p-assertion to enable a workflow graph to be reconstructed.
3. It is required that provenance store should be able to process XPath queries.
4. It is required that p-assertion should have a time stamp.

It is assumed that workflows are pre-defined, and specified by the application user/developer.

Some of the operational capabilities that are described in this document are satisfied by the following functionality provided by the logical architecture document.

1. The architecture provides presentation user interface which allows visualisation of query results and processing services' outputs. [WP3, section 3.3]. The navigation tool provided as part of the tool suite offers capability related to the logical architecture to perform browsing over provenance stores, visualise differences in different execution, and illustrate execution from a semantic viewpoint.
2. The architecture provides processing services for analysing and reasoning over recorded p-assertions [WP3, section 3.3]. The analysis, comparison, and conflict detection tool provided as part of the tool suite offers this capability. The capability can therefore be used to compare the processes that generate several data items, verify that a given execution was semantically valid, and identify points in the execution where results are no longer up-to-date in order to resume execution from these points.

Based on Figure 3.1 [WP3] the graphical tools provide support that can be added to an Application User Interface, the comparison tools provide aspects of Matchmaking to

allow comparison between p-assertions. Similarly, functionality within a Trace Comparator is supported by both the comparison tool and the conflict detection tool. In general, the tools outlined in this document try to provide instantiations of some of the capability that has been outlined as necessary within the Logical Architecture.

## **2.6 Types of Tools and Scope**

### **2.3.1. Tools for Accessing and Analysing provenance store Contents**

This first category of tools only read data stored in the provenance store, primarily through the use of the query interface available at the provenance store. A Provenance Trace in this instance is defined as a collection of p-assertions that have been retrieved as a result of a query. Two extreme conditions include: (1) only a single p-assertion matches the query, (2) all p-assertions match the query. The following capabilities are provided:

#### ***Navigating a Provenance Trace***

Navigation may involve simply retrieving all the p-assertions based on a query specified by the user in the navigation tool, or it may involve displaying p-assertions using a graphical format that demonstrates some relationships between the p-assertions visually. A user may specify some constraints on the number of p-assertions to be retrieved, or the type of p-assertions of interest.

A query in this instance is expected to be specified according to the schema made public through the Query API of the provenance store. Queries may be specified using XPath expressions, for instance. Retrieval of p-assertions from the provenance store may involve issuing a set of repeated queries via the provenance store Query API.

#### ***Analysing a Provenance Trace***

Analysis in this instance may involve specifying:

- a relationship between p-assertions, and using this as a constraint for determining what should be retrieved,
- a constraint on the type of data that must be contained within a p-assertion,
- a constraint on schema elements that may form part of a p-assertion. This is to be undertaken in collaboration with WP7 and WP8,
- a relationship between schema elements that form part of a p-assertion (defined with reference to the P-Structure of the p-assertion),
- a relationship between p-assertions based on their schema elements,
- a relationship between p-assertions based on their data.

Analysis in this instance would be undertaken based on the definition of a set of rules to specify “relationships” or “constraints” mentioned above. Such an analysis will also form the basis for comparing two traces, for detecting conflicts between traces, and for checking whether a trace is up-to-date.

Analysis in this instance would correspond to verifying a set of “assertions” on the provenance store, using the navigation tool also developed in the WP. Assertions model an atomic set of action that can evaluate to either true or false. Each of the 6 analysis modes defined above should be specified through this assertion mechanism. Assertions may be applied over elements of a p-assertion schema, or over the data contained within these elements.

A key aspect will be to identify how assertions may be specified using a language that is easy to use and adopt for the two application scenarios outlined in WP7 and WP8. As an example, consider the analysis of two traces which encode a particular “process”. In this

instance, a user may be interested to know whether a particular activity, or set of activities, have been undertaken in a particular order. The analysis in this instance will involve identifying a set of assertions, which if found to be valid, indicate that either a particular activity has occurred or not. Some examples are provided in Section 6.

The tool will therefore only provide the capability to specify a set of assertions that can be verified, in some order, by issuing queries to the provenance store. Describing an application specific requirement as a set of assertions will not be undertaken by the tool. It will therefore be necessary to work alongside WP7 and WP8 to identify some mechanism whereby a user can translate their higher level, application specific requirement, into a set of assertions that can then be confirmed using this tool.

Assertion definitions is currently being investigated. Two candidate technologies are being investigated: (1) The Java Expert System Shell/Drools, or (2) WS-Policy. The choice of these technologies are based on their suitability to the particular application scenarios outlined in WP2 (as part of Deliverable D2.2.1). Alternative query mechanisms such as those based on databases (using the SQL language) do not easily allow representation of analysis rules. Furthermore, such approaches do not enable such rules to be executed over an XML document easily. A number of alternative approaches, such as those based on Web Services Relationships Language [WSRL] do not have suitable implementations, and are often very restrictive in the type of reasoning support that they provide.

### **2.3.2. Comparing Provenance Traces**

The comparison tool makes use of the Analysis capability described above. Comparison in this instance involves verifying if two Provenance Traces are (1) identical, or (2) “similar”. Considering traces T1 and T2, the comparison tool will perform the following operations:

- Loosely identical: T1 and T2 are identical if they contain the same document structure (defined with reference to a p-assertion schema).
- Exactly identical: T1 and T2 are identical if they contain the same document structure (defined with reference to a p-assertion schema), and the same data associated with these elements.
- Similar: T1 and T2 are similar if some “semantic” similarity can be found between the elements contained in T1 and T2. Semantic similarity in this instance must be defined with reference to a domain specific ontology, for instance (interaction with WP7 and WP8 will be necessary to describe this).

A comparison performed between p-assertions assumes that they are defined according to the same schema. It is possible that a single provenance store may contain Pas that are specified with reference to different schemas.

### **2.3.3. Detecting Conflicts in Provenance Traces**

Conflict detection in this instance is used to determine if two p-assertions submitted either by a client and a service provider, for the same interaction (identical interactionId), contain differences. A conflict in this instance is therefore used to identify differences observed in the recording of the same event by different actors (in particular the client requesting a service, and the service provider). To support such a detection, it would be necessary to ensure that p-assertions corresponding to the same event in one or more provenance stores are actually recording the same event. For instance, if the interaction consists of exchange of a float value, the sender might record a float value in its p-assertion, and send it to the provenance store. However if the recipient is not designed to receive float value but only integer value, in this case, it will store within its p-assertion an integer value. As these two p-assertions are recorded with the same interactionId, the conflict detection mechanism is able to detect that, for the same interaction, the data used by the two parts is not the same. The conflict detection

mechanism therefore needs to evaluate the contents of a minimum of two p-assertion – each coming from a different actor.

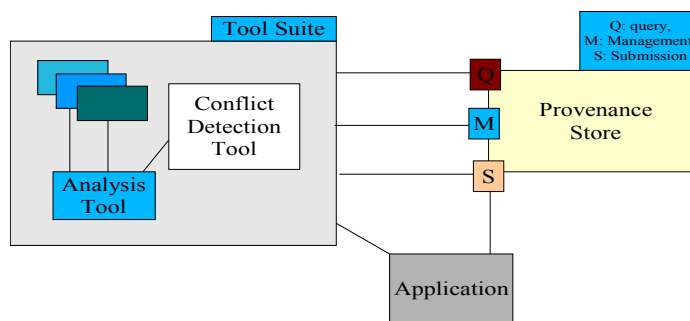
This example was rather simple and the conflict detection was easy mainly because of the type of the data we have chosen. However, data type stored can be application specific type, so the conflict detection tool also uses the data comparison architecture provided by the analysis tool being developed in this WP.

**2.3.4. Checking that a trace is up-to-date**

This check will be undertaken by examining the time stamps associated with a p-assertion – based on the time stamp of the provenance store (i.e. the time when the p-assertion was recorded). This tool will therefore identify when a p-assertion was last recorded for a given actor, and use this as a basis to determine if a submission has been made by an actor recently. The tool may also enable a user to determine how many submissions have been made by an actor over a given time period, or the time interval between a given number of submissions. This tool will also make use of the analysis tool being developed in this WP.

Figure 1 illustrates the interaction between the tool suite (WP6) with other components within the Provenance System. A key component within the tool suite is the Analysis tool, which is made use of by all the other tools. Interaction with the provenance store is via the Query, Management and Submission APIs. The tool suite may contain both an API to allow other application to interact with the provenance store, or it may contain a graphical interface (in the case of the Navigation tool, for instance). An Application may submit data to the provenance store, but must make use of the the capability provided in the tool suite to interact with the provenance store.

**2.8 General Constraints and Assumptions**



**Figure 1: High level interaction between Tool Suite, provenance store and Application.**

Figure 1 illustrates the interaction between the tools, the provenance store and the Application. This demonstrates that an Application user can make a direct submission to the provenance store.

### **3 Tool Suite Description and Implementation**

The two main tools that form the core of WP6 are the Navigation and Analysis tool. Others include the graph tool, comparison tool, and conflict detection tool. Each of these tools plus their relationship with each other is detailed below.

#### **3.1 *Navigation***

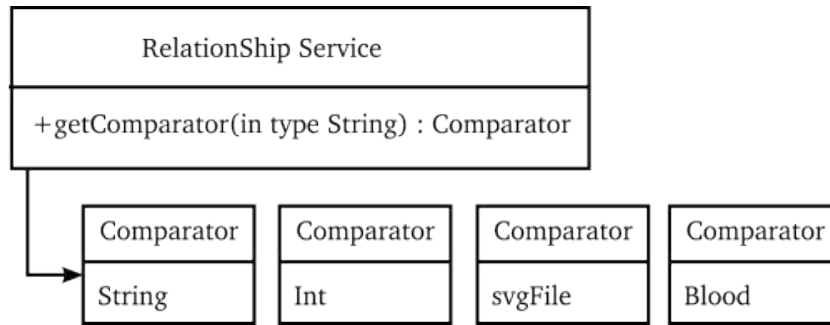
The main objective of the navigation tool is to provide interfaces to retrieve, display and analyse p-assertions. The two main `getAll()` public method which accepts XPath query as one of the inputs, acts as the link between the application and the navigation interfaces. Each of the two methods is supplemented with either an option to retrieve a graphical representation of the resultant p-assertions, or to restrict the total number of p-assertions retrieved from the query. The tool offers capability to perform repeated queries on the provenance store to retrieve specific p-assertions that meet the constraints defined by an application user. Navigation tool is linked to the Graph tool that provides graphing capability and Analysis tool that allows application users to test relationships between the retrieved p-assertion's. The relationship tool interfaces with the navigation tool to provide relationship definition that would allow retrieving results based on a particular constraint.

The analysis tool interacts with the navigation tool to also retrieve p-assertions. This interaction however represents two different aims (i) First, the analysis tool formulates rules that can be directly converted to XPath queries thus being able to answer the rules just based on the query results returned by the navigation tool (ii) Second, analysis tool interacts with the navigation tool repeatedly to retrieve results for various XPath queries, which are then processed by the analysis tool using the help of assertion engine to formulate the result for a rule analysis.

#### **3.4 *Relationship Definitions***

Data recorded within a p-assertion can be either a simple data type like string or integer, or can also represent application specific data types like a sound or an image. In the former case, data type are well-know and supported in virtually all programming languages, thereby allowing us to compare them easily. However, in the latter case, we cannot make any supposition how to compare these data. Hence, the aim of this tool is to:

- provide a generic tool to compare arbitrary data by providing an interface used by other tools;
- allow the comparison of application specific data by providing a plug-in enabled architecture. An application should provide one or more plug-ins according the type of data it handles. Each plug-in provides a data type specific mechanism to compare between two entities that are of that type.



**Figure 2 Comparison Plug-in Architecture**

When a new data type needs to be handled, the corresponding plug-in must be registered using the relationship tool. The relationship tool stores all plug-in in a comparison store. At registration time, a plug-in provides to the comparison tool the type of data it can handle. This plug-in store is co-located with the relationship tool. The name associated with a given data type must be defined within the application schema.

Each plug-in contains only one method with the following signature:

```
int compare(in String data1, in String data2)
```

The two parameters (data1 and data2) represent the data recorded within p-assertion we want to compare. According to the application and the plug-in logic, these data could represent, for instance, a URI (a file on a local hard disk, on a web page, on a ftp server), or the actual data object that needs to be compared.

The returned value type is an integer, because this data type is widely supported. The semantics of the returned value is (in this instance, <, > and = are assumed to have some meaning based on the plug-in being used):

- -1 if data1 < data2
- 0 if data1 == data2
- 1 if data1 > data2

### 3.5 *Analysis*

The main objective of the analysis tool is to provide an assertion checking interface to all the other tools. The analysis tool returns a Boolean outcome, indicating whether the particular condition being analysed holds or not. The analysis tool provides two main methods `recordMatch()` and `recordSchemaMatch()` to the comparison and the conflict tools. The first method provides an interface to check if two p-assertion are similar, and the later one checks if a p-assertion conforms to a given schema. The first method will be used through the navigation interface while the second method would primarily be used through the application interface. All of the methods in this tool assume that the reference to the p-assertion is sent as part of the string argument – indicating therefore that each of the methods here apply to a p-assertion that has already been defined. Other methods in the analysis tool provide the following functionality:

**a)** `check(in element : String)` - checks if the element exists within a p-assertion schema. For example, checks if an element “temperature” exists within a p-assertion schema.

**b)** `check(in data : String)` - checks if a particular data value exists within a p-assertion. For example, checks if a data value of “34” exists within a p-assertion.

**c)** `check(in element : String, in data : String)` - checks if the element and the data exist within a p-assertion. For example, checks if an element “temperature” consisting of a data value “34” exists within a p-assertion.

**d)** `checkRelation(in data : String, in relation : String)` - checks if data exists within a p-assertion that meets the required relationship constraint. For example, checks if a data value of greater than “34” exists within a p-assertion. In this example the data string would consist of “34” and the relation string would consist of “greater than”.

**e)** `checkRelation(in element : String, in data : String, in relation : String)` - checks if an element and data exists within a p-assertion that meets the required relationship constraint. For example, checks if an element = temperature consisting of a data value greater than 34 exists within a p-assertion.

**f)** `checkSubsumption(in element1 : String, in element2 : String)` - checks if two elements within a p-assertion schema are linked by subsumption relationship.

#### 3.4.1 *p-assertionAM : Provenance Assertion Abstraction Mechanism*

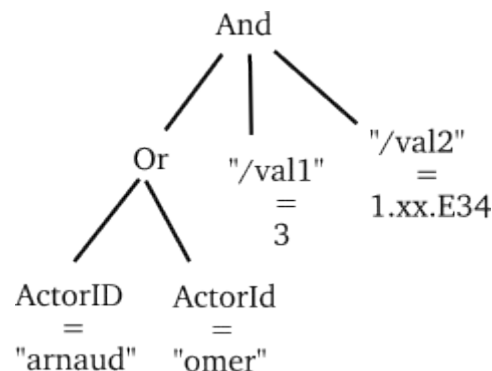
This mechanism provided a generic set of rules that will be manipulated by any assertion engines through specific wrappers. An assertion engine here represents an interpreter that can process rules applied on a p-assertion. This rule allows to check basic constraints on a p-assertion. Basically, a rule returns true or false when evaluated against a p-assertion.

Several kinds of rules can be used :

- Basic rules validate a condition on a particular data located in the application specific part of a p-assertion. These condition are : equality, greaterThan, lesserThan, exists, notNull.

- Meta rules check a condition on the provenance specific part of a p-assertion like Date Invocation Time, Date Receive Result, Actor Name, Client Id, Service Id, etc.
- Operator rules combine basic rules and meta rules using operators like AND, OR, NOT

An example of an assertion combining user- and provenance-defined rules is shown in figure 1b. This assertion requires that, for application data, data val1 equals 3, data val2 equals 1.xx.E34 (val2 is an application specific data) and, for provenance related data, the actor of this record could be either “Arnaud” or “Omer”



**Figure 3 Assertion combining User and Provenance defined data**

As we want to create a generic mechanism, a given rule should be able to handle various kinds of data. For example, the rule which checks the equality of two data is not designed to know how to compare these data. For this action, the p-assertionAM relies on the relationship tools we introduced previously.

Hence, when a rule is instantiated with a XPath string, the first thing it does is to look for the type of the value designed by the string inside the schema. Next, it asks the relationship service for a comparator tool matching the type. Finally it checks the value according to its own behavior and returns a boolean.

### ***3.6.1 Assertion Engine***

The assertion engine tool acts as a core to the analysis tool. The main task of the assertion engine tool is to provide an interface to accept assertions from as an input and provide an appropriate Boolean output. Many candidate technologies are currently being investigated for assertion definitions: (1) The Java Expert System Shell, (2) WS-Policy, or (3) OpenRules.

#### ***Bindings***

For the moment, Rules are designed as standard Java classes.

### **3.9 Comparison**

The comparison tool makes use of the Analysis capability described above. Comparison in this instance involves verifying if two p-assertions are (1) identical, or (2) “similar”. The comparison tool consist of two main types of analysis one that accepts a particular type of match and the p-assertions to provide a Boolean result, while the second one just accepts p-assertions and compares them to return a string that specifies what type of match exists between the p-assertions. A method is also provided within the comparison tool that allows an application user to specify a plug-in algorithm using the navigation interface. The plug-in algorithm will represent an application users understanding of similarity and based on this the analysis will be performed for p-assertions. Comparison tool as mentioned before also has interface to either analyse relationship between either two p-assertion or a single p-assertion and the schema.

### **3.11 Conflict**

Conflict tool interfaces with the analysis tool to provide conflict detection mechanism. Conflict detection in this instance is used to determine if two p-assertion submitted by a client and a service provider, for the same interaction, contain differences. A conflict in this instance is therefore used to identify differences observed in the recording of the same event by different actors. To support such detection, it would be necessary to ensure that p-assertions corresponding to the same event in one or more provenance stores (provenance stores) are actually recording the same event. Conflict tool provides similar functionality as mentioned in the comparison tool in terms of analysing either an (1) identical, or (2) “similar” conflict has occurred and providing plug-in mechanism for user defined conflict detection.

### **3.13 Temporal**

This is a particular tool that may be used by the analysis and conflict tools, and deals with providing methods that can analyse information on time relationships between p-assertions. This check will be undertaken by examining the time stamps associated with a p-assertion – based on the time stamp of the provenance store (i.e. the time when the p-assertion was recorded). This tool will therefore identify when a p-assertion was last recorded for a given actor, and use this as a basis to determine if a submission has been made by an actor recently. The tool will also enable a user to determine how many submissions have been made by an actor over a given time period, or the time interval between a given number of submissions.

It is useful to note that time stamps are provided by the provenance store and not be actors making a p-assertion. The reason for this is that it cannot be assumed that actors making p-assertion to a given provenance store have synchronised clocks. Therefore, a p-assertion is only acknowledged as being made after it has been recorded within the provenance store, and has been given a time stamp.

### **3.16 Graphical Interface and Portlets**

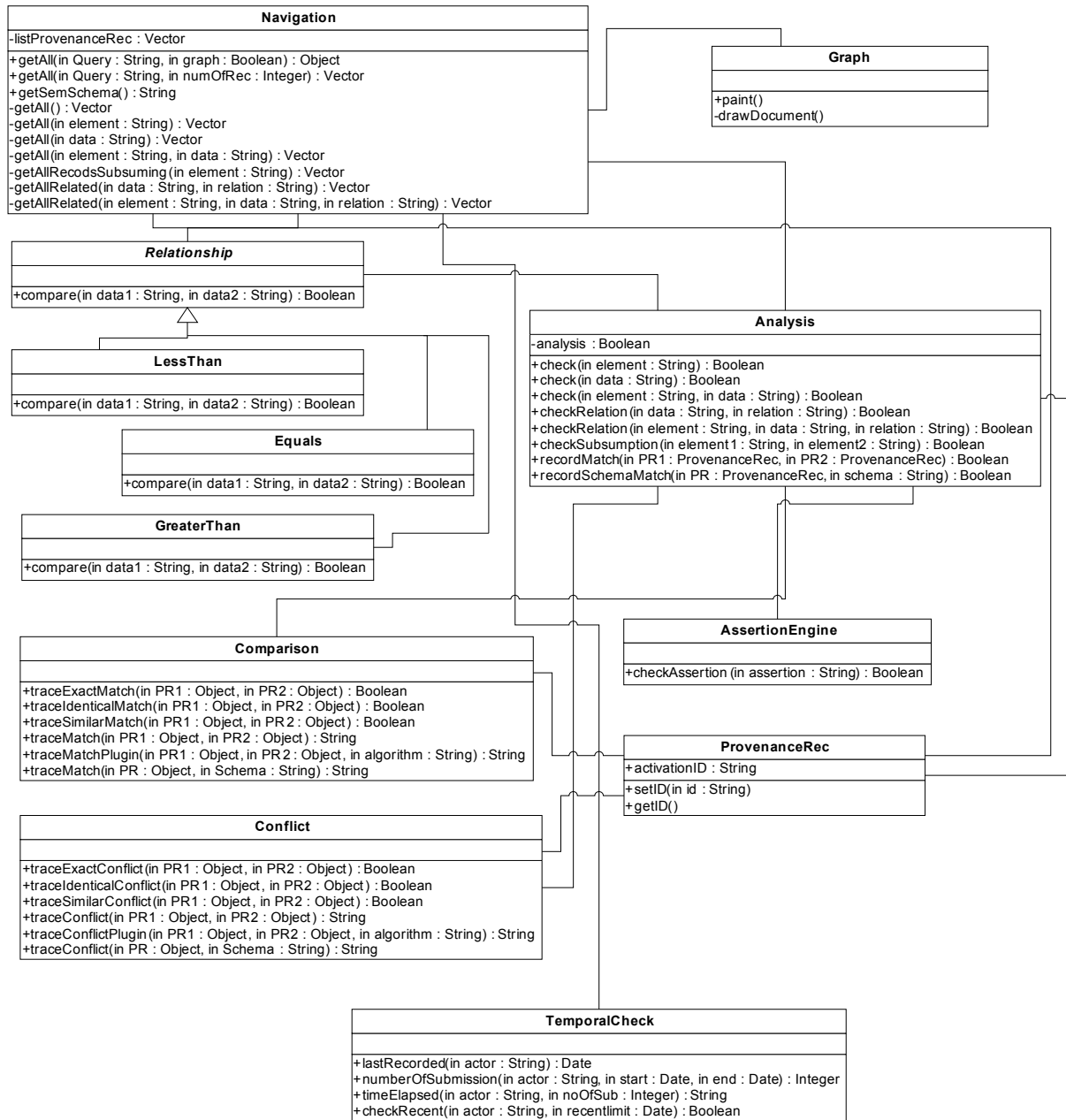
The Graph tool is used for displaying p-assertions using a graphical format that demonstrates relationships between the p-assertions visually. The GUI tool provides an interface that is used by the navigation tool to retrieve the graphical object. In the first instance, the Graph tool will simply provide a mechanism to view the structure of a p-

assertion – by plotting the tree associated with a p-assertion schema, and mapping data values associated with each p-assertion to this schema. Subsequent versions will also contain additional methods, that would allow a navigation tool to send in a “graph type” request (the type corresponding to different visual), which would then allow application users to visualise different types of graphs.

There are a number of different ways in which workflow may be displayed to a user. Each makes use of a process graph that demonstrates how a collection of processes are executed in a sequence. An example of such an approach for displaying workflow can be found at: <http://www.ilog.com/products/jviews/workflow/>

A provenance trace can be displayed graphically. Two main representations are available :

- Workflow hierarchy (default) displays all the p-assertions contained within a Provenance Trace. If one p-assertion has been generated after another (based on a time stamp), an arrow is displayed from the ancestor to the child. This description is also dependent on the workflow graph that has been supplied by a user (Section 7).
- An alternative representation technique would make use of a time line. This representation may be used to compare two similar provenance traces – essentially by monitoring when particular p-assertion have been submitted by a user. The duration of each step can also be compared using this approach. A user may interact with this time line representation by clicking on a trace and deriving information about the p-assertions associated with a particular trace.



**Figure 4: UML Class Diagram of the Tool Suite**

## 4 Tools and Provenance Store Interaction

The sequence diagram [Figure 5] helps illustrate the communication between the tool suite (WP6) and the provenance store. All the interaction between the various tools and the provenance store is consists of either: (1) query for data contained within a p-assertion, or (2) query to determine the schema for a p-assertion. Below we detail different interactions taking place between the tool suite and provenance store.

**Navigation – provenance store interaction:** The Navigation tool provides browsing capabilities for content maintained in the provenance store. This capability extends that already provided by the query API of the provenance store. For instance, the query API is generally intended to manage XPath queries, whereas the Navigation tool can allow Visualization of results from such XPath query, or specialist conditions (generally specified as a set of rules) that need to hold over the results that have been generated through XPath.

Once a query has been received by the Navigation tool, it first breaks the query into sub-queries, converts each into an XPath query, and submits these to one or more provenance stores. The number of provenance stores involved is based on the configuration provided by a user during the setup process. Subsequently, the navigation tool interacts with the provenance store Query API to retrieve the required results for each of the sub-queries. This process would be repeated several times until all the sub queries are processed.. Once all results have been received, a final processing is performed by the navigation tool to generate a response based on the collected information from the provenance store. The two main public methods that initiates the navigation tool to query the provenance store are: a) `getAll(in Query String, in graph Boolean)` and b) `getAll(in Query String, in numOfRec Integer)`. Both of the above APIs accept queries in different formats from the application, these are then converted to the XPath query format, so that the available Query API at the provenance store can be used from this point onwards.

Most of the interactions taking place between the tool suite and the provenance store is therefore through the Navigation tool. The outcome of a query generated to the provenance store may be therefore be passed on to other tools for analysis, or may be stored for delivery to the end user.

**Analysis – provenance store interaction:** The Analysis tool provides capabilities to compare relationships between a p-assertion and the current Schema used to specify the structure of content within one or more provenance stores. In most cases, the Analysis tool may also use results generated through the Navigation tool. However, some specialist interactions are also provided between the Analysis tool and the provenance store to enable faster operation. One such interaction involves retrieval of the schema of a provenance store in order to perform either similarity checks or conflict detection.

The interaction between the provenance store and the tool suite are limited to just the above two interaction for now, but in future new tools that might get added to the tool suite may provide additional capability.

## 5 Tools and Application Interaction

The sequence diagram [Figure 5] helps illustrate the communication between the tool suite (WP6) with other components within the Provenance System. A key component within the tool suite is the Analysis and the Navigation tool, which is made use of by all the other tools, namely the compare, conflict and temporal tools. Below we detail different interaction taking place within the sequence diagram.

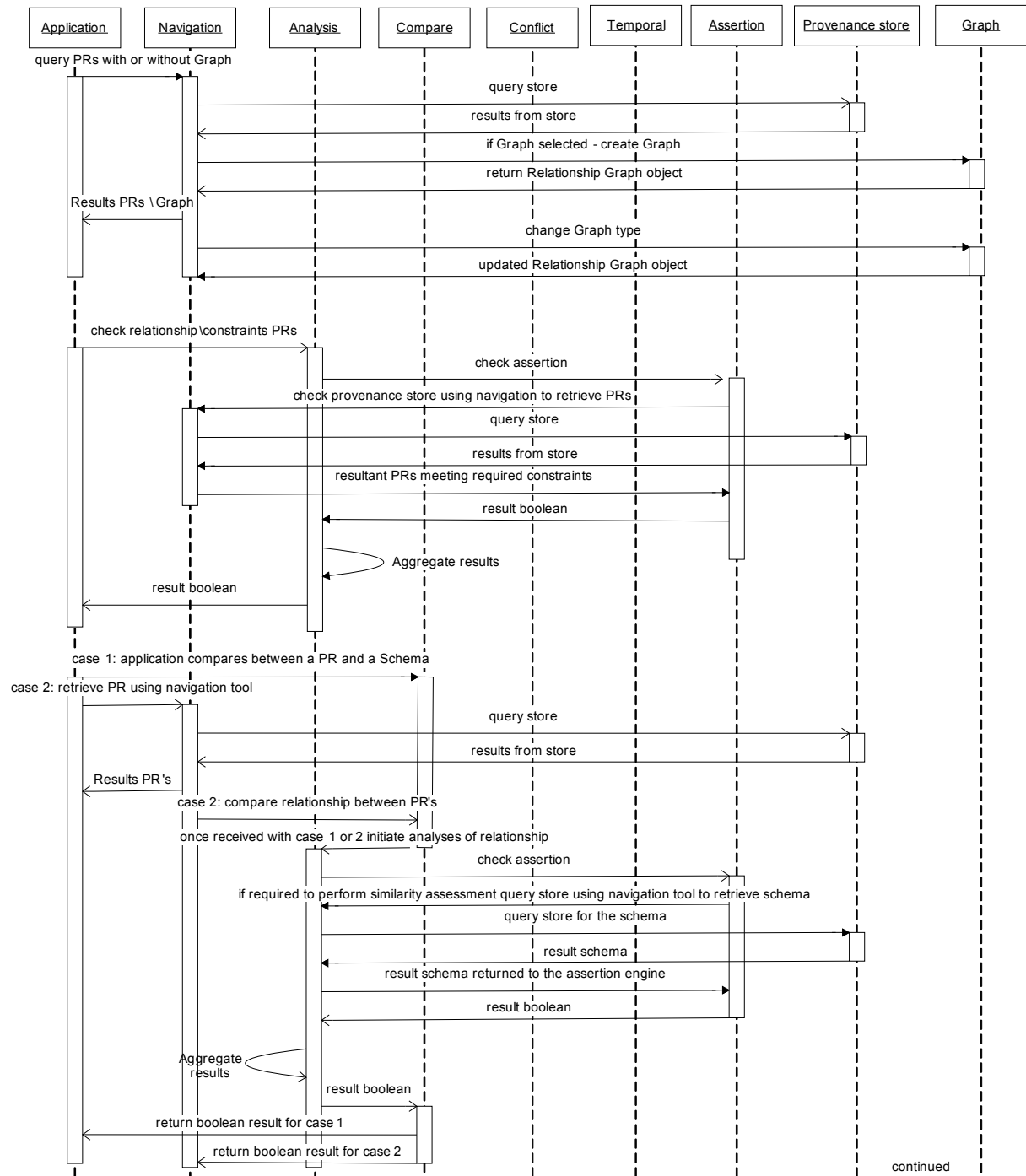
**Application-Navigation interaction:** Communication starts between the two components with the application tool issuing a query call to the navigation tool. This query call can include retrieving p-assertions with or without a graphical representation option. This method call will include the query plus a String input which would identify an applications interest in retrieving the graph type. One of the graph type would include an option of no graph, this would result in retrieval of p-assertions only. The navigation tool interacts with the provenance stores Query API using the query preference to retrieve the required results. Once returned with a set of results, the navigation tool interacts with the Graph creation tool, to create appropriate visualisation of the returned result. Once completed, results are returned to the application.

**Application-Analysis interaction:** An Application tool could interact directly with the analysis tool to check assertions. These assertions can either be on an element, a data or a combination of an element and data. Assertions can also include analysing relationships based on some constraints over the data contained within the element. Once a method call is initiated by the application tool with appropriate input parameters, the analysis tool uses the assertion engine to analyse assertion. Once results are retrieved from the assertion tool on a single assertion the analysis tool may need to aggregate the results to formulate the final Boolean result for multiple assertion, which is then sent to the application tool making the initial call.

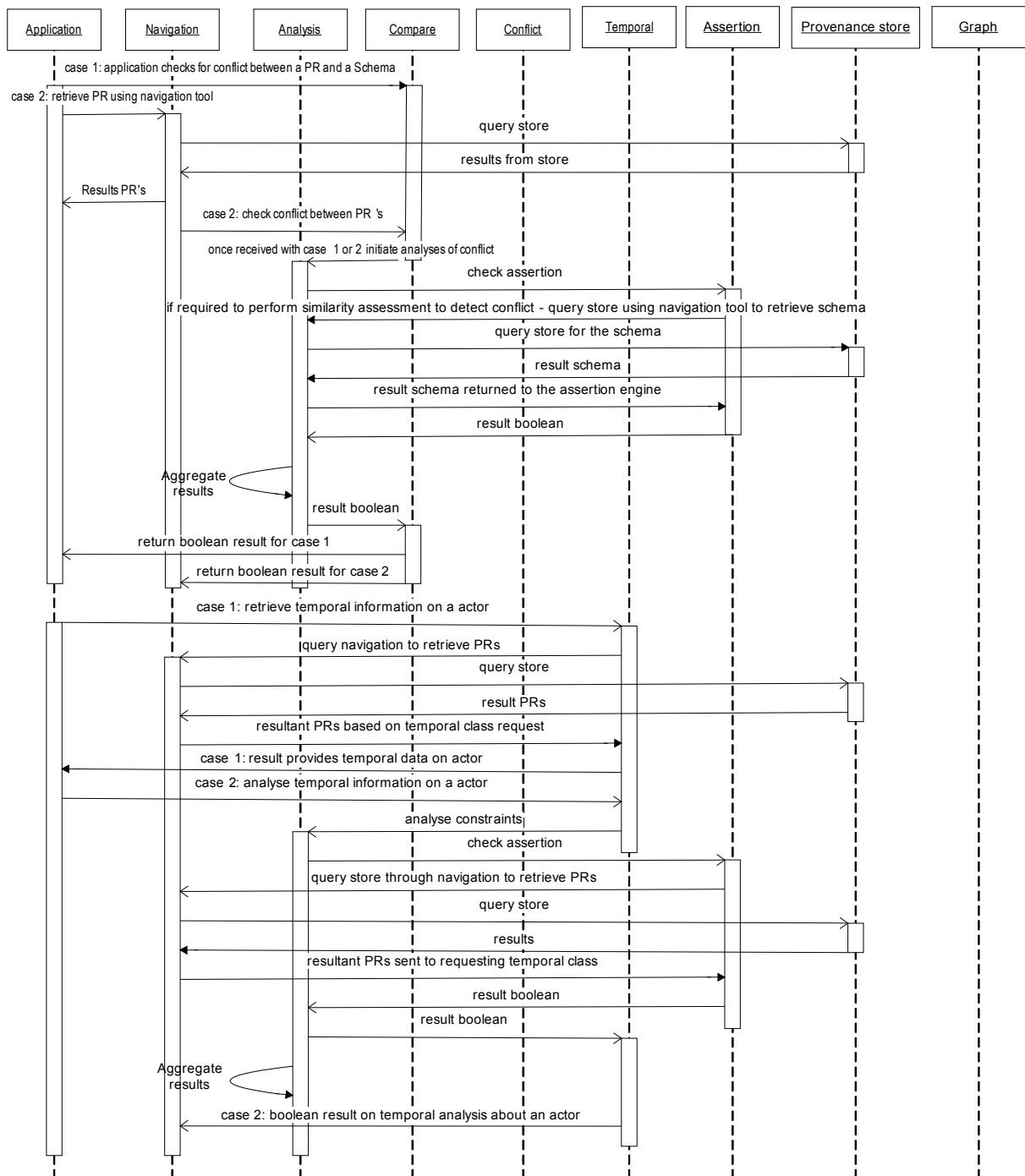
**Application-Comparison interaction:** The interaction between the application tool and the comparison tool could either be initiated directly if the comparison is taking place between a p-assertion and a schema or could be via the navigation tool, once the user has acquired appropriate p-assertions to compare. Interaction between the application and the comparison tool for each of the above two cases are displayed in the sequence diagram in figure 2. The comparison tool upon receipt of a call from either of the two cases, initiates a further call to the analysis tool to analyse the relationship. Analysis tool as always uses the assertion engine to analyse relationships between the p-assertions, upon completion, the analysis tool sends the result to the comparison tool which in turn forwards the result to the application tool making the initial call.

**Application-Conflict interaction:** This interaction between the application tool and the conflict detection tool is exactly the same as the interaction between the application tool and the comparison tool except that in case of comparison tool the aim of the analysis tool is to compare similarity, while in the later case it is to detect conflicts.

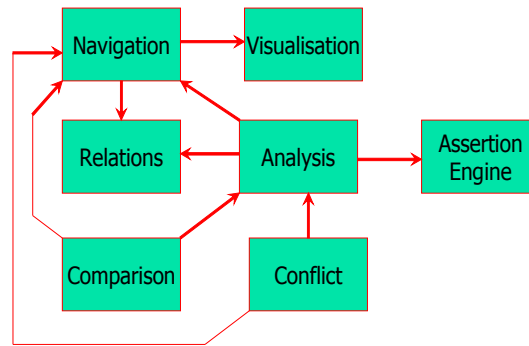
**Application-Temporal interaction:** The interaction between the application and the temporal tool initiates by a call from application. The temporal tool allows two main types of interaction. One that retrieves temporal information about an actor and second to ascertain some fact. In the first case, the temporal tool interacts with the provenance store using the navigation tool to retrieve the required information, which is then sent to the application making the initial call. For the later case, the temporal tool uses the analysis tool which in turn interacts with the assertion tool to analyse a fact. The temporal tool sends the results obtained from the analysis tool back to the application initiating the method call.



**Figure 5: UML Sequence Diagram of the Tool Suite**



**Figure 5 - continued: UML Sequence Diagram of the Tool Suite**



**Figure 6: Tool Interactions**

Figure 4 illustrates the interaction between the tools. Each arrow illustrates a “makes use of” relationship. Hence, the Navigation Tool makes use of a Visualization Tool. The Analysis tool is a significant component within the system, as explained previously.

## 6 Examples of Usage

Some queries that may be handled by the tools include the following (in this description  $\langle e \rangle$  refers to an element in the p-assertion, and  $d:\langle e \rangle$  refers to data value associated with element  $\langle e \rangle$ ) are discussed below. It is assumed that the p-assertion schema already exists and is specified as part of the setup protocol by the application user.

- Find all p-assertions submitted by actor  $\langle a \rangle$ 
  - The result of this query is a PT containing all the p-assertion that contain the element  $\langle a \rangle$ . The query makes use of the Navigation tool.
  
- Find a p-assertion submitted at/over time  $\langle t \rangle$ 
  - The result is a p-assertion or a PT which may be ordered based on the some start time also specified by an application user. This query makes use of the Navigation tool, and the results can then be ordered using the Analysis Tool.
  
- Find n p-assertions containing element  $\langle e \rangle$ 
  - Returns the first n p-assertion that can be found in the provenance store. This is to restrict the total number of p-assertions returned. Makes use of the Navigation tool.
  
- Find n p-assertions containing data  $d:\langle e \rangle$ 
  - Returns the first n p-assertions. This is to restrict the total number of p-assertions returned. Makes use of the Navigation Tool.
  
- Is condition  $\langle X \rangle$  on provenance store valid. Where  $\langle X \rangle$  is defined using an application schema + the set of relationships that must hold on  $\langle X \rangle$ .
  - Returns True/False. Makes use of the Navigation and the Analysis Tools.

Based on the Organ Transplant Management (OTM) application, it is possible to specify the following queries:

Query 1: Retrieve metadata and references to all actions/events associated with a case. In this instance, the Navigation Tool is used, with the query  $\langle e \rangle = \text{CaseID}$ , retrieve all p-assertion.

Query 2: Determine decision tree for a particular case. In this instance, the Navigation Tools: Given  $\langle e \rangle = \text{CaseID}$ , retrieve all p-assertions associated with  $\langle e \rangle$ . The results of the Navigation Tool are then compared based on a  $\langle \text{timestamp} \rangle$  on each p-assertion, which may be used to order the p-assertion.

Query 3: Determine if a particular medical staff member was involved in a particular decision

In this instance, the Navigation Tool is used to first retrieve p-assertion that contain the CaseID and the particular staff member. The Analysis Tool is then used to check whether a given p-assertion contains both the CaseID and the mentioned individual. Hence, Given ( $\langle e1 \rangle = \text{CaseID} \ \&\& \ \langle e2 \rangle = \text{PersonX}$  True?

## References

- [D2.2.1] Project Deliverable D2.2, “Software Requirements Document”. Version 1.0, 2005,  
<http://twiki.gridprovenance.org/pub/Restricted/DeliverableD2dot2dot1/SRD-v1-ofr-JBI10-03-05.sxw> (Project internal web site).
- [Moreau05] “Logical Architecture Strawman for Provenance Systems”. Version 1.11, 2005,  
<http://twiki.gridprovenance.org/pub/Restricted/LogicalArchitecture/strawman1-11.pdf> (Project internal web site).
- [D8.1.1] Project Deliverable D8.1.1, “Specification of mapping to provenance architecture, and domain specific provenance handling”. Version 1.0, 2005,  
[http://twiki.gridprovenance.org/pub/Restricted/DeliverableD8dot1dot1/OTM\\_Mapping\\_and\\_Provenance\\_Handling\\_Document\\_v0.62.sxw](http://twiki.gridprovenance.org/pub/Restricted/DeliverableD8dot1dot1/OTM_Mapping_and_Provenance_Handling_Document_v0.62.sxw) (Project internal web site).
- [WP3] “An Architecture for Provenance Systems”. Version 0.3, 2005,  
<http://twiki.gridprovenance.org/pub/Restricted/LogicalArchitectureFrozen/logarch-v0.3.pdf> (Project internal web site).
- [Omer05] “WP6: Provenance Setup”. May 2005,  
<http://twiki.gridprovenance.org/pub/Restricted/WorkPackage6/wp6-setup.pdf> (Project internal web site).
- [Jetspeed] <http://portals.apache.org/jetspeed-1/>, September 2005.
- [Gridsphere] <http://www.gridsphere.org/gridsphere/gridsphere>, September 2005.
- [Liferay] <http://www.liferay.com/web/guest/home>, September 2005.
- [eXo] <http://www.exoplatform.com/portal/faces/public/exo>, September 2005.
- [WSRL] <http://webservices.sys-con.com/read/39555.htm>