Title:	*Software Requirements Document*
Author:	*Work Package 2 ("Requirements Capture")*
Editor:	*Árpád Andics (MTA SZTAKI)*
Reviewers:	*John Ibbotson (IBM), László Zs. Varga (MTA SZTAKI), Omer F. Rana (UWC), Luc Moreau (SOTON), Victor Tan (SOTON), Steven Willmott (UPC)*
Identifier:	*D2.2.1*
Type:	*Deliverable*
Version:	*1.0*
Date:	*Monday, May 02, 2005*
Status:	*Public*

Summary

This document contains the set of software requirements for the provenance architecture. It has been prepared based on the analysis of the User Requirements Document (deliverable D2.1.1) by the developers of the provenance architecture. This document forms the basis for the system design and it is also the reference against which the prototype system will be verified.

The PROVENANCE project receives research funding from the European Community'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 fur Luft- und Raumfahrt s.V.	Germany
Universitat Politecnica de Catalunya	Spain
Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutató Intézet	Hungary

Foreword

This document has been compiled by Árpád Andics (MTA SZTAKI) based on studies of the members of the developer team on the contents of User Requirements Document (deliverable D2.1.1, result of Work Package 2, Task 2). Individual contributors and sections they have contributed are as follows:

- John Ibbotson: Model description (section 2.5)
- Luc Moreau: Relation to other systems (section 2.3), Model description (section 2.5)
- Omer Rana: Model description (section 2.5), Functional requirements (section 3.1)
- Victor HK Tan: Security requirements (section 3.6)
- László Varga: Function and purpose (section 2.1), Relation to other systems (section 2.3)

The document has been prepared in accordance with the recommendations of the ESA Software Engineering Standard PSS-05-02. The ESA standard was partially followed, because the standard is designed for software production, whereas this document defines the software requirements for architecture design.

Table of Contents

Table of illustrations

Figures:

List of acronyms

- DMG – DataMiningGrid (demo application scenario)
- ESA – European Space Agency
- HCI – Human-computer interface
- HLSF – Healthcare and Life Sciences Framework (demo application scenario)
- OTM – Organ Transplant Management (demo application scenario)
- TMA – Traffic Management Application (demo application scenario)

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.
- *Provenance record:* Provenance data submitted to the provenance system's interface for recording purpose.
- *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.

References

1 Applicable documents

- "ESA Software Engineering Standards" by C. Mazza, J. Fairclough, B. Melton, D. de Pablo, A. Scheffer, R. Stevens. Published by Prentice Hall 1994.

2 Reference documents

- [PASOA] Miles, S., Groth, P., Branco, M. and Moreau, L. The requirements of recording and using provenance in e-Science experiments. Technical Report, Electronics and Computer Science, University of Southampton, 2005. http://eprints.ecs.soton.ac.uk/10269/

1 Introduction

1.1 Purpose of the document

The purpose of this document is to form the basis of system design and verification by documenting the requirements on the provenance architecture that have been determined based on the analysis of the User Requirements Document by the software architects and developers of the project team. This document presents the developers' view on the required functionality of the provenance architecture based on the examination and evaluation of the end users' requirements, which were captured in the User Requirements Document.

This document constitutes the problem analysis phase of the life cycle of the Provenance project. The document is based on the ESA Software Engineering Standard PSS-05-02. The ESA standard was partially applied in order to adapt it to architecture requirements capture.

The document is addressed to all project partners involved in the design, implementation, testing and deployment of the provenance architecture. The document is a primary input for Workpackage 3 ('Architecture'), Workpackage 6 ('Tools and Set up') and Workpackage 9 ('Implementation, Integration and Test').

The provenance architecture is meant to be the basis for provenance system implementations, therefore the software requirements in this document should capture the core of the provenance architecture and be long lasting. The aim is to make the requirements document as concise as possible and free from less relevant details.

1.2 Scope of the Provenance Architecture

Provenance enables users to trace how a particular result has been arrived at by identifying the individual services, aggregation of services as well as intermediate data used in producing the result. The overarching aim of the Provenance project is to design, conceive and implement an industrial-strength open provenance architecture for grid systems, and to deploy and evaluate it in complex grid applications, namely aerospace engineering and organ transplant management.

The provenance architecture itself is not a direct software product, but the main concepts and relations of provenance systems in general. Based on the provenance architecture a provenance system can be developed and deployed in applications.

The provenance system is used to document, analyse and prove how a piece of data was produced in an application. The provenance system is different from a logging system, because it provides additional and different functionalities. The provenance system may not log all information, may not log information directly related to the operation of the application, or may not even log information of the operation of the application. In general the information entered into the provenance system and returned by the provenance system is not like a log file. The provenance system is not a monitoring system, because it does not provide real time information, rather it provides post event analysis and investigation support. Performance oriented information is subject of the provenance system only if it is relevant to the production of a piece of data. The provenance system provides information which is derived from or part of the documentation of how a piece of data was produced.

1.3 Overview of the document

The document contains the general description of a provenance architecture followed by the set of requirements identified during the requirements capture and evaluation process. The basis for this evaluation was the User Requirements Document, which was produced based on the exploration on

several application scenarios ranging from academic research projects to industrial applications in the field of healthcare, aerospace, scientific research, data mining, traffic management and digital libraries. (For a detailed description of these application scenarios refer to section 2.2 of the User Requirements Document.) Though the need to manage provenance information exists for a long while in several application areas, our experience shows that the intention to use principled methodologies for this task is new even in the majority of these scenarios. Most of the projects we have surveyed are just about to find an adoptable tool for this purpose, while other projects are in the state of problem recognition and exploration of possible solutions. For this reason, the majority of the requirements on a provenance architecture we have collected and evaluated during our work express requirements based on experiences with legacy and experimental systems and problem analysis conclusions.

The structure of the document is as follows:

- Chapter 2 describes briefly the motivation behind the project, which is followed by a high level description of the crucial aspects of a provenance architecture including function and purpose, system context, constraints on the development process and an abstract model for the further discussion.
- Chapter 3 gives a classified listing of the requirements imposed on the provenance architecture as inferred from the user requirements recorded in the User Requirements Document (URD, project deliverable D2.1.1). Additional requirements originating from system developers as well as reasoning for rejected user requirements are also contained in this chapter.
  As part of the adoption of the ESA standard to architecture requirements capture we omitted some recommended subsections that we consider to be not applicable for our subject. These subsections are: Resource requirements, Verification requirements, Acceptance testing requirements, Portability requirements, Quality requirements, Reliability requirements, Maintainability requirements.
- Chapter 4 contains two tables, of which the first gives a mapping of user requirements to software requirements, while the second is a tabular summary of the software requirements listed in chapter 3 including source user requirements as well. The two tables together act as a traceability matrix between the set of user requirements (documented in the URD) and the set of software requirements, which is the essential content of this document.

Requirements presented in this document are assigned the following priorities:

- *Essential:* A requirement is marked as ‘essential’ if any of the source user requirements is ‘essential’, which means that a demo application is interested in the given feature. These requirements have high priority.
- *Desirable:* A requirement is marked as ‘desirable’ if all source user requirements are ‘desirable’, meaning that they originate from use case(s) other than the demo applications. These requirements have normal priority.
- *Nice to have:* A requirement is marked as ‘nice to have’ if it originates from user requirements having the same priority. It denotes optional features. These requirements have low priority.
- *Critical:* Requirements having ‘critical’ source user requirements are marked with this flag. It should be considered for the highest priority for the requirement.

Each requirement is flagged with one of ‘essential’, ‘desirable’ or ‘nice to have’ according to the above rules. The ‘critical’ flag is an extra flag that might be assigned to a requirement.

Requirements are labelled by the following pattern:

SR - X - Y [- Z]

where:

- SR stands for ‘Software Requirement’

- X is a number that corresponds to different sections in this document; and
- Y is either the ordinal number of a requirement within a section or it corresponds to a subsection within a section and Z is the ordinal number of the requirement.

2 General Description

As [PASOA] points out there is no existing technology at the moment that provides a principled, application-independent way of provenance information handling, therefore our project is to do pioneering work in the field including scientific research and experimention. The achievement of the goals set by the project can give a significant impulse to the evolution and application of grid technologies by improving the trust of grid computations and data handling. With the help of the provenance system users can check how a result was achieved resulting in improved trust in the result.

2.1 Function and purpose

The purpose of the software architecture to be developed within the project is to enable provenance handling in grid environments in a principled, application-independent way. Provenance handling includes the recording, management and analysis of provenance information.

The provenance system will support the recording of information related to the production of a piece of data within an application. The application is executed within an information system through distributed computational steps. The provenance may include the steps of the computation, the partial results of the steps, the execution place of the steps, the ordering of the steps, the reasoning for the execution order, the reasoning behind the steps, derived values or invariants during execution, and other features. The recording is a joint responsibility of the provenance system and the application. The provenance system provides some functionality for the recording, but the information to be recorded is provided by the application itself. The provenance system cannot record information which is not provided by the application.

The management of the provenance information includes general management of the information provided to the provenance system as well as controlling the access to this information. Information management includes the creation, transformation, and destruction of information. Provenance information is created by submitting it to the provenance system. Provenance information can be transformed for example in order to make it more compact, or make it more descriptive for general users, or to improve its availability or longevity. Under given circumstances provenance information can be destroyed. All the above information management activities must be controlled and only actors with appropriate access rights can execute them. The provenance system supports the basic provenance information management functionalities, but complex and application specific provenance information management functionalities must be built for each application on top of these basic functionalities.

The utilisation of provenance information includes for example raw information retrieval from the provenance system, analysis of this information, or checking some derived features of the provenance information. Provenance information utilisation is under access rights control. The provenance system provides basic functionalities for provenance information utilisation. Application specific utilisation is built on top of these basic functionalities.

2.2 Environmental considerations

Grids typically integrate services running on different hardware and software platforms and implemented using different software technologies. The interoperability of these services is due to the standardised interfaces they implement. Being developed for grid applications, the provenance architecture should fit into such heterogeneous environments.

The provenance architecture to be developed is intended to be a new building block for grid enabling technologies. Therefore the primary direct users of it are expected to be IT professionals (i.e. programmers) who develop application specific pieces of software building on the functionality of the provenance architecture.

2.3 Relation to other systems

A provenance system will typically be embedded in a given application. We refer to Section 2.5 for details.

The interaction between the given application and the provenance system may have several forms, but all of them requires some modification of the existing application. In the simplest case a component of the provenance system is inserted between the components of the application. This provenance system component behaves like a proxy: from the application point of view the information flows through this component without modification, but in the meantime some information is recorded as provenance information. In a more complex case, the given application is modified so that whenever some provenance information is to be recorded, then the application interacts with the provenance system and sends to it the information to be recorded.

2.4 General constraints

The provenance architecture should be designed in a way that supports the ease of integration with existing software systems. The low costs of integration are important to make the introduction of a provenance architecture a reasonable choice versus the custom development of functionality or omitting provenance related features.

2.5 Model description

This section describes an abstract model that can be used to describe the major components of a provenance architecture. The simplicity of this model is intentional and due to the fact that the design of the architecture – including the design of its abstract model – has a dedicated phase in the workplan of the project and this is an important goal of the project itself. This document focuses on the interface of the provenance architecture rather than on its internal structure.

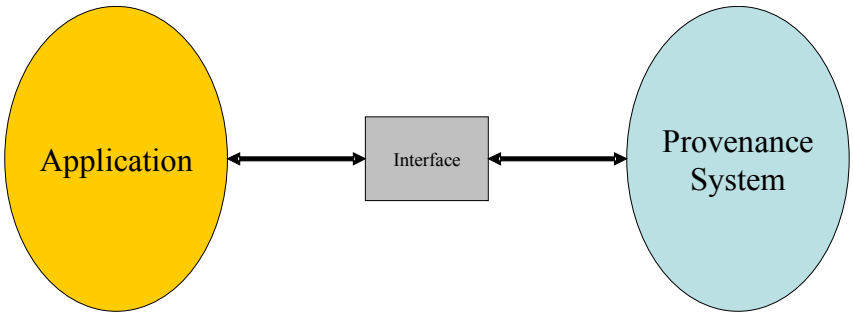

Figure 1: Abstract model for Provenance Architecture requirements specification

A provenance system is typically embedded in an application and should provide functionality to:

1. Record provenance data
2. Query the recorded provenance data
3. Manage the provenance system.

All interactions with a provenance system take place through an interface. The following are some examples of how an application (whether a software component or a user) may interact with the provenance system through its interface:

1. An application that generates provenance data to be recorded by the provenance system.
2. An application that analyses data in the provenance system.
3. A human user who wishes to access data in the provenance system.
4. An application that configures the provenance system.

Provenance data submitted to the provenance system's interface for recording purpose will be referred to as *provenance record*.

The purpose of the SRD is to take the single ellipse of the Provenance System in the logical model diagram and identify requirements to produce components of the architecture.

3 Specific Requirements

3.1 Functional requirements

Based on the study of several use cases [PASOA] identifies three important types of provenance information:

- *Interaction provenance:* A record of the interaction between services that took place, including the data that was passed between them.
- *Actor provenance:* Extra information from a service participating in a workflow at the time that the service was used.
- *Input provenance:* Given a piece of data, *X*, input provenance refers to the set of data used in the creation of *X*.

(Definitions of [PASOA] have been slightly modified in order to fit the broader context of our project. PASOA focuses on scientific applications and uses the term 'experiment'. This has been replaced here by the term 'workflow'.)

The third type, namely *input provenance* is derivable from *interaction provenance*, because data flow within a workflow is reconstructable if adequate interaction provenance information is available. Therefore we take the classification of *interaction provenance* and *actor provenance* for our further discussion.

This taxonomy fits well with the application scenarios explored by our examination. Provenance information to be recorded in the explored application scenarios can all be classified into one of these two provenance types:

- *Interaction provenance:*
    - **OTM:** information about service invocations and service responses in terms of *interacting entities* and *data exchange* (AR-1-7, TR-1-1-A-1, TR-1-1-A-2)
    - **TENT:** interactions between system components (AR-2-1, AR-2-2, AR-2-3, AR-2-4, TR-1-1-B-4, CR-5-4), input files for the TAU and the Aeroelastic module (TR-1-1-B-2, TR-1-1-B-3), data flow between system components (CR-5-4)
    - **eDiamond:** which processes were executed together with their input and output (TR-1-1-C-1), request and response messages (TR-1-1-C-5), other context information on the interactions (TR-1-1-C-6)
    - **HLSF:** the identity of the source (i.e. the submitting service) of each provenance data entry (TR-1-1-D-1)
    - **myGrid:** execution logs (TR-1-1-E-2)
    - **CombeChem:** identity of process executed (TR-1-1-F-1), intermediate data generated during an experiment* (TR-1-1-F-5) (*In a workflow this is the overall output of the individual processes except for the end result data set.)
    - **GENSS:** algorithmic information meaning: which service – performing what kind of algorithm – has been used (TR-1-1-G-2), parameter information (TR-1-1-G-3)
    - **traffic management application:** input parameters of simulation processes (TR-1-1-H-2)
    - **DataMiningGrid:** data about processes/algorithms used in a workflow of processing raw data in terms of *processing chain* (AR-7-1, TR-1-1-i-1)

- *Actor provenance:*
    - **OTM:** information about service invocations and service responses in terms of *time* and *user identities* (TR-1-1-A-1, TR-1-1-A-2), information state in service (TR-1-1-A-3), additional information on "side effect" type actions submitted by the applications (TR-1-1-A-4)
    - **TENT:** version information on used algorithms (TR-1-1-B-1)
    - **eDiamond:** user identities (TR-1-1-C-2), timing information (TR-1-1-C-2, TR-1-1-C-3), location and version information (TR-1-1-C-4), other context information on the interactions (TR-1-1-C-6)
    - **HLSF:** the date and time that each provenance data entry is created (TR-1-1-D-2), additional information relating to provenance information as provided by the application (TR-1-1-D-3)
    - **myGrid:** version information (algorithms, databases) (TR-1-1-E-1), execution time (TR-1-1-E-3)
    - **CombeChem:** identity of process version (TR-1-1-F-1), identity of operator (TR-1-1-F-2), timing information (TR-1-1-F-3), 'ambient conditions' like temperature (these parameters are known for the system during execution) (TR-1-1-F-4)
    - **GENSS:** calendrical information (TR-1-1-G-1), algorithmic information in terms of version information (TR-1-1-G-2)
    - **traffic management application:** configuration parameters of simulation processes (TR-1-1-H-1)
    - **DataMiningGrid:** data about processes/algorithms used in a workflow of processing raw data in terms of characteristics of applied processes and algorithms (AR-7-1, TR-1-1-i-1)

Requirements on provenance information that the provenance architecture should be able to record are summarised as the following software requirement:

SR-1-1: The provenance architecture should provide for the recording and querying of interaction and actor provenance.

Flags: essential, critical (OTM, TENT)
Source: AR-1-1, AR-1-2, AR-1-3, AR-1-5, AR-1-6, AR-1-7, AR-2-1, AR-2-2, AR-2-3, AR-2-4, AR-3-1, AR-3-2, AR-3-3, AR-5-1, AR-5-4, AR-5-5, AR-5-6, AR-5-7, AR-5-8, AR-5-9, AR-5-10, AR-5-12, AR-5-13, AR-6-2, AR-7-1, TR-1-1-A-1, TR-1-1-A-2, TR-1-1-A-3, TR-1-1-A-4, TR-1-1-B-1, TR-1-1-B-2, TR-1-1-B-3, TR-1-1-B-4, TR-1-1-C-1, TR-1-1-C-2, TR-1-1-C-3, TR-1-1-C-3, TR-1-1-C-5, TR-1-1-C-6, TR-1-1-D-1, TR-1-1-D-2, TR-1-1-D-3, TR-1-1-E-1, TR-1-1-E-2, TR-1-1-E-3, TR-1-1-F-1, TR-1-1-F-2, TR-1-1-F-3, TR-1-1-F-4, TR-1-1-F-5, TR-1-1-G-1, TR-1-1-G-2, TR-1-1-G-3, TR-1-1-H-1, TR-1-1-H-2, TR-1-1-i-1, CR-3-1, CR-3-2, CR-5-1, CR-5-4

Based on the user requirements identified in the URD, the focus of Provenance tools (part of WP6) in the first instance will be to help address requirements identified in the Organ Transplant Management (OTM) and the TENT application. Only technical level requirements from the URD are being addressed here.

A "tool" in this instance corresponds to either an Application Programming Interface (API) or a Human-Computer Interface (HCI) that may be used either directly by another service or by a human user. Development of the tools will assume that both of these applications (OTM and TENT) will generate provenance data that is recorded in one or more Provenance Stores directly through specialised software used within an application. Such a Provenance Store may be co-located to the user or accessible at a remote site. Each application must provide suitable (internal) software to

capture data that is relevant to a provenance query, and submit this using a recording interface to a Provenance Store. The tools will make use of a "Management" and "Submission" API published by the Provenance Store.

The provenance tools being developed as part of WP6 will support the following features within the logical architecture.

SR-1-2: The provenance architecture should allow the retrieval of a provenance trace from the Provenance Store. Either a complete trace or a subset may be retrieved.

Flags: essential
Source: AR-1-1, AR-1-2, AR-1-3, AR-1-5, AR-1-6, AR-1-7, AR-2-3, AR-2-4, AR-3-2, AR-3-3, AR-5-1, AR-5-2, AR-5-3, AR-5-4, AR-5-5, AR-5-6, AR-5-8, AR-5-10, AR-5-12, AR-5-13, AR-6-1, TR-1-1-A-1, TR-1-1-A-2, TR-1-1-A-3, TR-1-1-G-2, TR-1-1-G-3, TR-1-1-H-1, TR-1-1-H-2, TR-4-1, CR-3-1

SR-1-3: The provenance architecture should allow the back-up of a Provenance Store to be taken. This will generally include an archiving facility that allows data within a Provenance Store to be saved for future use.

Flags: essential
Source: TR-3-6

SR-1-4: The provenance architecture should allow comparisons to be made across Provenance Records within a Provenance Store with reference to particular data attributes within a Provenance Record.

Flags: essential
Source: AR-1-1, AR-1-6, AR-5-6, TR-1-1-C-1, TR-1-1-C-2, TR-1-1-C-3, TR-1-1-C-4, TR-1-1-C-5

SR-1-5: The provenance architecture should allow the results of a query to the Provenance Store to be captured for future use. A query in this context must be specified with reference to the structure of the Provenance Store.

Flags: essential
Source: TR-2-1-C, TR-2-1-D, TR-3-1, TR-3-3, TR-3-7

SR-1-6: The provenance architecture should allow a user to access a Provenance Record based on the time and date (calendrical information) at which the Record was stored.

Flags: desirable
Source: AR-3-3, AR-5-1, TR-1-1-D-2, T-R-1-1-F3, T-R-1-1-G1, CR-3-1

SR-1-7: The provenance architecture should allow a user to verify the contents of a Provenance Store against a specified set of rules. Verification in this context means that the contents of the Provenance Store meets the set of constraints expressed by the set of rules.

Flags: desirable
Source: AR-1-1, AR-1-7, AR-3-1, AR-4-1, AR-5-3, AR-5-4, AR-5-5, AR-6-1, AR-6-2, TR-1-1-C-6, TR-5-2

SR-1-8: The provenance architecture should 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.

Flags: essential
Source: TR-4-2

SR-1-9: The provenance architecture should allow capabilities provided by the tools to be accessible as an API. This is to allow such capabilities to be embedded within an existing application.

Flags: essential
Source: TR-4-3, TR-5-3, TR-6-1, TR-6-4-A, TR-6-4-B

SR-1-10: As part of the initialisation of the provenance recording process, the provenance architecture should allow a service or user to specify the identity of the Provenance Store to which data should be recorded.

Flags: essential
Source: Omer Rana

Additional functional requirements on the provenance architecture:

SR-1-11: The system should support the multiple storage of a provenance record, i.e. the system should provide a way to store copies of a provenance record in more than one repository.

Flags: desirable
Source: TR-3-1

SR-1-12: The system should support the recording of different provenance information views related to an event or an entity.

Flags: essential
Source: TR-3-2

SR-1-13: The provenance architecture should support the migration of provenance data among provenance stores.

Flags: essential
Source: TR-3-3

SR-1-14: The system should support the storage of recorded provenance data for an indefinite period of time.

Flags: essential
Source: TR-3-5-A, TR-3-5-B, TR-3-5-C

SR-1-15: The provenance architecture should support the storage of results of analysis and reasoning operations performed on the provenance data by tools that are not part of the generic architecture (3rd party tools on the application layer).

Flags: essential
Source: TR-4-3, TR-1-2

SR-1-16: The provenance architecture should provide support for maximum automation of the provenance recording mechanism.

Flags: desirable
Source: TR-5-1, CR-5-9

SR-1-17: The provenance architecture should be deployable as an integrated part of a system, as a service within the same administrative domain as the client system and as a 3rd (external) party operated service, too.

Flags: essential
Source: TR-5-3

SR-1-18: Client side components of the provenance architecture should not block an executing workflow if any provenance services are unavailable.

Flags: desirable
Source: TR-5-4

3.2 Performance requirements

SR-2-1: The additional execution overhead for an application recording provenance information should be kept to a minimum.

Flags: essential
Source: CR-1-1-A, CR-1-1-B, CR-1-1-C, CR-1-1-D

SR-2-2: Storage space requirements of the provenance architecture for provenance information recording should be kept at a reasonably low level.

Flags: essential
Source: CR-1-2-A, CR-1-2-B, CR-1-2-C

SR-2-3: The provenance architecture should guarantee reliable once-and-once-only delivery of provenance information to and from a provenance store.

Flags: desirable
Source: CR-2-1, CR-2-2

SR-2-4: The provenance architecture should be capable of handling large amounts of provenance data submitted frequently by user applications. The provenance architecture should not be the cause of any bottlenecks in the overall system due to the processing of provenance data.

Flags: essential, critical
Source: CR-5-3

3.3 Interface requirements

3.3.1 Application Programming Interface (API) requirements

SR-3-1-1: All of the functions of the provenance architecture should be accessible through its API so it can be used as an embedded component in a system.

Flags: essential, critical
Source: CR-5-5, CR-5-1, TR-6-4-B

SR-3-1-2: The provenance architecture should support a rich set of published, generic APIs that allow application specific analysis and reasoning tools to be built upon.

Flags: essential, critical
Source: TR-6-1, CR-5-1, CR-5-7

SR-3-1-3: The provenance architecture should provide a programmatic interface for the administration of the system.

Flags: essential, critical (eDiamond)
Source: TR-6-4-A, TR-6-4-B

SR-3-1-4: The provenance architecture should support an XML-based API format for provenance data.

Flags: desirable
Source: TR-2-1-B

3.3.2 Requirements on data export format

SR-3-2-1: Export formats for provenance data should be non-proprietary to allow tools and applications to be built without violating IPR rules. A format based on an existing data representation standard (with special focus on XML defined by XML Schema) would be highly preferred.

Flags: essential, critical
Source: TR-2-1-A, TR-2-1-C, TR-2-1-D

3.4 Operational requirements

SR-4-1: Provenance information displayed by the provenance architecture on a HCI should be updatable on user request.

Flags: essential
Source: TR-6-6-A

SR-4-2: HCIs presented by the provenance architecture for displaying the contents of a Provenance Store should support continuous monitoring, i.e. the displayed information should be updated automatically on every change as soon as possible.

Flags: essential
Source: TR-6-6-B

SR-4-3: The update frequency of provenance information displayed by the system on a HCI should be configurable based on policies.

Flags: essential
Source: TR-6-6-D

SR-4-4: Human-computer interfaces presented by the provenance tools should be designed to allow multilingual support.

Flags: essential
Source: TR-6-2

3.5 Documentation requirements

SR-5-1: Detailed documentation of the provenance architecture public interfaces should be produced both for application programming interfaces (APIs) and human-computer interfaces (HCIs).

Flags: essential
Source: TR-7-1

SR-5-2: A detailed description of the administrative interface of the provenance architecture should be produced.

Flags: essential
Source: TR-7-1

3.6 Security requirements

SR-6-1: The provenance architecture should have a configurable access control system over the resources it provides, with a granularity that is sufficient to protect these resources.

Flags: essential, critical (myGrid)
Source: CR-4-1

SR-6-2: The provenance architecture should allow both automated and manual determination of access control rights.

Flags: essential
Source: CR-4-2

SR-6-3: The provenance architecture should allow a service or user to request the level of security they wish to be associated with the recording process. The level of security can range from no security through encrypted data transfer to more complex security mechanisms.

Flags: essential
Source: Omer Rana

SR-6-4: The provenance architecture should provide a way to map access rights information of embedding systems into its security subsystem.

Flags: essential
Source: CR-4-3, TR-1-1-B-5, CR-4-4-A, CR-4-4-B

Note: For examples on user groups and their required access rights in two application scenarios refer to user requirements CR-4-4-A and CR-4-4-B.

SR-6-5: Security related procedures for accessing the provenance system should be subsumed under the existing security related procedures for the embedding system if possible, so that changes or additions to the existing procedures are minimized.

Flags: desirable
Source: CR-4-6, CR-4-5

SR-6-6: The provenance architecture should provide a mechanism for recording provenance data in an unmodifiable form and also ensuring that the party responsible for the recording process cannot deny having recorded that provenance data.

Flags: desirable
Source: CR-4-7, CR-3-2, AR-2-3, AR-7-2, AR-4-1, AR-4-2, AR-5-2, AR-5-10

SR-6-7: The provenance architecture should provide a mechanism for the authentic timestamping of provenance records. Authenticity should be guaranteed by the mechanism on a level that is enough even for the use in legal procedures.

Flags: desirable
Source: CR-3-2, AR-4-1

3.7 Other requirements

SR-7-1: The provenance architecture should have the properties of cost efficiency and robustness versus an in-application hand-engineered logging system.

Flags: essential, critical
Source: CR-5-2

SR-7-2: The provenance architecture should be loosely coupled and independent from the applications as much as possible. Integration costs for existing systems should be minimal, ideally existing system components should remain unaffected.

Flags: desirable, critical
Source: CR-5-6, CR-5-8

3.8 Unadopted user requirements

There are user requirements, which were rejected during the analysis phase or not directly transformed into any software requirement. Reasonings for such decisions are presented in this section for each case.

- **TR-3-4-A** and **TR-3-4-B**:
  Since provenance records are submitted by the user applications to the provenance architecture, the choice of the time instant for submission (immediate or delayed) depends on the application, as well as the kind of data transfer (submission of a single record or a batch submission of records). Therefore there is **no need for dedicated software requirements** in order to fulfill user requirements TR-3-4-A and TR-3-4-B.
- **AR-1-4**, **AR-1-8** and **AR-5-11**:
  The above mentioned user requirements are **rejected**, because they require application specific knowledge to interpret and process provenance information. These application specific functions should be implemented by application layer software components building upon the core provenance management functionality that the provenance architecture is intended to provide.
- **TR-6-3**:
  The provenance architecture is to provide core functionality, around which provenance management can be built for a given user application. The provenance architecture is not intended to support application specific analysis and reasoning operations by itself, therefore user requirement TR-6-3 is **rejected**.
- **TR-6-5-A**, **TR-6-5-B** and **TR-6-5-C**:
  The interpretation of application specific content of recorded provenance information is not to be supported by the provenance architecture itself, neither any visualisation functionalities that require such capabilities. For this reason, user requirements TR-6-5-A, TR-6-5-B and TR-6-5-C are **rejected**.
- **TR-6-6-C**:
  According to the planned model of provenance information recording, which is application initiated submission of provenance records to the provenance system, it is not expectable that the provenance system is aware of the individual execution sessions of user applications. For this reason, TR-6-6-C is **rejected**.

4 User Requirements vs Software Requirements Traceability matrix

1 Mapping of User Requirements to Software Requirements

1. Abstract level capability requirements

URD ID, flags, source	*Text of User Requirement*	*SRD identifier*
AR-1-1 *essential* OTM	The provenance system should support the following operation: Check a given set of decisions in a case against the established rules to ensure that it is conformant. These rules may or may not be automatically enforced by the transplant management software – however in the general case many of them will not be. This provenance question is a post-hoc check as to whether rules were followed. (asked by Transplant Authority, Families, 3rd parties)	SR-1-1, SR-1-2, SR-1-4, SR-1-7
AR-1-2 *essential* OTM	The provenance system should support the following operation: Derive a trace of the arguments, contributing factors and intermediate results which lead to a particular decision. (asked by Transplant Authority, Families, 3rd parties, Physicians)	SR-1-1, SR-1-2
AR-1-3 *essential* OTM	The provenance system should support the following operation: Derive aggregate information across many cases such as the percentage of incidents of a certain type, success rates by center, etc. (asked by Transplant Authority, researchers, physicians)	SR-1-1, SR-1-2
AR-1-4 *nice to have* OTM	As an advanced feature the provenance system could support the following operation: Truth maintenance for "next best candidate" or other dynamic information. Advanced functionality: meaning that the system could be used to keep up to date precalculated lists of recipients ready for an incident. This is a type of result which may need to be modified as underlying data changes. (asked by transplant system itself, physicians)	rejected *(reasoning provided in section 3.8)*
AR-1-5 *essential* OTM	The provenance system should support the following operation: Extraction of an entire case-trace: gather all the records related to one incident into a single case-file. (asked by physicians, families, patients)	SR-1-1, SR-1-2
AR-1-6 *essential* OTM	The provenance system should support the following operation: Identify all individual users related to an incident. (asked by physicians, Organ Transplant Authority, 3rd parties (legal challenges))	SR-1-1, SR-1-2, SR-1-4
AR-1-7 *essential* OTM	The provenance system should support the following operation: Provide a simulated walkthrough on service execution flow and verify this against template workflows and/or rules governing procedures (sophistication may vary). (asked by physicians, organ transplant authority, 3rd parties (legal challenges))	SR-1-1, SR-1-2, SR-1-7

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
AR-1-8 *nice to have* OTM	As an advanced feature the provenance system could support the following operation: Identify abstract derivation process of the result – based on some shared high level notions of the types of actions/content logged (e.g. having a standard view of what is an assertion, what is a decision etc.) and what follows what.	rejected *(reasoning provided in section 3.8)*
AR-2-1 *essential* TENT	The provenance architecture should be able to store all kinds of information that is needed to trace back the preceding process of data transformation within a workflow.	SR-1-1
AR-2-2 *essential* TENT/SikMa	Recorded provenance information should make it able to automatically restart workflows or parts of a workflow by the TENT system.	SR-1-1
AR-2-3 *essential* TENT/SikMa	The provenance architecture should be able to provide a trusted historical record of user access to produced data during a workflow (including intermediate data, result data and associated metadata as well), which can be used as evidence that the given data set has been accessed only by authorised users (as specified by the initiator of the workflow).	SR-1-1, SR-1-2, SR-6-6
AR-2-4 *essential* TENT/SikMa	The provenance architecture should make it able to identify unauthorised accesses to produced data during a workflow (including intermediate data, result data and associated metadata). Access rights are specified by the initiator of the workflow.	SR-1-1, SR-1-2,
AR-3-1 *desirable* eDiamond	For the protection of patient data and its usage, the provenance system should support the following operation: Report to show that an imposed policy has (or has not) been followed.	SR-1-1, SR-1-7
AR-3-2 *desirable* eDiamond	In order to ensure that doctors make correct diagnoses on correct data, the provenance system should support the following operation: List all occasions an image has been used and find the diagnosis produced by the processes applied to it.	SR-1-1, SR-1-2
AR-3-3 *desirable* eDiamond	For the diagnosis of process failure cases, the provenance system should support the following operation: Perform a network analysis of paths within a process. Identify bottlenecks in the process using elapsed timing information.	SR-1-1, SR-1-2, SR-1-6
AR-4-1 *desirable* HLSF	Proof of correct process management is required to be supported by the provenance architecture. Examples of policies and processes to be followed include the regulations defined by the Federal Drugs Administration in the US.	SR-1-7, SR-6-6, SR-6-7
AR-4-2 *desir*able HLSF	Proof that created data has not been tampered with is required to be supported by the provenance architecture.	SR-6-6
AR-5-1 *desirable* scientific apps	The provenance architecture should support the following operation: Accessing a historical record or aide memoire of work conducted.	SR-1-1, SR-1-2, SR-1-6

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
AR-5-2 *desirable* scientific apps	The provenance architecture should support the following operation: Proving that the experiment claimed to have been done was actually done.	SR-1-2, SR-6-6
AR-5-3 *desirable* scientific apps	The provenance architecture should support the following operation: Proving that the experiment done conformed to a required standard.	SR-1-2, SR-1-7
AR-5-4 *desirable* scientific apps	The provenance architecture should support the following operation: Checking that the experiment was performed correctly, and the services involved used correctly.	SR-1-1, SR-1-2, SR-1-7
AR-5-5 *desirable* scientific apps	The provenance architecture should support the following operation: Verifying that services used are working as they should be.	SR-1-1, SR-1-2, SR-1-7
AR-5-6 *desirable* scientific apps	The provenance architecture should support the following operation: Checking whether results were due to interesting features of the material being experimented on or nuances of the experiment performed.	SR-1-1, SR-1-2, SR-1-4
AR-5-7 *desirable* scientific apps	The provenance architecture should support the following operation: Linking together data and experiments by their provenance data to provide extra context to understanding those experiments.	SR-1-1
AR-5-8 *desirable* scientific apps	The provenance architecture should support the following operation: Tracing where data came from and the processes it had been through to reach its current form.	SR-1-1, SR-1-2
AR-5-9 *desirable* scientific apps	The provenance architecture should support the following operation: Tracing which source data was used to produce given result data and vice-versa.	SR-1-1
AR-5-10 *desirable* scientific apps	The provenance architecture should support the following operation: Providing the process information required for publishing an experiment's results.	SR-1-1, SR-1-2, SR-6-6
AR-5-11 *desirable* scientific apps	The provenance architecture should support the following operation: Deriving the higher-level processes that have been gone through to perform an experiment, so that they can be checked and re-used.	rejected *(reasoning provided in section 3.8)*
AR-5-12 *desirable* scientific apps	The provenance architecture should support the following operation: Allowing experiments to be re-enacted to check that services and/or data has not changed in a way which affects the results.	SR-1-1, SR-1-2

URD ID, flags, source	*Text of User Requirement*	*SRD identifier*
AR-5-13 *desirable* scientific apps	The provenance architecture should support the following operation: Determining the probable effectiveness of similar future experiments.	SR-1-1, SR-1-2
AR-6-1 *desirable* TMA	The provenance architecture should support the certification of the simulation workflow against a reference workflow description.	SR-1-2, SR-1-7
AR-6-2 *desirable* TMA	The provenance architecture should support the certification of input parameters against reference schemas for consistency.	SR-1-1, SR-1-7
AR-7-1 *desirable* DMG	The provenance architecture should support: Recording data about processes and/or algorithms used in a workflow of processing raw data.	SR-1-1
AR-7-2 *nice to have* DMG	The provenance architecture should support: Providing a trusted historical record of user access to confidential data.	SR-6-6

2. Capability requirements

URD ID, flags, source	*Text of User Requirement*	*SRD identifier*
TR-1-1-A-1 *essential* OTM	Recording of the following provenance information is required: Service invocation: Who accessed a particular service, when, with what input parameters (or a summary thereof) and on whose authority. 'Who' can refer to either a human or a service.	SR-1-1, SR-1-2
TR-1-1-A-2 *essential* OTM	Recording of the following provenance information is required: Service response: Who a service sent data messages to, in response to which invocation, the content of the response (or a summary thereof). 'Who' can refer to either a human or a service.	SR-1-1, SR-1-2
TR-1-1-A-3 *nice to have* OTM	Recording of the following provenance information would be useful: Information state: A summary of the information state in the service at the time a particular action is taken.	SR-1-1, SR-1-2
TR-1-1-A-4 *essential* OTM	In addition to the logging of message based activities the provenance service also needs to capture "side effect" type actions (e.g. those which may not directly lead to a response message): • Carrying out an action in the real world • Recording a decision or fact	SR-1-1
TR-1-1-B-1 *essential* TENT/ SikMa	For the output of the TAU module the version information of the involved TAU code should be recorded by the provenance system.	SR-1-1
TR-1-1-B-2 *essential* TENT/ SikMa	For a given output of the TAU module the processed input files should be recorded by the provenance system.	SR-1-1

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
TR-1-1-B-3 *essential* TENT/ SikMa	For a given output of the Aeroelastic Module the processed input files should be recorded by the provenance system.	SR-1-1
TR-1-1-B-4 *essential* TENT	Rejection of job submission by the TENT framework to cluster batch systems should be recorded by the provenance system, so this event can be recognised by TENT and it can restart the workflow or the appropriate modules.	SR-1-1
TR-1-1-B-5 *essential* TENT	The provenance architecture shall provide a way to map TENT access rights to ensure that no misuse of provenance data will take place.	SR-6-4
TR-1-1-C-1 *desirable* eDiamond	The following provenance information should be captured: Which process were executed together with their input and output.	SR-1-1, SR-1-4
TR-1-1-C-2 *desirable* eDiamond	The following provenance information should be captured: Who executed the process and when.	SR-1-1, SR-1-4
TR-1-1-C-3 *desirable* eDiamond	The following provenance information should be captured:: Processing elapsed times.	SR-1-1, SR-1-4
TR-1-1-C-4 *desirable* eDiamond	The following provenance information should be captured: Location and version information.	SR-1-1, SR-1-4
TR-1-1-C-5 *desirable* eDiamond	The following provenance information should be captured: Request and response messages.	SR-1-1, SR-1-4
TR-1-1-C-6 *desirable* eDiamond	The following provenance information should be captured: Other context information, which may include: • How was this service discovered? • policies established to invoke a service • information in SOAP message headers (security, reliability, transactionality etc.)	SR-1-1, SR-1-7
TR-1-1-D-1 *desirable* HLSF	The following provenance information is required to be recorded: The identity of the source of each provenance data entry.	SR-1-1
TR-1-1-D-2 *desirable* HLSF	The following provenance information is required to be recorded: The date and time that each provenance data entry is created.	SR-1-1, SR-1-6
TR-1-1-D-3 *desirable* HLSF	The provenance architecture should support storing the following information with provenance data: Attributes for each provenance data entry that may reference other objects stored either outside or inside the provenance repository.	SR-1-1
TR-1-1-E-1 *desirable* myGrid	The following provenance information should be recorded: Version information (algorithms, databases).	SR-1-1

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
TR-1-1-E-2 *desirable, critical* myGrid	The following provenance information should be recorded: Logs of what was done.	SR-1-1
TR-1-1-E-3 *desirable* myGrid	The following provenance information should be recorded: Estimates of quality of service metrics such as execution time.	SR-1-1
TR-1-1-F-1 *desirable* CombeChem	The following provenance information is required to be stored: Identity of process and version.	SR-1-1
TR-1-1-F-2 *desirable* CombeChem	The following provenance information is required to be stored: Identity of operator.	SR-1-1
TR-1-1-F-3 *desirable* CombeChem	The following provenance information is required to be stored: Time.	SR-1-1, SR-1-6
TR-1-1-F-4 *desirable* CombeChem	The recording of 'ambient conditions' of the experiments is required, like e.g. temperature. These parameters are known for the system during execution.	SR-1-1
TR-1-1-F-5 *desirable* CombeChem	The result and intermediate data of an experiment should be available and referenceable so that it can be linked to from papers and discovered for use in other experiments.	SR-1-1
TR-1-1-G-1 *desirable* GENSS	The following provenance information is required to be stored: Calendrical information.	SR-1-1, SR-1-6
TR-1-1-G-2 *desirable* GENSS	The following provenance information is required to be stored: Algorithmic information.	SR-1-1, SR-1-2
TR-1-1-G-3 *desirable* GENSS	The following provenance information is required to be stored: Parameter information.	SR-1-1, SR-1-2
TR-1-1-H-1 *desirable* TMA	The following information should be recorded: Configuration parameters of simulation processes.	SR-1-1, SR-1-2
TR-1-1-H-2 *desirable* TMA	The following information should be recorded: Input parameters of simulation processes.	SR-1-1, SR-1-2
TR-1-1-i-1 *desirable* DMG	The following information should be recorded: Data about processes/algorithms used in a workflow of processing raw data.	SR-1-1

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
TR-1-2 *essential* TENT, eDiamond, HLSF, myGrid, Combechem	The system should provide a way for the user to annotate the provenance data.	SR-1-15
TR-2-1-A *essential, critical* OTM	"Format must be a non-proprietary format which can in principle be used with another tool (to be built if necessary) without violating IPR rules. An open standard would be best."	SR-3-2-1
TR-2-1-B *desirable* HLSF	"The preferred API format for provenance data is the W3C Resource Description Framework (RDF). This is not mandatory, reports generated from the provenance data may be in other XML based formats and would have to conform to formats specified by external regulatory bodies. However RDF is preferred since this fits well with other standards in the Healthcare and Life Sciences arena, such as the Life Sciences Identifier (LSID)."	SR-3-1-4
TR-2-1-C *desirable* eDiamond	The export format of the provenance system should be XML defined by XML Schema.	SR-3-2-1, SR-1-5
TR-2-1-D *desirable* GENSS	The export format of the provenance system should be XML-based.	SR-3-2-1, SR-1-5
TR-3-1 *desirable* eDiamond, HLSF	The system should support the multiple storage of a provenance record, i.e. the system should provide a way to store copies of a provenance record in more than one repository.	SR-1-11, SR-1-5
TR-3-2 *essential* OTM, eDiamond, HLSF	The system should support the recording of different views on provenance information regarding to an event or an entity.	SR-1-12
TR-3-3 *essential* OTM, eDiamond, HLSF	The system should support the migration of provenance data among provenance repositories.	SR-1-13, SR-1-5
TR-3-4-A *essential* OTM, TENT, eDiamond, HLSF, DMG	On the fly recording of provenance data should be supported by the system.	no dedicated SR needed *(reasoning provided in section 3.8)*

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
TR-3-4-B *desirable* eDiamond, HLSF, DMG	Batch recording of provenance data should be supported by the system.	no dedicated SR needed *(reasoning provided in section 3.8)*
TR-3-5-A *essential* TENT	The system should support the storage of recorded provenance data for a complete simulation session. Runtime for a simulation session is between 1 minute and 1 month, its typical value is a few days.	SR-1-14
TR-3-5-B *desirable* eDiamond, HLSF	The system should support the storage of recorded provenance data for an indefinite period of time.	SR-1-14
TR-3-5-C *desirable* myGrid	The system should support the storage of recorded provenance data for 3-4 years.	SR-1-14
TR-3-6 *essential* OTM, TENT, eDiamond, HLSF	The system should be able to archive recorded provenance data.	SR-1-3
TR-3-7 *essential* OTM, TENT, myGrid, Combechem, GENSS, eDiamond, HLSF, DMG	The system should be able to export recorded provenance data for external usage.	SR-1-5
TR-4-1 *desirable* HLSF	It should be possible to query all of the data associated with a particular provenance entry, or return all of the provenance entries that have attributes matching a search criteria. *Note:* The term 'provenance entry' refers to the information written to a provenance service.	SR-1-2
TR-4-2 *essential* OTM, TENT, Combechem, myGrid, eDiamond, HLSF	The architecture should support the dynamic processing of provenance data, i.e. recorded provenance data should be instantly queriable even if a recording session (recording of interrelated provenance records belonging to e.g. the same workflow) is still in progress.	SR-1-8
TR-4-3 *desirable* eDiamond, HLSF, TMA, DMG	The provenance architecture should support the storage of results of analysis and reasoning operations performed on the provenance data by tools that are not part of the generic architecture (3rd party tools on the application layer).	SR-1-9, SR-1-15

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
TR-5-1 *desirable*	The provenance architecture should support for the maximum automation of the provenance recording mechanism.	SR-1-16
TR-5-2 *desirable, critical (eDiamond)* eDiamond, HLSF	Provenance handling should be policy-driven.	SR-1-7
TR-5-3 *essential* OTM, TENT, myGrid, Combechem, GENSS, eDiamond, HLSF, TMA, DMG, DILIGENT	The provenance architecture should be deployable as an integrated part of a system, as a service within the same administrative domain as the client system and as a 3rd (external) party operated service, too.	SR-1-17, SR-1-9
TR-5-4 *desirable* myGrid	Client side components of the provenance architecture should not block workflow if provenance services are unavailable and client explicitly expresses their wish to turn off provenance recording.	SR-1-18
TR-6-1 *essential, critical (eDiamond)* OTM, TENT, eDiamond, HLSF, TMA	The architecture should support a rich set of generic APIs that allow analysis and reasoning tools to be built upon.	SR-3-1-2, SR-1-9
TR-6-2 *essential* OTM	Human-computer interfaces presented by the system for analysis and reasoning should be designed to allow multilingual support.	SR-4-4
TR-6-3 *desirable* GENSS	Human-computer interfaces presented by the system for analysis and reasoning should be usable by a non computer scientist.	rejected *(reasoning provided in section 3.8)*
TR-6-4-A *essential, critical (eDiamond)* TENT, eDiamond, HLSF	The provenance architecture should provide a programmatic interface for the administration of the system.	SR-3-1-3, SR-1-9

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
TR-6-4-B *essential* TENT	The administrative interface of the provenance architecture should be able to be accessed and controlled through it's API. It has to be integratable into TENT or at least be accessible through the TENT system. Therefore some kind of user authentication may additionally be needed.	SR-3-1-1, SR-3-1-3, SR-1-9
TR-6-5-A *essential* OTM, HLSF, GENSS, TMA, DMG, DILIGENT	Provenance information should be trackable on human-computer interfaces presented by the system at set level (e.g. database table or spreadsheet).	rejected *(reasoning provided in section 3.8)*
TR-6-5-B *desirable* eDiamond, HLSF, DILIGENT	Provenance information should be trackable on human-computer interfaces presented by the system at individual data items (e.g. record in database or cell in spreadsheet).	rejected *(reasoning provided in section 3.8)*
TR-6-5-C *desirable* HLSF	The granularity of provenance information displayed by the system on a human-computer interface should be configurable based on policies.	rejected *(reasoning provided in section 3.8)*
TR-6-6-A *essential* OTM, eDiamond, GENSS	Provenance information displayed by the system on a HCI should be updatable on user request.	SR-4-1
TR-6-6-B *essential* OTM, GENSS, DMG, DILIGENT	HCIs presented by the provenance system for provenance monitoring should support continuous monitoring, i.e. the displayed information should be updated automatically on every change as soon as possible.	SR-4-2
TR-6-6-C *desirable* TMA	Provenance information displayed by the system on a HCI should be updated on each execution session of the monitored application.	rejected *(reasoning provided in section 3.8)*
TR-6-6-D *desirable* HLSF	The update frequency of provenance information displayed by the system on a HCI should be configurable based on policies.	SR-4-3

URD ID, flags, source	*Text of User Requirement*	*SRD identifier*
TR-7-1 *essential* TENT	There should exist different levels of system documentation, including the following: • a detailed API documentation for programmers who intend to integrate the provenance architecture into their systems, • a detailed description of the administrative interface of the system for system administrators, • a detailed description of other human-computer interfaces presented by the system e.g. for analysis and reasoning. Different audiences should be taken into account here including end-users as well, who want to use the provided tools as a stand-alone applications.	SR-5-1, SR-5-2

3. Constraint requirements

URD ID, flags, source	*Text of User Requirement*	*SRD identifier*
CR-1-1-A *essential* OTM	Provenance recording should not impede a human entering data in real time.	SR-2-1
CR-1-1-B *essential* TENT	Within TENT the execution overhead due to provenance recording has the upper constraint of not affecting the interaction with the system in a significant manner. In terms of the applications used in TENT workflows: due to typical execution times of e.g. the flow solver TAU, overhead has to be kept at minimum level.	SR-2-1
CR-1-1-C *desirable* myGrid	Provenance recording should not slow down workflow execution by significant magnitude. (Significant not quantified.)	SR-2-1
CR-1-1-D *desirable* eDiamond	Provenance recording should increase end-to-end elapsed execution time by no more that 5%.	SR-2-1
CR-1-2-A *essential* OTM	Recorded provenance data should not exceed 20% of overall system record data.	SR-2-2
CR-1-2-B *essential* TENT	There are the same constraints for storage overhead as for execution overhead (see CR-1-1-B), but less restricted.	SR-2-2
CR-1-2-C *desirable* eDiamond	Recorded provenance data should be less than 100 KBytes per patient image. (Patient images in eDiamond are usually around 32 MB.)	SR-2-2
CR-2-1 *desirable* eDiamond	Generated provenance data must not be lost.	SR-2-3

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
CR-2-2 *desirable* eDiamond	The provenance architecture should guarantee reliable once-and-once-only delivery (no copies) as much as technically possible and up to the measure it depends on the architecture itself and not on operating conditions.	SR-2-3
CR-3-1 *desirable* CombeChem	The provenance data should provide ability to ensure that appropriate regulations (such as those set by bodies like the Food and Drug Administration or the health and safety rules of a department) were adhered to.	SR-1-1, SR-1-2, SR-1-6
CR-3-2 *desirable* CombeChem	The provenance data should provide protection for intellectual property right issues, for example through the use of digital signatures and time stamping.	SR-6-6, SR-6-7, SR-1-1
CR-4-1 *essential, critical (myGrid)* OTM, myGrid, eDiamond, TMA, HLSF	The provenance architecture should have a configurable, fine-grained access control system over recorded provenance data.	SR-6-1
CR-4-2 *essential* OTM, eDiamond, DILIGENT	The provenance architecture should allow both automated and manual determination of access control rights on generated provenance data.	SR-6-2
CR-4-3 *essential* TENT	Access rights to the provenance system must be consistent with access rights to the rest of the TENT system. The provenance system should provide a way to map access rights information of TENT into its security subsystem. Access rights are stored in TENT in an LDAP server.	SR-6-4
CR-4-4-A *essential* TENT	The provenance architecture should be configurable in a way that assigns the following access rights to the given user groups: • User: Access to provenance data directly involved in the data manipulation process of the simulation (s)he has started and configured. • System designer: Access to secondary provenance data, which is the collection of all user provenance data and derivations from them for analysation and reasoning purposes. • System developer: Access to all kinds of provenance data. This especially includes data coming directly from the TENT core components. This data has to be visible only for this user group.	SR-6-4

URD ID, flags, source	***Text of User Requirement***	***SRD identifier***
CR-4-4-B *desirable* DILIGENT	The provenance architecture should be configurable in a way that assigns the following access rights to the given user groups: • Digital Library user: Access to the provenance data of his/her own initiated processes. • DILIGENT Administrator: General access. • Virtual Organisation Manager, Digital Library Manager, DILIGENT Resource Manager: Access to his/her own local provenance data.	SR-6-4
CR-4-5 *desirable* eDiamond	The security infrastructure of the provenance architecture should have single sign-on.	SR-6-5
CR-4-6 *desirable* eDiamond	The security infrastructure of the provenance architecture should be the same as the one of the application – particularly for any end-user clients. *Note:* In the case of eDiamond this infrastructure is Globus GSI integrated with OGSA-DAI. However it is not a stable version at the moment, because the National Health Service are changing their infrastructure and moving towards PKI.	SR-6-5
CR-4-7 *desirable* myGrid	The provenance architecture should provide a mechanism for recording adequate provenance data in an unmodifiable way to make results non-repudiable.	SR-6-6
CR-5-1 *essential, critical* OTM	The provenance architecture should have good application fit, meaning: meet the basic logging needs and have additional potential for more complex questions outlined in the scenario description.	SR-1-1, SR-3-1-1, SR-3-1-2
CR-5-2 *essential, critical* OTM	The provenance architecture should have the properties of cost efficiency and robustness versus an in-application hand-engineered logging system.	SR-7-1
CR-5-3 *essential, critical* TENT	The provenance system should be capable of handling huge amounts of provenance data coming in very frequently from the application itself. It should not create any bottlenecks disturbing the system.	SR-2-4
CR-5-4 *essential, critical* TENT	The provenance system should provide more and more detailed information about the different data and control flows taken place during workflow execution.	SR-1-1
CR-5-5 *essential, critical* TENT	On top of the API of the provenance system TENT must be able to access all its functions and provide them to the users through appropriate interfaces.	SR-3-1-1
CR-5-6 *desirable, critical* eDiamond	The provenance architecture should be loosely coupled and independent from the application so that current system is unaffected. Provenance can depend on the application, but the application should not depend on the provenance.	SR-7-2

URD ID, flags, source	*Text of User Requirement*	*SRD identifier*
CR-5-7 *desirable, critical* eDiamond	Tooling should be based on published APIs and not on hidden internal APIs	SR-3-1-2
CR-5-8 *desirable, critical* DILIGENT	The provenance architecture should support transparent integration and operation with the DILIGENT infrastructure.	SR-7-2
CR-5-9 *desirable, critical* DILIGENT	Provenance mechanisms should be handled at grid middleware level and/or as a third party service.	SR-1-16

2 Mapping of Software Requirements to User Requirements

SRD ID, flags	*Text of Software Requirement*	*Source (URD ID)*
Functional requirements		
SR-1-1 *essential, critical (OTM, TENT)*	The provenance architecture should provide for the recording and querying of interaction and actor provenance. Source (URD ID): AR-1-1, AR-1-2, AR-1-3, AR-1-5, AR-1-6, AR-1-7, AR-2-1, AR-2-2, AR-2-3, AR-2-4, AR-3-1, AR-3-2, AR-3-3, AR-5-1, AR-5-4, AR-5-5, AR-5-6, AR-5-7, AR-5-8, AR-5-9, AR-5-10, AR-5-12, AR-5-13, AR-6-2, AR-7-1, TR-1-1-A-1, TR-1-1-A-2, TR-1-1-A-3, TR-1-1-A-4, TR-1-1-B-1, TR-1-1-B-2, TR-1-1-B-3, TR-1-1-B-4, TR-1-1-C-1, TR-1-1-C-2, TR-1-1-C-3, TR-1-1-C-3, TR-1-1-C-5, TR-1-1-C-6, TR-1-1-D-1, TR-1-1-D-2, TR-1-1-D-3, TR-1-1-E-1, TR-1-1-E-2, TR-1-1-E-3, TR-1-1-F-1, TR-1-1-F-2, TR-1-1-F-3, TR-1-1-F-4, TR-1-1-F-5, TR-1-1-G-1, TR-1-1-G-2, TR-1-1-G-3, TR-1-1-H-1, TR-1-1-H-2, TR-1-1-i-1, CR-3-1, CR-3-2, CR-5-1, CR-5-4	*sources indicated in previous cell*
SR-1-2 *essential*	The provenance architecture should allow the retrieval of a provenance trace from the Provenance Store. Either a complete trace or a subset may be retrieved. Source (URD ID): AR-1-1, AR-1-2, AR-1-3, AR-1-5, AR-1-6, AR-1-7, AR-2-3, AR-2-4, AR-3-2, AR-3-3, AR-5-1, AR-5-2, AR-5-3, AR-5-4, AR-5-5, AR-5-6, AR-5-8, AR-5-10, AR-5-12, AR-5-13, AR-6-1, TR-1-1-A-1, TR-1-1-A-2, TR-1-1-A-3, TR-1-1-G-2, TR-1-1-G-3, TR-1-1-H-1, TR-1-1-H-2, TR-4-1, CR-3-1	*sources indicated in previous cell*

SRD ID, flags	*Text of Software Requirement*	*Source (URD ID)*
SR-1-3 *essential*	The provenance architecture should allow the back-up of a Provenance Store to be taken. This will generally include an archiving facility that allows data within a Provenace Store to be saved for future use.	TR-3-6
SR-1-4 *essential*	The provenance architecture should allow comparisons to be made across Provenance Records within a Provenance Store with reference to particular data attributes within a Provenance Record.	AR-1-1, AR-1-6, AR-5-6, TR-1-1-C-1, TR-1-1-C-2, TR-1-1-C-3, TR-1-1-C-4, TR-1-1-C-5
SR-1-5 *essential*	The provenance architecture should allow the results of a query to the Provenance Store to be captured for future use. A query in this context must be specified with reference to the structure of the Provenance Store.	TR-2-1-C, TR-2-1-D, TR-3-1, TR-3-3, TR-3-7
SR-1-6 *desirable*	The provenance architecture should allow a user to access a Provenance Record based on the time and date (calendrical information) at which the Record was stored.	AR-3-3, AR-5-1, TR-1-1-D-2, T-R-1-1-F3, T-R-1-1-G1, CR-3-1
SR-1-7 *desirable*	The provenance architecture should allow a user to verify the contents of a Provenance Store against a specified set of rules. Verification in this context means that the contents of the Provenance Store meets the set of constraints expressed by the set of rules.	AR-1-1, AR-1-7, AR-3-1, AR-4-1, AR-5-3, AR-5-4, AR-5-5, AR-6-1, AR-6-2, TR-1-1-C-6, TR-5-2
SR-1-8 *essential*	The provenance architecture should 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.	TR-4-2
SR-1-9 *essential*	SR-1-9: The provenance architecture should allow capabilities provided by the tools to be accessible as an API. This is to allow such capabilities to be embedded within an existing application.	TR-4-3, TR-5-3, TR-6-1, TR-6-4-A, TR-6-4-B
SR-1-10 *essential*	As part of the initialisation of the provenance recording process, the provenance architecture should allow a service or user to specify the identity of the Provenance Store to which data should be recorded.	Omer Rana

SRD ID, flags	***Text of Software Requirement***	***Source (URD ID)***
SR-1-11 *desirable*	The system should support the multiple storage of a provenance record, i.e. the system should provide a way to store copies of a provenance record in more than one repository.	TR-3-1
SR-1-12 *essential*	The system should support the recording of different provenance information views related to an event or an entity.	TR-3-2
SR-1-13 *essential*	The provenance architecture should support the migration of provenance data among provenance stores.	TR-3-3
SR-1-14 *essential*	The system should support the storage of recorded provenance data for an indefinite period of time.	TR-3-5-A, TR-3-5-B, TR-3-5-C
SR-1-15 *essential*	The provenance architecture should support the storage of results of analysis and reasoning operations performed on the provenance data by tools that are not part of the generic architecture (3rd party tools on the application layer).	TR-4-3, TR-1-2
SR-1-16 *desirable*	The provenance architecture should provide support for maximum automation of the provenance recording mechanism.	TR-5-1, CR-5-9
SR-1-17 *essential*	The provenance architecture should be deployable as an integrated part of a system, as a service within the same administrative domain as the client system and as a 3rd (external) party operated service, too.	TR-5-3
SR-1-18 *desirable*	Client side components of the provenance architecture should not block an executing workflow if any provenance services are unavailable.	TR-5-4
Performance requirements		
SR-2-1 *essential*	The additional execution overhead for an application recording provenance information should be kept to a minimum.	CR-1-1-A, CR-1-1-B, CR-1-1-C, CR-1-1-D
SR-2-2 *essential*	Storage space requirements of the provenance architecture for provenance information recording should be kept at a reasonably low level.	CR-1-2-A, CR-1-2-B, CR-1-2-C
SR-2-3 *desirable*	The provenance architecture should guarantee reliable once-and-once-only delivery of provenance information to and from a provenance store.	CR-2-1, CR-2-2
SR-2-4 *essential, critical*	The provenance architecture should be capable of handling large amounts of provenance data submitted frequently by user applications. The provenance architecture should not be the cause of any bottlenecks in the overall system due to the processing of provenance data.	CR-5-3
Interface requirements		
SR-3-1-1 *essential, critical*	All of the functions of the provenance architecture should be accessible through its API so it can be used as an embedded component in a system.	CR-5-5, CR-5-1, TR-6-4-B

SRD ID, flags	***Text of Software Requirement***	***Source (URD ID)***
SR-3-1-2 *essential, critical*	The provenance architecture should support a rich set of published, generic APIs that allow application specific analysis and reasoning tools to be built upon.	TR-6-1, CR-5-1, CR-5-7
SR-3-1-3 *essential, critical (eDiamond)*	The provenance architecture should provide a programmatic interface for the administration of the system.	TR-6-4-A, TR-6-4-B
SR-3-1-4 *desirable*	The provenance architecture should support an XML-based API format for provenance data.	TR-2-1-B
SR-3-2-1 *essential, critical*	Export formats for provenance data should be non-proprietary to allow tools and applications to be built without violating IPR rules. A format based on an existing data representation standard (with special focus on XML defined by XML Schema) would be highly preferred.	TR-2-1-A, TR-2-1-C, TR-2-1-D
Operational requirements		
SR-4-1 *essential*	Provenance information displayed by the provenance architecture on a HCI should be updatable on user request.	TR-6-6-A
SR-4-2 *essential*	HCIs presented by the provenance architecture for displaying the contents of a Provenance Store should support continuous monitoring, i.e. the displayed information should be updated automatically on every change as soon as possible.	TR-6-6-B
SR-4-3 *essential*	The update frequency of provenance information displayed by the system on a HCI should be configurable based on policies.	TR-6-6-D
SR-4-4 *essential*	Human-computer interfaces presented by the provenance tools should be designed to allow multilingual support.	TR-6-2
Documentation requirements		
SR-5-1 *essential*	Detailed documentation of the provenance architecture public interfaces should be produced both for application programming interfaces (APIs) and human-computer interfaces (HCIs).	TR-7-1
SR-5-2 *essential*	A detailed description of the administrative interface of the provenance architecture should be produced.	TR-7-1
Security requirements		
SR-6-1 *essential, critical (myGrid)*	The provenance architecture should have a configurable access control system over the resources it provides, with a granularity that is sufficient to protect these resources.	CR-4-1
SR-6-2 *essential*	The provenance architecture should allow both automated and manual determination of access control rights.	CR-4-2
SR-6-3 *essential*	The provenance architecture should allow a service or user to request the level of security they wish to be associated with the recording process. The level of security can range from no security through encrypted data transfer to more complex security mechanisms.	Omer Rana

SRD ID, flags	*Text of Software Requirement*	*Source (URD ID)*
SR-6-4 *essential*	The provenance architecture should provide a way to map access rights information of embedding systems into its security subsystem. *Note:* For examples on user groups and their required access rights in two application scenarios refer to user requirements CR-4-4-A and CR-4-4-B.	CR-4-3, TR-1-1-B-5, CR-4-4-A, CR-4-4-B
SR-6-5 *desirable*	Security related procedures for accessing the provenance system should be subsumed under the existing security related procedures for the embedding system if possible, so that changes or additions to the existing procedures are minimized.	CR-4-6, CR-4-5
SR-6-6 *desirable*	The provenance architecture should provide a mechanism for recording provenance data in an unmodifiable form and also ensuring that the party responsible for the recording process cannot deny having recorded that provenance data.	CR-4-7, CR-3-2, AR-2-3, AR-7-2, AR-4-1, AR-4-2, AR-5-2, AR-5-10
SR-6-7 *desirable*	The provenance architecture should provide a mechanism for the authentic timestamping of provenance records. Authenticity should be guaranteed by the mechanism on a level that is enough even for the use in legal procedures.	CR-3-2, AR-4-1
Other requirements		
SR-7-1 *essential, critical*	The provenance architecture should have the properties of cost efficiency and robustness versus an in-application hand-engineered logging system.	CR-5-2
SR-7-2 *desirable, critical*	The provenance architecture should be loosely coupled and independent from the applications as much as possible. Integration costs for existing systems should be minimal, ideally existing system components should remain unaffected.	CR-5-6, CR-5-8