Title:	*Specification of Mapping to Provenance Architecture, and Domain Specific Provenance Handling* *Work Package 7 ("Application 1: Aerospace Engineering")*
Author:	*Guy K. Kloss (DLR)*
Editor:	*Guy K. Kloss and Andreas Schreiber (DLR)*
Reviewers:	*Steven Willmott (UPC), Lazlo Varga (SZTAKI), Simon Miles (University of Southampton)*
Identifier:	*D7.1.1*
Type:	*Deliverable*
Version:	*1*
Date:	*20th September 2005*
Status:	*Restricted*

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 für Luft- und Raumfahrt e. V.	Germany
Universitat Politecnica de Catalunya	Spain
Magyar Tudomanyos Akademia Szamitastechnikai es Automatizalasi Kutato Intezet	Hungary

Document History

Remove this page upon release of the final version of the deliverable

Date	***Version***	***History***
2005-08-02	v0.1	Initial version
2005-08-09	v0.2	further enhancements and additions to sections
2005-09-13	v0.3	reworked version after reviews

Foreword

This document describes the relationship between the Aerospace Engineering Application (AEA) application adopted as a demonstration case in the Provenance project and the Provenance architecture. The mapping will be used as the basis for the implementation of the AEA demonstration system in months 12 – 24 of the project.

- Detailed descriptions of the AEA domain.
- High level mappings between the AEA and the Provenance architecture.

The primary audience of this document includes: Grid computing practitioners seeking to understand how Provenance technologies might be applied and, information technology practitioners in the computer aided engineering (CAE) domain interested in applying Provenance to their own systems.

Table of Contents

Table of Illustrations

List of Acronyms

Acronym	*Description*
ACL	Access Control List (for WebDAV: Access Control Protocol Extensions, RFC 3744)
AEA	Aerospace Engineering Application
API	Application Programming Interface
CFD	Computational Fluid Dynamics
CORBA	Common Object Request Broker Architecture [1]
CPU	Central Processing Unit (Processor in a computer)
FQDN	Fully Qualified Domain Name or IP address
GUI	Graphical User Interface
MPI	Message Passing Interface (system for computational parallelization in SMP systems)
SDK	Software Development Kit
SMP	Symmetrical Multi Processing
SOAP	Simple Object Access Protocol
SSH	Secure Shell
URI	Uniform Resource Identifier
WebDAV	Web based Distributed Authoring and Versioning (RFC 2518) [4]
WP	Work Package
XML	Extensible Markup Language

1 Introduction

This document describes the relationship between the Aerospace Engineering Application (AEA) application adopted as a demonstration case in the Provenance project and the Provenance architecture. The mapping will be used as the basis for the implementation of the AEA demonstration system in months 12 – 24 of the project.

The primary audience of this document includes:

- Grid computing practitioners seeking to understand how Provenance technologies might be applied, and
- Information technology practitioners in the scientific/engineering field interested in applying Provenance to their own integrational systems.

1.1 Purpose Of The Document

The purpose of the document is to act as an initial specification for a demonstration application: AEA over Provenance. A detailed demonstration specification taking into account these mappings will subsequently be developed for the application. Secondary goals include:

- Providing an example for others to build upon in future uses of Provenance technology.
- Generating feedback on the current Provenance architecture.

In order to achieve these goals, the document aims to:

- Detail the Aerospace Engineering Application (AEA) modeled in the project.
- Describe the architecture of the AEA system implementation being developed by various DLR institutes and deployed in the SikMa project (section 3 of the document).
- Describe a set of high level mappings applied to the application in order to guide detailed mapping to the Provenance architecture at implementation time (section 4 of the document).
- Provide a guide as to which items for the process documentation for provenance purposes will be stored, managed, and queried at what places in the AEA system (section 5 of the document).

1.2 Links Of Other Provenance Documents

The contents of this document are based on the following existing Provenance project documents:

- Requirements expressed for the AEA in the WP2 requirements deliverables D2.1.1 and D2.2.1.
- The Provenance project frozen architecture document D3.1.1, preliminary version 0.3.
- Project internal note on "Representing Provenance in the DLR Application" [6].

Further supporting documents are provided in the references section.

2 Application Overview

This chapter describes the aerospace engineering application scenario as it is currently in use. It covers a TENT application overview and the aerospace scenario implemented with it.

2.1 The TENT Application

Computer based simulation is used extensively in many scientific and technical applications, for instance in the development and design of aircraft and automobiles, or in forecasting the weather. In recent years, parallel computing has been very successful in reducing the computing time required for such modeling to acceptable levels. Nevertheless, the lack of interoperability and transparency of use of the tools belonging to a parallel and distributed simulation system still causes problems. With more complex and multi-disciplinary problems (see fig. 4) users tend to spend much of their time configuring and connecting the tools needed for one particular simulation than running the simulation itself. The fields of development are diverging between the "traditional" HPC programming paradigm, using Message Passing, and the success and effort made with distributed object computing. In addition, to the focus on pure parallel performance of a simulation, an environment for the process to run in is needed. TENT is an environment to overcome these problems and close the gap.

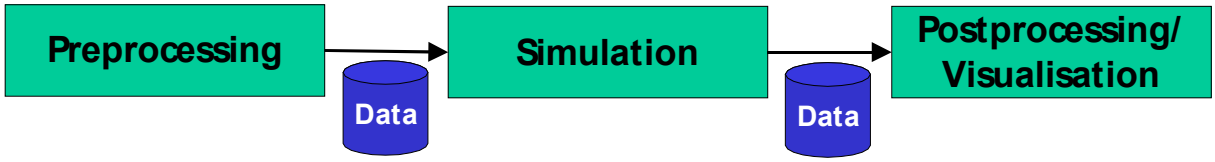

Figure 1: Process chain to form a simulation workflow.

In summary, TENT pursues the following goals:

- Provide an easy integration of existing applications and tools into one simulation environment.
- Combine the Message Passing paradigm and the Distributed Object Computing in one system.
- Provide interactive simulations from a single point of control.
- Be efficient and open transferring large amounts of simulation data.

The TENT base system is developed using the modern object oriented programming language Java, and native code application codes are integrated using Java, Python, or Fortran/C/C++ as necessary.

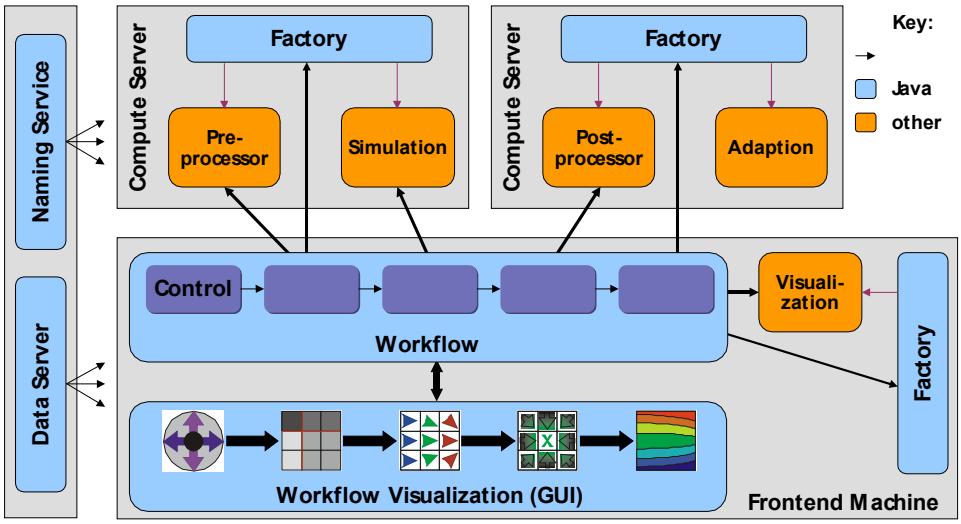

Figure 2: Architecture and deployment of TENT components and numeric components.

A complete TENT environment in use consists of the following parts:

- TENT Naming Service (TENT internal CORBA information service)
- One or more TENT Factories (to start TENT components)
- TENT Container (the instance where the workflow is controlled)
- TENT GUI (managing the workflow and steering the container)
- Data Server (WebDAV server for persistent data storage)

Application programs are integrated into the TENT environment as components using wrapper code containing a CORBA [1] interface. Customized wrappers are available for a growing number of standard applications including various simulations, optimizations, pre-/post-processors, and visualization codes. Other code can be easily integrated using Python scripts using the given TENT scripting API. A wrapper development library facilitates the integration of new applications and legacy codes without the need to modify them.

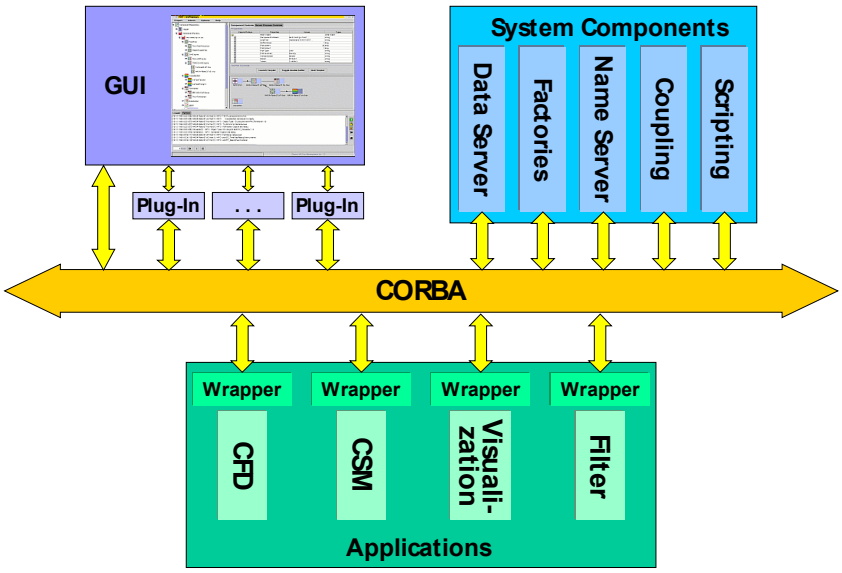

Figure 3: CORBA communication between TENT and numeric instances.

The GUI presents a list of available application components that the user can drag and drop to the workflow editor panel. Connecting lines are then inserted between components to define the control flow. The resulting workflow is treated as an object that can be executed, modified, stored, and retrieved. By setting parameters for individual components, or even modifying them during runtime, the user can perform parameter studies, while monitoring the progress of the computation via online visualization while still keeping the ability to control the process on line.

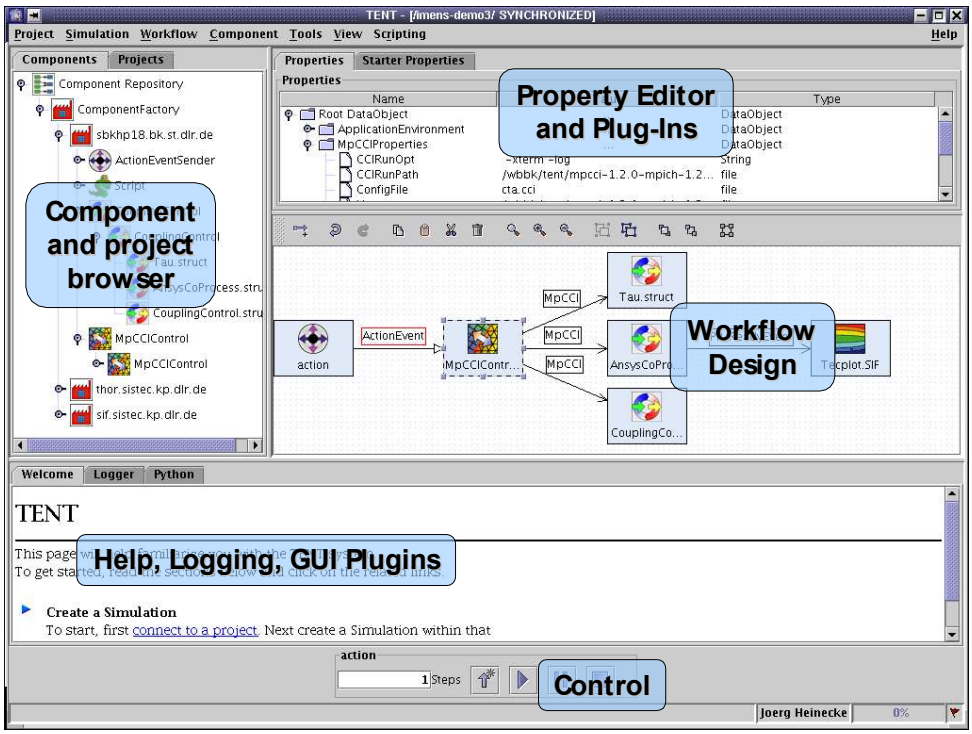

Figure 4: TENT graphical user interface.

A complete TENT environment consists of the following

- TENT Base System:
  - TENT Graphical User Interface
  - TENT Container (workflow container, control instance of a simulation workflow)
  - TENT Factories (components for launching components within a workflow)
  - TENT Naming Service (managing and announcing of other components' references)
- TENT Support System:
  - Data Server (WebDAV server)
  - optionally a Directory Server
    (LDAP server, for user authentication and information retrieval)
  - Utility Components (numeric and process monitoring, utility scripts, etc.)
- TENT Workflow Components:
  - TENT Components that wrap and manage functional code for the workflow
    (started by the TENT Factories and governed by the TENT Container),
    these can be numerical codes, visualization components, pre-/post-processors, scripts controlling workflow execution, etc.

2.2 Aerospace Application: Flight Maneuver Simulation

The DLR project SikMa (Simulation of Complex Maneuvers) is under the direction of the DLR Institute of Aerodynamics and Flow Technology. The objective is to develop an interactive simulation environment for the simulation of a freely flying, fully configured, elastic warplane. To implement the simulation, the aerodynamic, flight mechanical, and aeroelastic equation systems will be calculated at every step, for a time-accurate coupling of aerodynamics, flight mechanics, and aero-elasticity.

The results from the SikMa project will drive the idea of TENT, as a uniform simulation environment for the DLR, and make a substantial contribution to the DLR core area of the virtual aircraft. Along with the development of the interactive integration environment and the computing processes and coupling algorithms, the SikMa project will also do wind-tunnel testing with different models in different maneuver scenarios, in order to obtain a comprehensive data set for the verification and validation of the overall simulation environment.

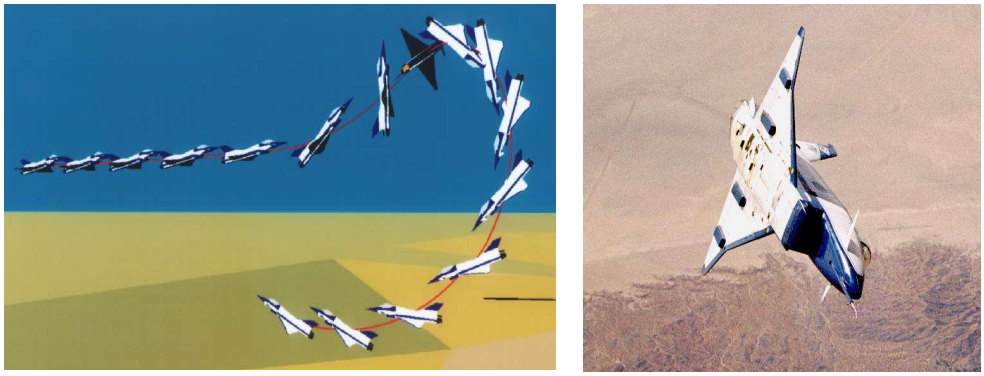

Figure 5: Visualized flight maneuver simulation (left) of the aircraft X-31 (right).

As a test case the experimental military aircraft X-31 is used. A simplified model for wind tunnel experiments has been created which in turn is validated via numeric simulations to keep a close relation of results obtained from those sources.

3 Application System Architecture

As stated in the previous chapter the simulation of a freely flying, fully configured, elastic warplane requires the cooperation of various numeric compounds at a number of time steps. The computations needed are tied together in the simulation integration system TENT to form a governing "shell" for the simulation. For the simulation of flight maneuvers a coupled simulation of the following partial tasks is required:

- Aero-dynamics
- Aero-elasticity
- Flight mechanics

Additionally the logical coupling mechanisms between the three numerical tasks, the iteration through the numerous time steps, and the necessary pre-/post-processing steps for data preparation, conversion, etc. are needed.

In this document the term *component* appears in connection with the TENT system. This specifies a piece of software that implements certain CORBA interfaces to integrate in a well specified way with TENT.

3.1 Logical Coupling Of Simulation Task

In figure 6 the simulation workflow as represented within the TENT system is shown. The first ("Action") component is the point of entry for TENT to start, stop, or continue the simulation. The coupling manager implements all the logic needed to tie the before mentioned numeric components together (TAU, SIMULA, and the aeroelastic module). It governs the order in which the three components will be triggered and it initiates the transfer of required data to and from those components.

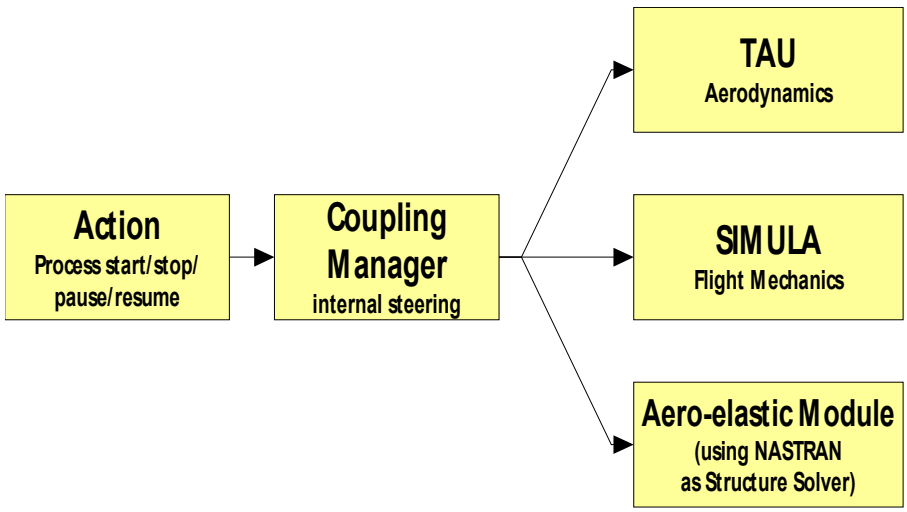

Figure 6: SikMa coupling scenario.

The next workflow diagram highlights the internal logical workflow as guided by the above mentioned component "Coupling Manager". The three "worker components" from above are represented here as "Flight Mechanics" on the top left, "NASTRAN" for the aeroelastic part in the center shaded area, and finally the complete dark gray box for the computational fluid dynamics (CFD) on the right utilizing the TAU code(s). So this workflow diagram in figure 7 represents the complete numeric process which has been separated and implemented for the TENT system as described above. Due to the fact that the DLR developed CFD code TAU is subject to constant change and improvement the schematic is in too much (potentially non-current) detail to be described here. Basically the whole dark grey box labeled "CFD" matches the component "TAU" in figure 6,

whereas the boxes "Flight Mechanics" and "NASTRAN" in the lighter grey shaded areas are matched by the components "SIMULA" and "Aero-Elastic"

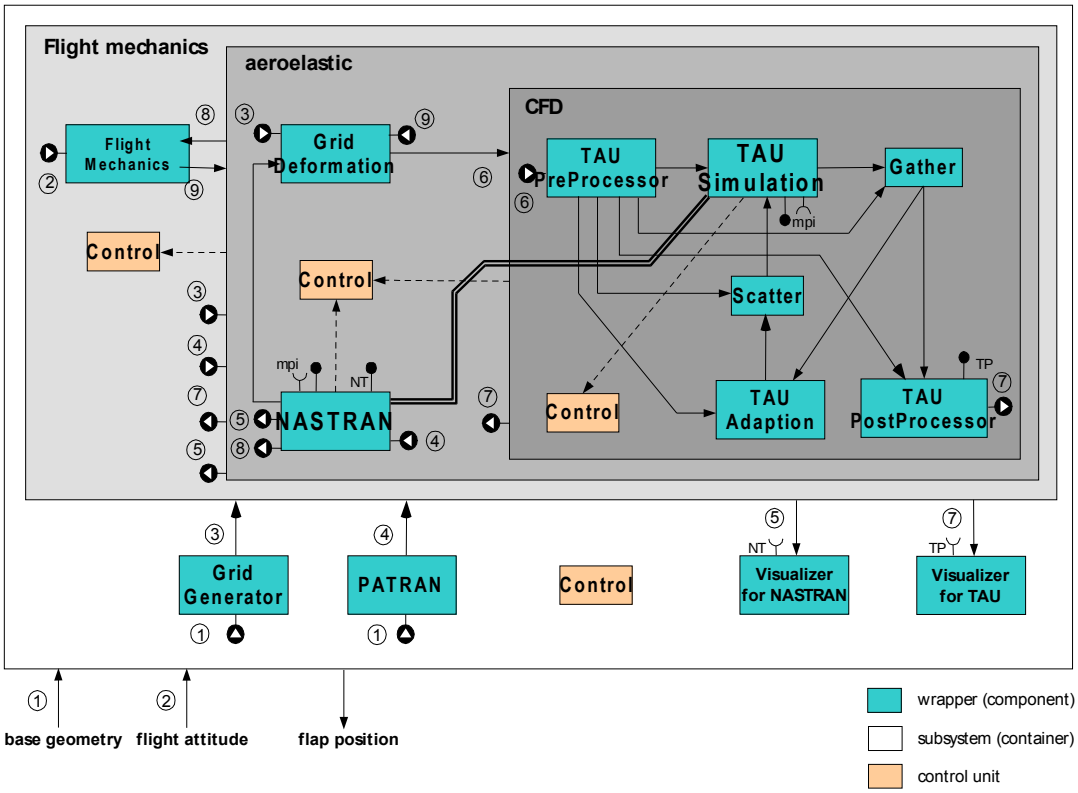

Figure 7: Detailed aeroelastic simulation workflow.

3.2 Deployment Of SikMa Scenario

Figure 8 shows the deployment of the simulation workflow as it is used for the SikMa project. To simplify the deployment scheme the scenario has been reduced by one simulation component (aero-elastics). Additionally only data and control streams relevant for the workflow have been added, supporting services (TENT Naming Service and Factories) and workflow persistence on the data server have been omitted.

An engineer's desktop machine hosts the TENT GUI in order to define and perform a simulation. Upon activation of a workflow the components contained within the workflow are started on the computational remote server. Then an activated workflow can be started (using the "Action" component). The start action triggers the "Coupling Control" component to start to govern the numerical simulation by triggering the numerical components in turn as needed.

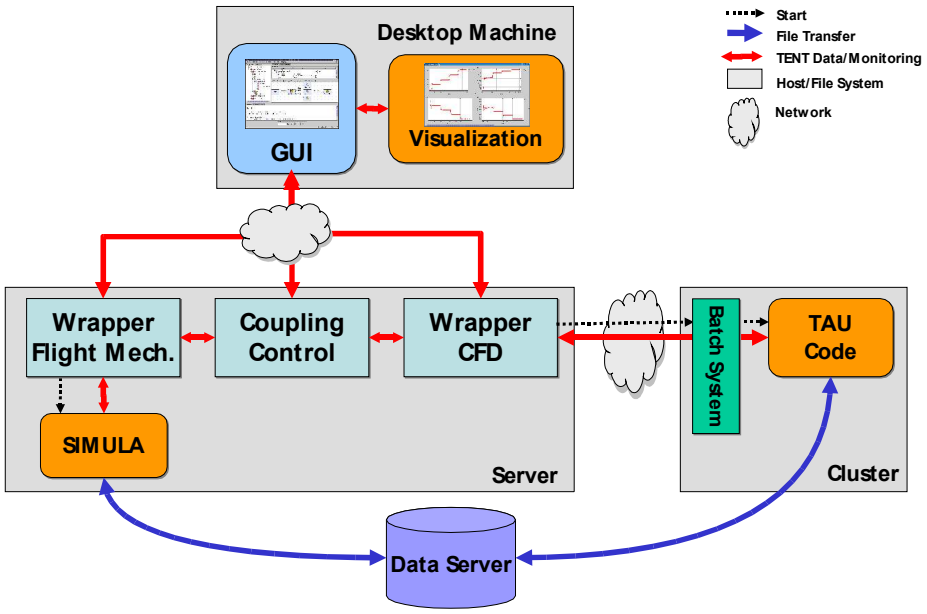

Figure 8: Deployment scenario for SikMa project.

If for example a flight mechanics computation is needed, a corresponding signal ("event") is sent to the "Flight Mechanics" component's wrapper code. This will start the numeric native application code SIMULA on the computational server and perform that part of the simulations as needed. Input and result data is pulled from and written back to a centrally accessible data server (WebDAV server). Upon completion of the computation the flight mechanics wrapper will signal back to the "Coupling Control".

The launch of a computational fluid dynamics (CFD) computation works similarly, only with the difference that the computational host for the native TAU code itself is not located on the same host running the wrapper component. The wrapper component behaves identically towards the TENT system, but internally it delegates the actual processing to an external batch system or Globus [2] for processing on a computational cluster or Grid. Usually this is a cluster located at the High Performance Computing Center in Stuttgart fed by the batch system PBS [3]. Data access again is handled via the same data server, so the output of one component may be used for input of another component.

Data transfer and communication between the TENT components, native codes, batch systems, etc. can be handled in various ways:

- The signaling process between the TENT components through events can contain some amounts of prior specified types of data.
- An event between TENT components can trigger a direct data transfer using GridFTP.
- Data may be made available globally to all involved components via the central repository on the data server.
- In case one of the computational hosts (e. g. accessed through a batch system or Globus) does not have direct access to the data server or the submitting host of the wrapper has got to employ circumventive measures. Cases like these require the component wrapper to communicate through secure (SSH) tunnels and/or transfer data prior using other means as GridFTP for example. In these cases the wrapper has got to "know" about this and perform the transfer actively.

3.3 Multi User Scenario

Multiple users may use the same SikMa environment to compute concurrently from different desktop systems. This means, that on the one hand the above mentioned simulation hosts need to be able to

share their resources without conflicts as well as the data repository on the data server. On the other hand access to the distributed computational resources need to be accessed in a way that they will not interfere with each other.

3.4 Data Storage

All data of the SikMa scenario is stored in the WebDAV [4] data server. There workflow definitions and component properties/configurations reside. Additional application data resides on local scratch disks on the hosts local to the execution of the individual integrated applications. This storage is not to be considered as persistent and is present only for the meantime of either the simulation itself or just a single computation for a simulation's (time) step. Scratch data will be discarded after completion of the component's computational duty. If it is to be stored for further reference of the simulation's history or decision making it will be transferred by the component wrapper to the centralized data server.

As no abstract storage layer, which transparently takes care of migrating data to logical file names, is used, the transfers of data need to be tracked in the provenance store.

Additionally some logging information is generated by the different components. This logging is performed by Log4j [5] and displayed within the GUI it's own purpose of giving the user feedback about the state of the running application and simulation. To keep the user well informed it may as well contain information to some extent that can be evaluated to track part of the provenance of the simulation as well. But on the one hand this information is by no means complete in order to keep thorough provenance records on the simulation as desired. On the other hand (depending on the Log4j [5] configuration) the given information is very volatile and may only live in the instance of the currently running GUI, or a temporary local log file that gets overwritten again after some time.

Due to the fact that the whole simulation system is constructed from an inhomogeneous collection of computational hosts and various methods for storage, it is impossible to establish a single system for data storage security restrictions. Depending on the platform hosting a workflow component local file access may be extremely different in protection, and accessibility to that host may be just as variable. The data server (WebDAV server) itself in the setup is restricted to authenticated users, and access may be authorized depending on the WebDAV implementation using access control lists.

As sensible data may be handled throughout the whole process, the environment containing the involved hosts will need to provide the minimum protection. Additional protection is provided by the services collecting and distributing data centrally, as here not just atomic items of information, but chained sets of data would be accessible.

4 Logical Application – Provenance Mapping

This section provides a high level view of how the Aerospace Engineering Application (AEA) maps to the provenance architecture. Description is in terms of general principles, identification of components, mapping rules etc.

4.1 Mapping To The SikMa Scenario

In the SikMa project the AEA's use of the provenance infrastructure is based upon the following main principles:

- All major Web Services components in the scenario (see submission clients in 4.2.2.1 and management/query client 4.2.2.2) will connect to one central provenance store. One physical data server (see 3.4) needs to be accessible, so by infrastructure it is not difficult to deploy and access a single provenance store. The need for physical distribution is not expected to be necessary considering the estimated amount of provenance data.
- Each TENT component (at least TENT GUI, TENT factories, and the component wrappers, as denoted by the light blue boxes in figure 8) need to submit to the provenance store as distributed instances.
- Each TENT component (see above) represents an individual actor towards the provenance Web Service.
- (Extensive) data sets are referenced from the provenance store using URIs to the data server (WebDAV server, see 3.4) or similar pointers.

4.2 Provenance And Application Data Mapping

While the precise objectives of storing certain items of data is dependent on the queries finally supported (see later section) the general aims of the data stored in the system is:

- Provide a coherent way of tracking/locating interim steps along the path related to a simulation across all p-assertions.
- Provide a link of the step in a specific simulation to its interim results, if available.
- Provide information or pointers to information that led to results, decisions, or outcomes without needing to examine complex result data sets.

In the previous chapter the data storage concept of the TENT system has been discussed. It is important to note that all the data that has been referred there, is data stored by the application itself for the purposes of the application (no provenance involved). All data to be archived will be defined during the implementation of the simulation process. If interim results is to be archived as well, mechanisms need to be implemented as well. All this data will be transferred to the data (WebDAV) server for storage, and can be referenced by URIs contained in the p-assertions.

The following presentation relates to the provenance data which might be stored at runtime in addition to this data.

4.2.1 Requirements For Provenance Data

The provenance service must provide storage for up to a few simulations (at least one complete simulation session). The typical runtime for one complete simulation run is approximately one week..

Access control to the stored p-assertions should be handled with some type of access control. This is especially important for military applications. All simulations are processed within a secured

environment, and the p-assertions contain additional information tying the already stored and secured simulation data together for further inspection. The level of security for the data can be considered to be similar to that of the data server. For practical reasons it should be considered to match an access control list (ACL) system as used by (or similar to) the access control system used for the data server[1]. Anonymization or encoding of user information in this context is not an issue. As some specific users are required for running certain processes, the information on the user executing a process is essential information on the computational environment.

It is not desired to keep provenance information within the provenance store over very long times. It will be sufficient to keep the provenance information of the currently active projects within the store for performance, reference, and analysis reasons. But for long term documentation[2] and re-analysis a long term persistence is required. This should be achieved by the ability to export/import p-assertions (e. g. to the file system) for further storage in the given data management system.

This requirement can be deduced from the fact that the present computational environment to be used for provenance analysis should sustain data integrity between the provenance store and the linked information on the data server. The volume of simulation data on this type of problems can become very large. This means that currently unnecessary data will be migrated to other types of (off line) storage systems. These can be retrieved upon further need back into the system later on. For data integrity reasons the provenance data should be exportable into files to be stored and movable along with the simulation's data. For further data inspection and provenance queries of course this data will need to be reimported into the on-line accessible data server and provenance store.

4.2.2 Provenance Store Access

The provenance store is to be accessed in three different ways: For submission through the running TENT components during a simulation, for provenance queries through a query client, and finally for the management of contained p-assertions through a management client.

The distributed simulation environment will use only one provenance store. This will need to be accessible via Web Services from all places where TENT components are running. This does not need to include the hosts executing the integrated native codes (e. g. on a batch system), as the relevant information is present within the TENT components, and the interface to the provenance store can be implemented in the TENT components in contrast to the native codes.

A provenance query client or a provenance management client presents to the user an interface to access the provenance store. So this piece of software will be running local to a graphical user interface as the TENT GUI, into which it should be embeddable as well. As by deployment plan the TENT GUI has got access to the provenance store, the management clients will, too.

The major components of the aerospace engineering application are defined in terms of the architectural notions defined in the Provenance Logical Architecture D3.1.1. Figure 9 shows the mapping to the AEA scenario. The provenance roles, libraries, and interfaces in it are described in more detail in the following.

Provenance Roles

Application Actors
(responsible for carrying out the application’s business logic)

> All TENT components (GUI, factories, container, workflow components) and humans (as the application's user interface is used) are application actors towards the provenance system.

Provenance Stores
(responsible for making persistence, managing, and providing controlled access to recorded p-assertions)

1 Access Control Protocol Extensions to WebDAV (RFC 3744)
2 German agreement on "good scientific practice" requires ten years of reproducibility of results that lead to a publication. Aircraft and other regulations often lead to the obligation to document for 25 years or more.

All actors are accessing one centrally available provenance store.

Asserting Actors
(actors that create p-assertions about an execution)

All application actors except for the querying and managing actors are asserting actors.

Recording Actors
(actors that submit p-assertions to a provenance store for recording)

All asserting actors are recording actors. All recording actors by default record to the central provenance store.

Querying Actors
(actors that issue provenance queries to a provenance store)

Provenance store queries are not expected at application execution time but occur as a separate process. Queries are carried out by designated query actors.

Managing Actors
(actors that interact with the provenance store for management purposes)

Provenance management interactions are not expected at application execution time but occur as a separate process. Management actions are carried out by designated management actors which may be the identical to the query actors.

Libraries and Interfaces

Actor Side Libraries

Actor side libraries for provenance recording are embedded in all application actors. They themselves may employ the tool suit developed in WP6. The use of tools is primarily true for management and querying in the designated additional actors.

P-header
(provenance related context information, sent along with the interaction's message)

A P-header is included with every application message interchanged between any two application actors. This p-header may contain information as e. g. interaction IDs (4.2.2.1.1) or tracers (4.2.2.1.3).

Recording Interface

Recording Interfaces are used by recording actors only.

Query interfaces

Query interfaces will be used by specialized query and management actors only – not by general application actors.

Management interface

Management interfaces will be used by specialized query and management actors only – not by general application actors.

Processing Services and Presentation User Interfaces

These interfaces are instantiated by the management and query actors only.

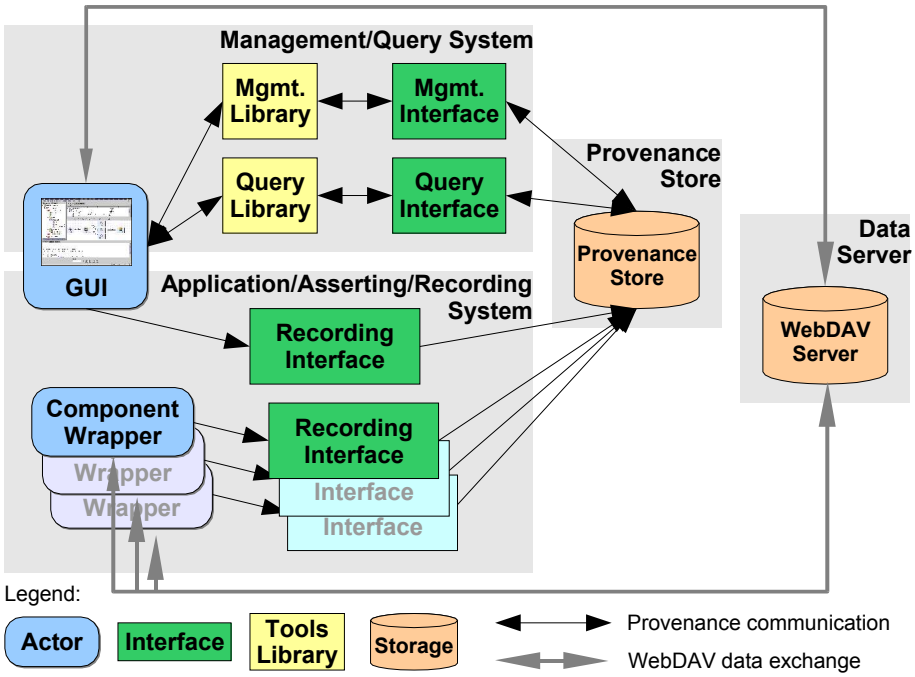

Figure 9: Mapping logical architecture to AEA scenario.

4.2.2.1 Provenance Submission Client

The provenance submission client will be an implemented Web Services interface, made available to TENT components through the TENT SDK's (Software Development Kit) API. Information to the p-structure needs to be submitted from the TENT GUI (basic information on the simulation process, change of simulation parameters, etc.), TENT factories (information on hosts, executables, etc.), and wrapper components (detailed information on the level of the simulation component).

The structure of TENT fits well with the proposed structure for querying the p-structure. The p-structure documents past processes in terms of the interactions (message exchanges) between actors, the causal relation between interactions, and the state of actors at the time of interaction. Here we try to identify how the application is mapped to the p-structure.

It is intended to browse the causal chain that led to a piece of data. It is also necessary to distinguish high-level steps in an executed workflow from finer-grained steps, so that the latter are only retrieved on demand. This can be achieved using several means: interaction IDs (4.2.2.1.1), tracers (4.2.2.1.3), or interaction labels (4.2.2.1.4) as described below.

4.2.2.1.1 Interactions

The p-structure allows the recording of interactions between actors, i. e. the data flow between services. Each interaction is given an *interaction ID* by the actor sending the message. This interaction ID must be passed from the sender to the receiver of the application message so that it can record causal links to the subsequent interactions it is involved in. This will be an additional attribute in the TENT-internal CORBA event sending information between actors. With the knowledge of the interaction ID the receiver itself can submit p-assertions with the same interaction ID, so there may be more than one message in a single interaction.

This should be clarified by an example: A result data file should be transferred to a post-processing receiving component. So the sending component sends a "File Transferrer Event" containing the source of the file to be transferred to the receiving component. This action will be recorded by a p-assertion be the sending component accompanied by an interaction ID. The receiving component requests the transfer of the data file to the component's local scratch disk. This action, as well, would be recorded with a p-assertion containing the same interaction ID, but this time it will be submitted by the receiving component. After the completed transfer a message containing the successful transfer with information about the transfer (e. g. transfer time, transferred bytes, communicating hosts) again

with the same interaction ID will be submitted. So from a user's point of view this data transfer is a single step in a process chain, which from the point of view of the system comprises of three individual p-assertions. These three messages can be grouped or related to each other for the user's perspective by the interaction ID.

It would be useful to provide a way to generate unique IDs in a provenance store client-side library.

4.2.2.1.2 Causal Relations

In order to track back along forks of the causal chain (workflow) that produced some data, each actor records the causal relation between a message it is about to send, and a message it has received ("I'm sending this message because I received that one"). These causal relations will be available in the p-structure at query time. They can be established by the actor by its knowledge of the workflow, or by means of using the interaction IDs, tracers, or interaction labels for "orientation".

4.2.2.1.3 Workflows And High-Level Actions

When an actor executes a process, which involves sending messages to other actors, it can add a *tracer* to its messages. A tracer is a unique ID that is propagated by all actors from the messages they receive to those that it sends as a consequence (so one interaction may contain a set of tracers that are being propagated).

The actor initiating a workflow introduces a tracer to its initiation message. Therefore, all interactions in the workflow will include that tracer. The actor performing a high-level action will introduce a tracer into its messages, so that all interactions performed within that high-level action will include that tracer.

By querying for tracers, the user interface can delimit all documentation relating to one workflow or to one high-level activity within a workflow, and allow browsing accordingly.

Tracers will also be sent in CORBA event attributes.

4.2.2.1.4 User-Supplied Interaction Labels

It would be useful for a user to supply an intuitive name for all interactions between particular actors in a given workflow execution. The quantity of these interactions may not be known beforehand due to loops in the workflow. The label name could then be used in querying.

Those labels would be defined during the implementation of the wrapper components and the setup of the simulation workflow before the workflow is run. The names are given from the point of view of a simulation's engineer, *not* from the point of view of a software engineer. It would be usually something descriptive regarding the process to filter for specific steps within the workflow. A typical flight mechanics simulation would contain many individual runs of the CFD and aero-elastic code before the next flight mechanics calculation is triggered. To easily filter for the relevant steps of that specific process stage over a complete simulation a fixed label "maneuverStep" could be defined. The p-assertions regarding these fraction of the complete computation would be easily identifiable via queries by the user. The actor can then include the name as an actor state p-assertion (there is a place in the actor state schema for meta-data about interactions).

4.2.2.2 Provenance Query And Management Client

The query and management client can be combined in one application. This client should be designed as a stand-alone application. If implemented in Java as a Java Bean it would be directly embeddable within the TENT GUI, otherwise it would be callable from within it for convenience. As stated in WP6 ("Tools and Setup") the tool suite may contain both an API to allow other applications to interact with the provenance store, or it may provide a GUI.

The purpose of the management client is to keep the amount of information within the p-structure within bounds for practical reasons. The problem faced which needs to be addressed here is not the size or number of p-assertions that are being handled by the provenance store, but rather the amount of data on the data server referenced by the p-structure. This data needs to be movable to off-line

media or other storage hosts. In order to keep the integrity of the whole system it should be possible to export the corresponding portions of the provenance store to a file that would be stored along with the data. The identification of those p-assertions would probably be handled through the highest level tracer introduced to a simulation set at the start of it. It would be available throughout all subsequent p-assertions belonging to that simulation. To complete the scenario p-assertions also will need to be deletable from the p-structure as well as importable to the p-structure from a file.

For browsing and analyzing the result sets of a provenance query on the p-structure the user interface needs to be intuitive and easy to use. A way for presenting those result sets still has got to be identified, either graphically, in a tree structure, or textually. The result browser may even give the choice of different perspectives to navigate through the result set. As the results may get very large and quite complex during a targeted typical simulation of a week's duration. It is essential that the browser provides the capability of folding and unfolding certain sub-sets of the result to give the user a chance to keep an overview of the total structure of the result(s).

4.3 Deployment Details

The TENT simulation integration system for the SikMa scenario – though as a project still in development – is set up and in use already. The extensions needed to employ it as a sample application for the Provenance project still need to be set up.

As for the Provenance project specific deployment a provenance server offering the required Web Services needs to be deployed. All communication with the Web Service should be handled through SOAP/XML messages. The provenance store needs to be stored persistently to outlive sessions and/or reboots of the provenance service hosting machine. This service however does not need to be capable of handling larger amounts of data due to the fact that larger scale data stored on a data server will be referenced wherever possible. References within the provenance store to external data would be given as URIs within the content of a p-assertion.

5 Domain Specific Provenance Handling

The general rules defined in section 4 provide an overall view of how provenance is to be applied i the aerospace engineering application. This section goes on to define specific solutions and decisions for individual elements of the mapping.

5.1 Run-Time Provenance Storage

During the processing of a simulation various actor-state p-assertions occur. Messages can be descriptive towards the simulation as a whole, and on the specific workflow. The TENT GUI is the only instance to submit p-assertions on the simulation, whereas mainly the TENT components composing the units within a workflow (simulation process chain) will submit workflow related messages.

All p-assertions need to be time-stamped by the provenance server and uniquely identifiable by an assertion ID. Time stamps can be used for relating p-assertions to events within the TENT logging mechanism, to track invocation orders of parallelized running components, etc.

5.1.1 Simulation Messages

Regarding the simulation as a whole not much information needs to be submitted. Mainly the p-assertion's content describes the simulation's descriptive meta data, a URI to the data server referencing the stored simulation, and finally a computational state of the simulation. The computational state can be "prepared" or "defined" for a composed and/or configured simulation that has not been processed, yet, "in process" if it has been started, and "processed" in case of a finished processing of it.

5.1.2 Workflow Messages

The workflow messages originate from the workflow's components. Here a distinction between process defining and run time p-assertions has got to be made. The process defining messages – just like the simulation messages mentioned in the section above – are read from the component definition prior to the process start. Run time messages, in contrast, are submitted during the process and describe the run time behavior and state of the simulation process. The sequence/invocation order of individual simulation steps (including execution throughout loops) can be traced this way.

For the process defining p-assertions only some characteristic parameters and settings will be recorded, *not* the whole configuration. The aim is to avoid information redundancy as much as possible, but also keep the provenance information to be queried preferable local within the p-structure rather than following references into large scale data sets inside the data server, as this may be a very time consuming process. So only some characteristic parameters of key components within the workflow are submitted directly by p-assertions. For example the air pressure (or density) for the CFD computation could be one of the key parameters to be recorded, as it may be desirable to compare a certain flight maneuver at different altitudes, which are characterizable by the air pressure.

It may be possible for certain computations to change certain computational parameters throughout the course of a (long running) simulation either manually or automatically from their initial setting. These characters are definitely to be considered characteristic parameters that need to be tracked by messages at run time, and thus should already be made available to the p-structure prior to the simulation's start with the process defining messages.

On a more abstract level run time p-assertions may be differentiated into "process" or "logical" messages on the one hand, and "technical" or "infrastructure" messages on the other hand. This distinction is not necessarily important to be retrievable from the p-structure, but it makes sense to

use it for grouping and clarity to the reader. In the following the information exchanged in messages is described according to the previous grouping.

5.1.2.1 Workflow p-assertions – Process/Logical:

The information in these messages describes information on how the simulation is conducted and what the results are.

- Components (changed properties)
- Invocation order
- Change of numeric parameters
- Workflow decisions
- Location of simulation data on server (URI)

5.1.2.2 Workflow p-assertions – Technical/Infrastructure:

In contrast to the previous messages the information contained in these messages are rather considerable as "meta information". Technical information on the infrastructure that has been used is collected here. The information is used to precisely maps the resources used for the simulation. This is necessary as some numerical computations may behave slightly different on different platforms, or in different levels of parallelization. Additionally resource accounting or run time profiling on the workflow level may be performed.

- Components
    - Hosts, executables, transfers
    - Batch queue
    - Code versions, startup time
    - Number of CPUs (MPI)
- Data Transfer
    - File names
    - Transfer times
    - Transferred bytes
    - Communicating hosts
    - Means of transport (CORBA events, GASS, GridFTP)
    - Origin of data

5.2 Provenance Queries

The following is a set of potential provenance queries which could usefully be asked over the provenance data logged in provenance stores. Previously only the synonym *simulation* has been used. As a simulation may be run using varying parameters and configurations (see characteristic parameters in section 5.1.2), to distinguish the term *case* will be used for a particular configuration of an aerodynamic situation (complete model). This includes the configuration of aerodynamic relevant parts, in a certain physical arrangement, and prescribed by specific physical boundary conditions. Every case may be simulated multiple times with changing boundary conditions and possibly changing geometries (e. g. flap settings)

- Given a certain item of result data, as part of which case has it been obtained?
- What is the history of that data item?

- Given a certain configuration, how often has it been simulated?
- How many/what data items have been obtained as the result of this configuration?
- Given a certain aerodynamic part, as part of which cases has this part been simulated?
- Given certain boundary conditions, in which cases have they been applied?
- ...

Information on why a workflow was run, or why a case was run in a certain way is information that may be embedded within the case's meta data, which is browsable in the TENT project browser. This will be stored in a simulation p-assertion (5.1.1). It's meta data might be queried just as it's configuration above. Workflow spanning queries as the consequences of a case to the simulation of other cases would be considered well outside the scope of this project. The only possible workflow spanning queries would be the ones as stated above and possibly similarity comparisons between cases. However, these similarity comparisons would stay on a rather "shallow" level of comparing sequences of recorded p-assertions from actors and maybe some characteristic parameters.

5.3 Summary

The aerospace engineering application present a significant and complex use-case for Provenance deployment. The demonstration system plan will involve several individual services and a single provenance store capturing a wide range of data. Furthermore the mapping provided here describes:

- How provenance data is separated from large volume simulation data in provenance stores.
- How such data is subsequently referenced in provenance assertions.
- Example queries which can be used in technical/scientific domains.

The next step in the provenance application development within the existing AEA is the finer grained analysis of actual occurring provenance data and a detailed data model suitable for implementing the p-assertions. Furthermore a detailed implementation and deployment plan will be determined in agreement with the institute using the AES. It is expected that some mapping assumptions described in this document may need to be revised as a result of the ongoing detailing generated by this activity. The deliverable will also be revised one or more times to remain in line with:

- Updates of the Architecture specifications (D3.1.1)
- Updates of the Tool specifications (D6.1.1)
- Updates of the application mapping provided in the OTM scenario of WP8 (D781.1)

Appendix A Data To Be Recorded

The following skeleton data items are an example of the types of elements contained in the simulation data items which may be stored in the AEA. Additional information on units, data types, structures, etc. of the data item may be given in parentheses behind the item's description.

A.1 Workflow p-assertions – Process/Logical:

(As described in section 5.1.2.1)

- Workflow definition (complex structure, to be determined)
- Components description (complex structure, to be determined)
- Characteristic component parameters (complex structure, to be determined)
- Properties/parameters changed during simulation (also for manual interventions)
- Invocation order
- Workflow decisions (essential for workflow forks)
- Location of simulation data on server (URI)

A.2 Workflow p-assertions – Technical/Infrastructure:

(As described in section 5.1.2.2)

- Components
    - Hosts (FQDN of component's host),
      executables (path name),
      specific environment (if necessary)
    - Inter-component transfers (source and destination *component* identification of transfer)
    - Data server transfers (source and destination *URI* of transfer)
    - Batch queue (FQDN and queue name of computation host),
      batch quere parameters (queue defining and executive parameters for batch system)
    - Code versions (code version string, e. g. a CVS tag of source version)
    - Startup time (for execution, not for submitting the batch job)
    - Number of CPUs (MPI information)
- Data Transfer
    - File names (URIs of files)
    - Transfer times (seconds used for transfer)
    - Transferred volume (bytes transferred for "pay load")
    - Communicating hosts (FQDNs of hosts)
    - Means of transport (CORBA events, GASS, GridFTP)

- Origin of data (component identification, possibly additional information on application that created the data)

References

[1] CORBA web site. http://www.omg.org/corba/
[2] Globus Project web site. http://www.globus.org
[3] Altair PBS Professional web site. http://www.altair.com/software/pbspro.htm
[4] WebDAV project web site. http://webdav.org
[5] Log4j Project web site. http://logging.apache.org/log4j/docs/
[6] Project internal document "Representing Provenance in the DLR Application", Simon Miles, July 2005. http://twiki.gridprovenance.org/bin/viewauth/Restricted/AerospacePStructure (project internal web site)