11-01-2010

Deliverable DJ3.3.1: Composable Network Services use cases

Deliverable DJ3.3.1

Contractual Date:	30-09-2009
Actual Date:	11-01-2010
Grant Agreement No.:	238875
Activity:	JRA3
Task:	T3
Nature of Deliverable:	R (Report),
Dissemination Level:	PU (Public)
Lead Partner:	REDIRIS
Document Code:	GN3-09-198
Authors:	Diego R. Lopez (RedIRIS), Ian Thomson (DANTE), Licia Florio (TERENA), Joan-Antoni García (i2CAT), Eduard Grasa (i2CAT), José M. Alcaraz (University of Murcia), Antonio G. Skarmeta (University of Murcia), Mary Grammatikou (GRNET-ICCS), Constantinos Marinos (GRNET-ICCS), Vassiliki Pouli (GRNET), Manuel Bernal (University of Murcia), Gregorio Martínez (University of Murcia), Cándido Rodríguez (RedIRIS), Maciej Glowiak (PIONIER), Maciej Strozyk (PIONIER), Bartlomiej Idzikowski (PSNC), Zbigniew Oltuszyk (PIONIER), Jan Rucizka (CESNET), Remco Poortinga (SURFnet).

Abstract

GEMBus will use the SOA paradigm to provide a framework to define, discover, access and combine services in the GÉANT multi-domain environment, spanning over different layers from the infrastructure up to application elements. This document presents a list of the use cases considered by the GEMBus team to illustrate the possibilities that an infrastructure like this will be able to offer.

Table of Contents

Table of Figures

0 Executive Summary

The Internet is changing. It is moving from the original model of a network layer capable of dynamically selecting a path from the originating source of a packet to its ultimate destination to a more dynamic and complex structure. This new structure is expected to become service-aware, meaning greater user involvement (user-centric paradigm), more intelligence built into the network (control-plane) to use it in a commodity-fashion, and to allow for a dynamic composition of resources needed to provision a service.

Typically, the greater level of flexibility and independence among the various components is achieved by decoupling the services' definitions and their business functions from the underlying physical implementations. Architectures that fulfil these characteristics are called Service Oriented Architectures (SOA). The benefit of SOA is that components are not rigidly integrated, allowing for a re-organisation of them whenever needed.

The JRA3 Composable Network Services task will use this paradigm to provide a framework to define, discover, access and combine services in the GÉANT multi-domain environment. It will span over different layers, from the infrastructure up to application elements. This framework will adhere to the Enterprise Service Bus (ESB) design pattern. This will allow the integration of services and applications, and provide a set of functionalities that are comparable to those of a physical bus that carries bits among devices in a computer. For this reason the task has been renamed as **GEMBus** (GÉANT Muti-domain Bus).

The goal of GEMBus is to establish seamless access to the network infrastructure and the services deployed upon it, using direct collaboration between network elements, and therefore providing more complex community-oriented services through their composition. The final result will be the availability of network middleware for seamless access to the network infrastructural services in the form of composable network services.

This document presents a list of the use cases considered by the GEMBus team to illustrate the new possibilities that an infrastructure like this will be able to offer. The use cases have been selected to cover the widest range of application domains and potential user communities, together with an ample set of individual services that cover the different layers in a networking environment. The GEMBus team plans to make available prototypes for the described use cases through the lifetime of the project. However, the usual caveats in such a dynamic environment like this apply, meaning that the descriptions presented here are subject to change as a better understanding of the technologies involved and additional requirements from the potential user communities become available.

1 Introduction

The competitive edge of research and education networks is defined by their ability to offer services beyond those available in the market. A key paradigm in this innovation process is seamless collaboration, moving toward an open cloud of services that users can freely compose, not only to access network resources but also to define these resources and allow others to share them, thereby building collaborating communities.

New services are becoming available for the end-user (e.g. repositories, storage facilities and HD videoconference). At the same time, daily work makes more and more usage of online collaborative tools, such as wikis, content management systems and others.

As indicated by many experts, the Internet is changing, evolving from the original model of a network layer capable of dynamically selecting a path from the originating source of a packet to its ultimate destination to a more dynamic and complex structure. This new structure is expected to become service-aware, meaning greater user involvement (user-centric paradigm), more intelligence built into the network (control-plane) to use it in a commodity-fashion, and a dynamic composition of resources needed to provision a service.

Typically the greater level of flexibility and independence among the various components is achieved by decoupling the services' definitions and their business functions from the underlying physical implementations. Architectures that fulfil these characteristics are called **Service Oriented Architectures (SOA)**. The benefit of SOA architectures is that components are not rigidly integrated, allowing for a re-organisation of them whenever needed.

Many experts identify the key concept in SOA as the possibility to have independent services with defined interfaces that can be called to perform their tasks in a standard way. This is without a service having foreknowledge of the calling application, and without the application having or needing knowledge of how the service actually performs its tasks.

An analogy for the SOA paradigm is the composition of music. In this analogy, the functions performed by various services represent digital notes, which can be composed to create different melodies. As a composer can create different melodies using the same notes, or adding new notes to existing ones, a SOA can provide a different service composing existing functionalities or components. Each service has a well-defined interface that is accessible across a distributed environment.

One of the key components of a SOA is the concept of the **Enterprise Service Bus (ESB).** As the term suggests, the functionalities of an ESB are comparable to those of a physical bus that carries bits among devices in a computer. In an architecture that uses an ESB, all communications are handled through it and the ESB acts as a broker between applications and services.

Figure 1.1 shows an abstract view of the ESB:

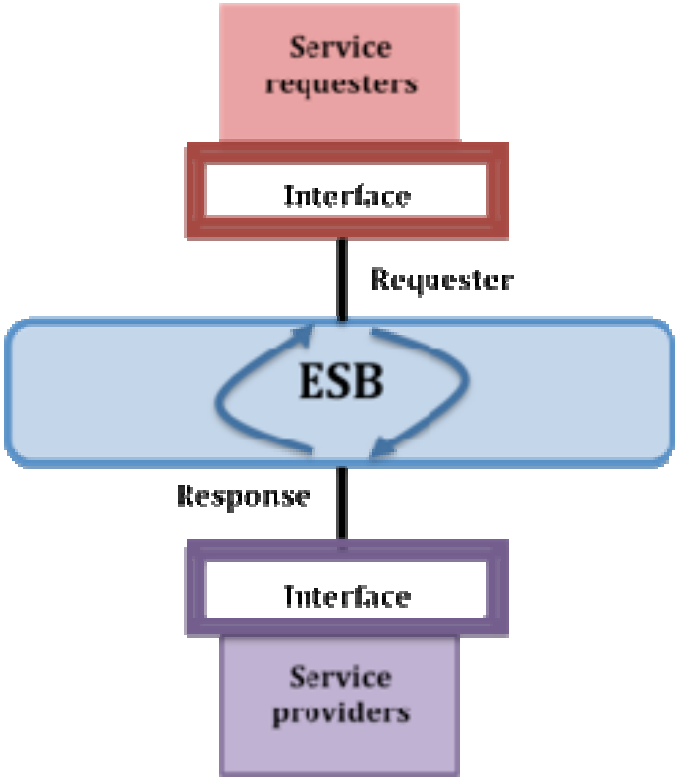

Figure 1.1: ESB representation

An ESB offers a set of infrastructure capabilities for handling communications between service requesters and service providers. In particular, the ESB is able to communicate with the requester using the requester's preferred interface (for instance Web Services, FTP, and others), while allowing the provider to use a different interface. This means that the set of elementary functions have been isolated and can be provided by different entities; the services. All those services communicate with each other using well-defined protocols based on XML message exchange using standardised message exchange patterns.

One of the ESB's responsibilities is to translate the information received by a service's requester in the format expected by the service's provider. The ESB is also responsible for routing the request to reach the right service, and for assigning the necessary priorities among requests. This process of handling requests and prioritising them is called "mediation".

In more sophisticated architectures the ESB also performs monitoring and statistics functions; for example providing information on how often a particular service is requested, by whom, and how many of the requests had problems.

Another advanced feature that some ESBs provide is the support for composition and coordinated execution of services, forming long-running executable business processes.

The main benefits of an ESB are:

- An ESB allows for loosely coupled services, the benefit of this being that new services/functionalities can be easily added to the existing infrastructure.
- Simplification and standardisation of the interfaces used by the service's requesters and the service providers.
- Services can also be requested by other systems, because of the adoption of the ESB.

1.1 GEMBus Goals

The JRA3 Composable Network Services task (T3) will use the ESB paradigm to provide a framework to define, discover, access and combine services in the GÉANT multi-domain environment, spanning over different layers from the infrastructure up to application elements. For this reason the task has been renamed as **GEMBus**: (GÉANT Muti-domain Bus).

The goal of GEMBus is to establish seamless access to the network infrastructure and the services deployed upon it, using direct collaboration between network elements, and therefore providing more complex community-oriented services through their composition. The final result will be the availability of network middleware (APIs) for seamless access to the network infrastructural services (in the form of composable network services).

GEMBus will therefore facilitate the model of building by composition as indicated in Figure 1.2:

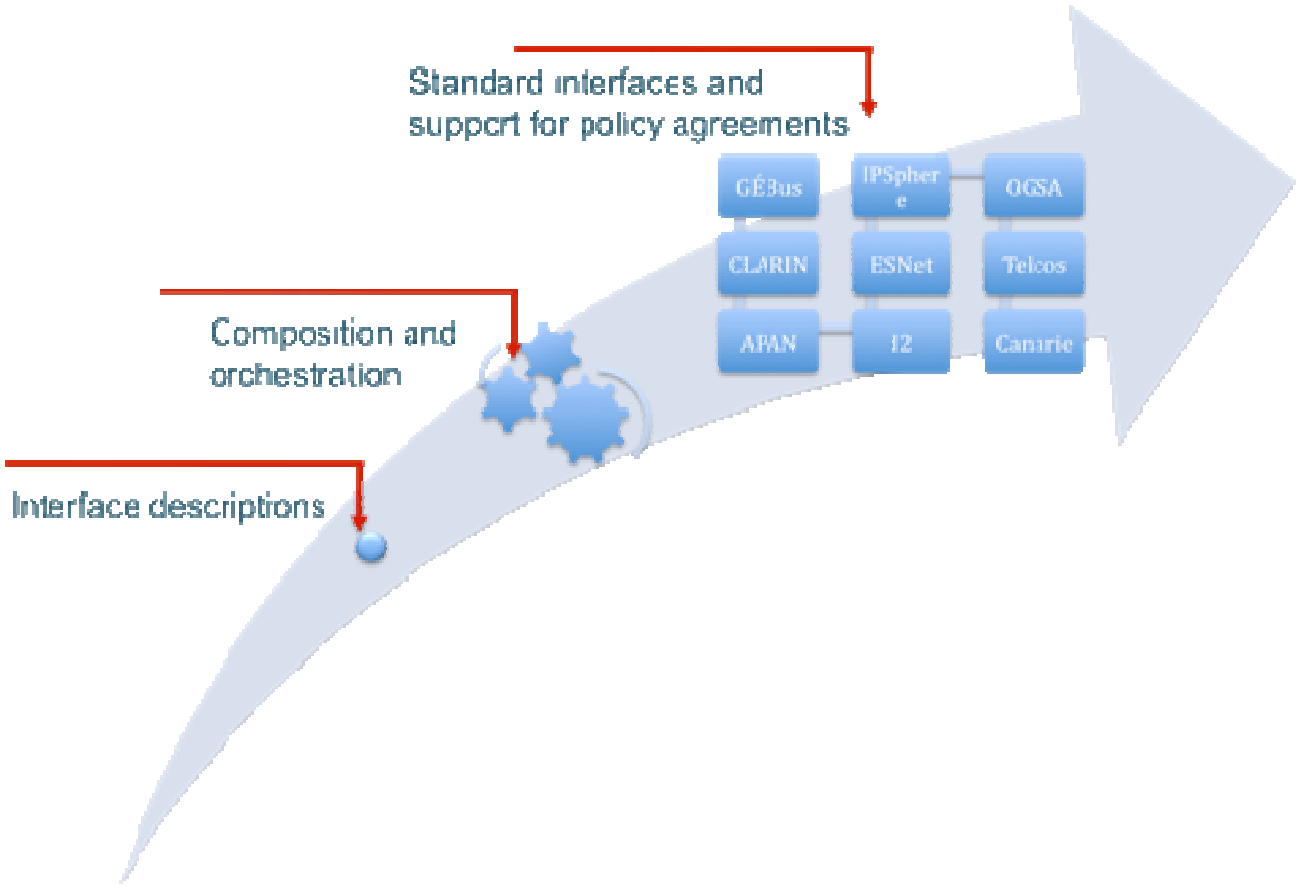

Figure 1.2: GEMBus Model

The figure deliberately includes references to commercial environments. This is dependent upon the industry expressing its commitment to the SOA architecture and the ESB concept in the future (as demonstrated by well-established initiatives, such as IPsphere[1]). Therefore, the development of GEMBus implies a better and more complete integration, not only among academic networks worldwide but also with commercial network infrastructures.

1.2 GEMBus Use Cases

The following sections present a list of the use cases considered by the GEMBus team to illustrate the possibilities that an infrastructure like this offers, together with the requirements that they impose on the features GEMBus has to provide. The use cases have been selected to cover the widest range of application domains and potential user communities, together with an ample set of individual services that cover the different layers in a networking environment. When selecting these particular cases, the group has taken a rather pragmatic approach, incorporating those application domains that could obtain a high potential benefit of the SOA paradigm in their usage of network infrastructures, according to the group's experience as network service providers and researchers

1 http://www.ipsphereforum.org/

The GEMBus team plans to make prototypes available for the described use cases throughout the lifetime of the project. However, in such a dynamic environment the descriptions presented here are subject to change as the technologies involved become better understood, and additional requirements from the potential user communities become available.

A specific subset of these use cases will be applied in the next phase of GEMBus development (the definition and selection of its supporting ESB framework), together with the interface requirements that individual services and its multi-domain nature impose. Furthermore, this validation process will lead to the availability of prototypes for the selected use cases.

Each use case is described according to a common structure:

- An introduction describes the nature of the use case and provides a general description of its context and goals.
- A context diagram is used to make a graphical representation of this general description of the use case.
- The actors in the use case are defined.
- A functional description (in terms of interactions among those actors) is provided, both in a tabular form and in terms of a UML diagram.
- A more detailed specification of the dynamic behaviour of the actors in each use case is included in a tabular form.

2 GEMBus and Grid Infrastructures: KoDaVis Application

2.1 System-wide Functional View

A Grid [GRID2] can be defined as a system that coordinates distributed resources using standard, open, general-purpose protocols and interfaces to deliver high qualities of service. It also integrates and coordinates resources and users that live within different control domains (for example, the user's desktop versus central computing, different administrative units of the same company, and/or different companies), and addresses the issues of security, policy, payment, membership, etc. that arise in these settings. Otherwise, we are dealing with a local management system.

A Grid is built from multipurpose protocols and interfaces that address such fundamental issues as authentication, authorisation, resource discovery, and resource access. It is important that these protocols and interfaces are standard and open. Otherwise, we are dealing with an application-specific system. A Grid allows its constituent resources to be used in a coordinated fashion to deliver various grades of quality of service (relating, for example, to response time, throughput, availability, and security), and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater that the sum of its parts.

Key to the realisation of this Grid vision is standardisation. This is so that the diverse components that make up a modern computing environment can be discovered, accessed, allocated, monitored, accounted for, billed for, etc., and in general managed as a single virtual system, even when provided by different vendors or operated by different organisations. Grid middleware is intended to provide this layer of Grid service harmonisation and orchestration. However, many times Grid workflows may require the use of services from existing, third party systems which are not fully compatible with the Grid middleware, due to specific implementation technology.

Standardisation is critical for the Grid as it aims to create interoperable, portable, and reusable components and systems. It also contributes to the development of secure, robust, and scalable Grid systems by facilitating the use of good practices. The Open Grid Services Architecture (OGSA) [OGSA] is a service-oriented architecture that addresses this need for standardisation by defining a set of core capabilities and behaviours that address key concerns in Grid systems. GEMBus should consider the existing Grid standards, and specially OGSA.

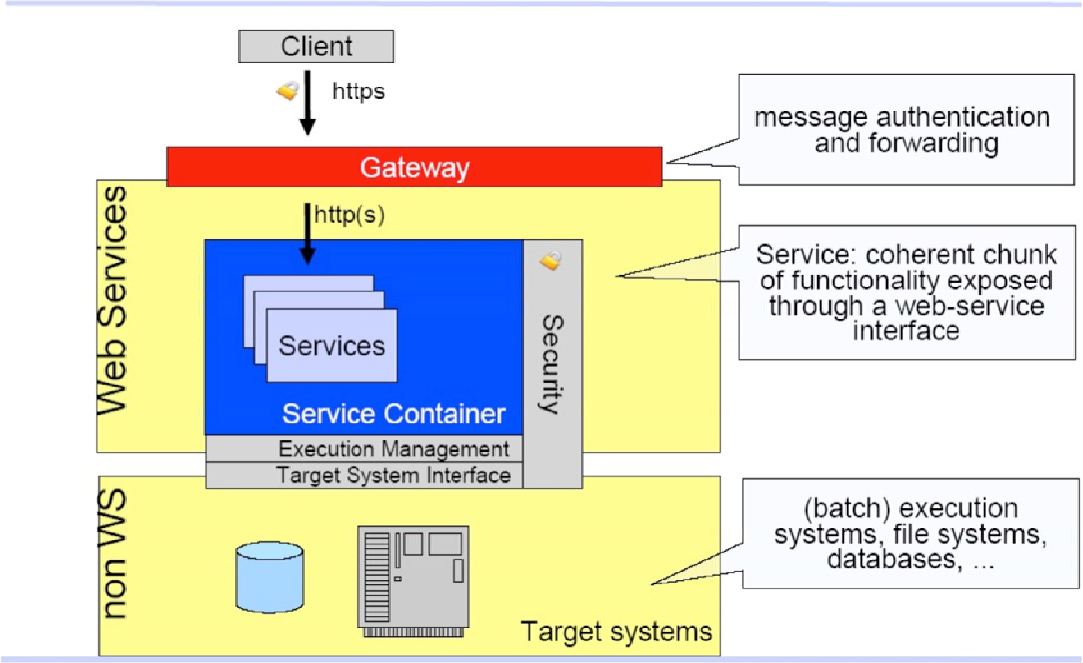

Figure 2.1: Generic context for Grid use case (OGSA architecture)

In this particular use case, OGSA is used by KoDaVis [KODAVIS] application, which stands for Collaborative Data Visualisation. Climatologists use KoDaVis to handle huge collections of data. This data is partly measured (as provided by the European Centre for Medium-Range Weather Forecasts, ECMWF), and partly data coming from simulations performed on supercomputers that simulate different scenarios based on the measured data. A typical dataset contains 1 terabyte of data. Such datasets are stored at the supercomputer site and not locally at the scientists' lab. For visual analysis, only part of the data is accessed, but which part of the data needs to be accessed during the exploration session cannot be fully specified in advance.

The Components of the OGSA in the KoDaVis use case [OGSAUC] are the following:

- **Client (based on GPE)**: A GUI User-Client implemented in Java. It is based on the Grid Programming Environment (GPE) developed by Intel.
- **Gateway**: A single point of entry for the Client to access server functions at a specific Unicore [UNICORE] site. The Gateway checks the users X.509 certificate for validity (issued by a trusted CA), provides the list of available sites and services (service registry).
- **Unicore Server**: This component provides core services (also called atomic services) and additional services available at a site. It uses the Unicore User Database (UUDB) to map the users X.509 certificates to user-IDs on the target systems (TS), where the user's jobs are actually executed. It uses a database called Incarnation Database (IDB) to map abstract job and resource descriptions to their concrete values on a target system. The IDB is fed by the target system. There may be more than one Unicore server at a single site (behind a single Gateway).
- **TSS (Target System Service)**: A core service at the Unicore server. It provides access to a target system.
- **TSI (Target System Interface)**: An adapter typically running on the target system. It communicates with the TSS and ADS and interacts with the local resource management system (batch system, reservation system) of the target system.

- **TS (Target System)**: The actual Grid resource, e.g. a Cluster or HPC-system, or a visualisation system.
- **ADS (Additional Services)**: Non-core services can be provided by the Unicore server as plug-ins, called ADS. Examples for such services are the interface to a network reservation system, like ARGON [ARGON], or services to manage interactive data connections for online visualisation and computational steering.
- **MSS (MetaScheduling Service)**: This component negotiates complex distributed resource requests of a user on behalf of the user. In VIOLA [VIOLA], the MSS was implemented as a separate component. In the European project Phosphorus (FP-6) [IST-PHOSPHORUS] it will be integrated more tightly into the Unicore 6 architecture. It will act as a specific Unicore server that provides only a single service.

2.1.1 Context Diagram

The following diagrams show the context and components involved in this use-case:

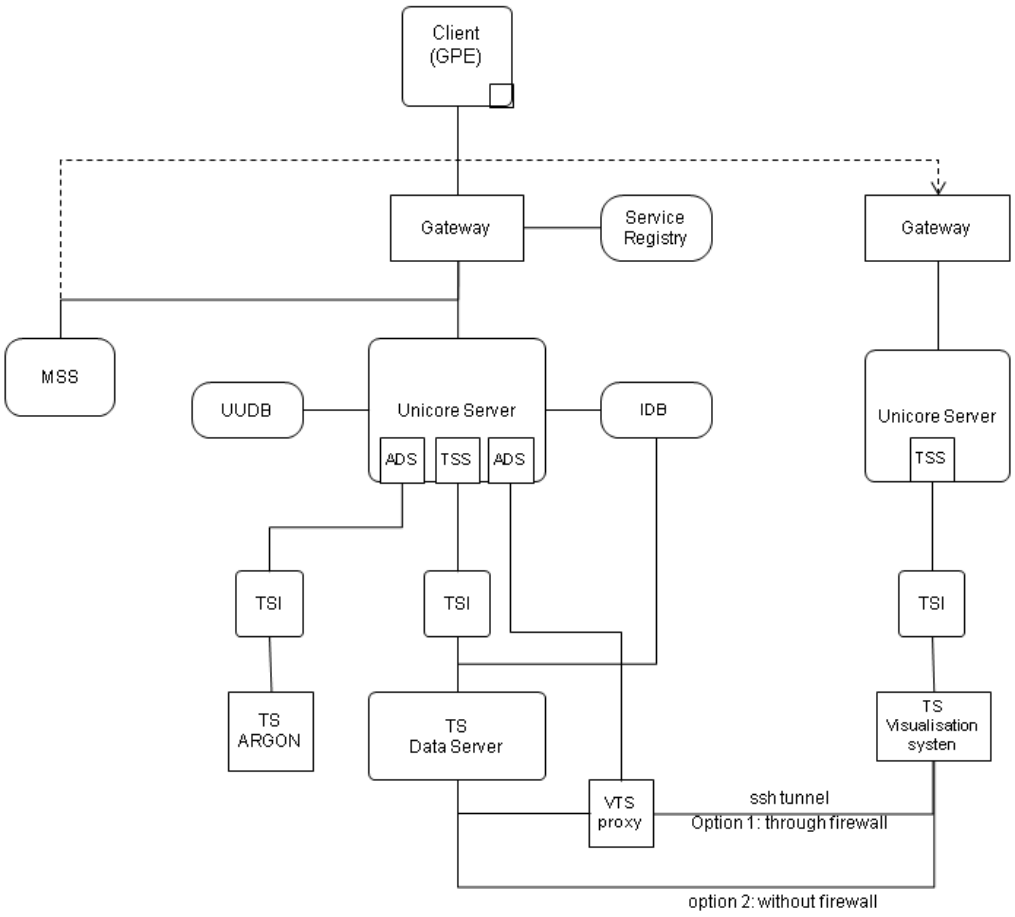

Figure 2.2: Context Diagram for the grid infrastructure use case

Other components in the figure are:

- **Data Server**: The KoDaVis data server is a parallel application that is executed on a target system (a PC-Cluster in the testbed). It provides network access to a huge repository of climate and weather data.
- **Visualisation System**: The visualisation application that retrieves data from the data server for interactive exploration. This application can be executed on any client. In this use case it will be executed on a visualisation system with a reservation system ("online calendar") capable of interacting with a Unicore server through a specific TSI.

2.2 Actors of the System

Actor	Description
VO Administrator	The administrator of a Virtual Organisation. VO administrator handles all necessary security infrastructure and middleware in order to allow coordinated actions from users and other administrators.
Grid Site Administrator	The administrator of a local site contributing resources to the Grid (computing power, storage, connectivity…). In this case, where the KoDaVis server is located. Grid site administrator has a passive role in service provisioning. It does enable the Grid infrastructure, but is not taking active part in the service lifecycle.
Grid User (a.k.a. Scientist)	The user that requests some grid resources to perform one or multiple jobs; typically some complex distributed computation on a computing grid or data mining/visualisation in a data grid.

2.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Grid resource announcement or discovery	Grid resource announcement	**Grid Site admin,** VO admin.	
	Grid resource discovery	**VO admin,** Grid Site admin.	
Session initiation for remote data visualisation	User accesses remote data sets	**Grid user**, Grid Site admin, VO admin.	
	Users access remote data sets collaboratively	**Grid user**, Grid Site admin, VO admin.	

Figure 2.3 presents the Grid infrastructure use case. VO admin administers Grid services from the whole virtual organisation and decides which ones have to be registered in GEMBus.

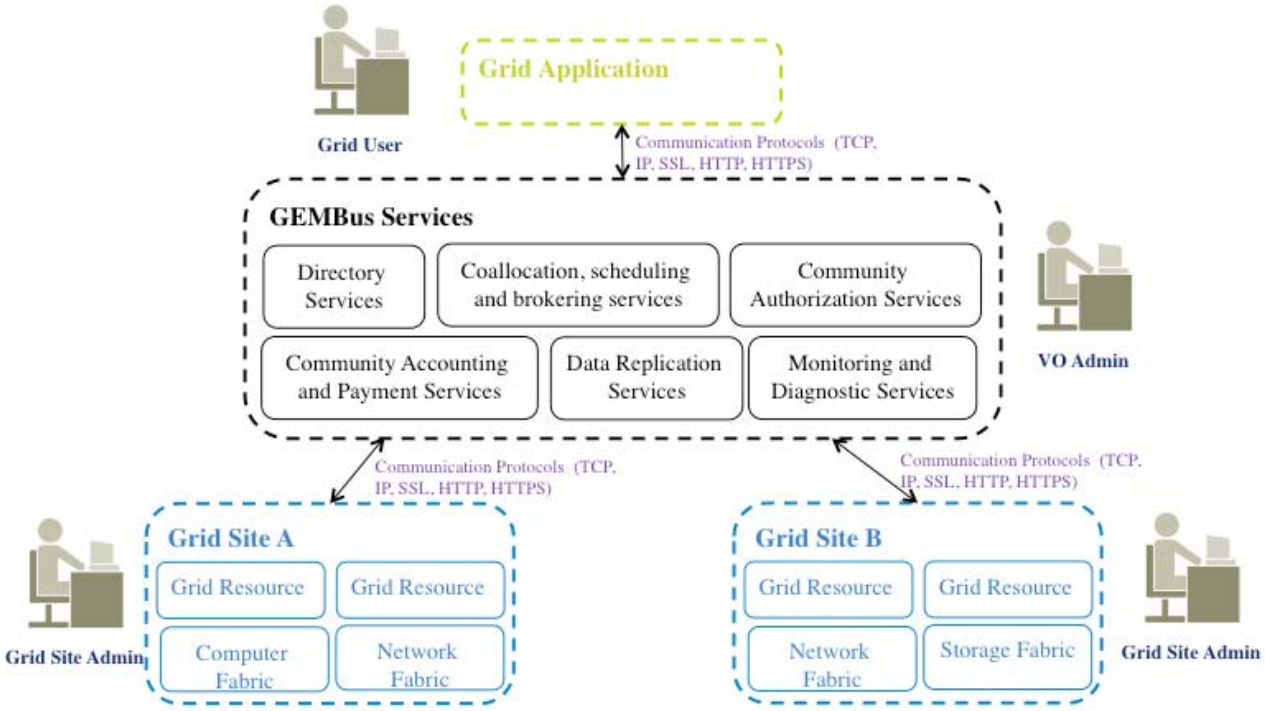

Figure 2.3: Grid infrastructure use case diagram

2.4 Dynamic View

2.4.1 Virtual Organisation Creation (Grid generic)

Use Case	GB01 VO Administrator creates a Virtual Organisation
Description	Either a Grid site administrator or an independent entity related to Grid (user community) wants to create a Virtual Organisation. In this case, the entity assuming the role of Virtual Organisation has to keep contact with user communities either localised in a single corporative entity (enterprise, university, research centre, etc.) or distributed (university campuses, decentralised companies, etc.). Moreover, the VO administrator has to deal with Grid site administrators in order to gather required localisation, scheduling and connectivity information about Grid resources, which may or may not be administered by a single Grid site admin. Note: Typically, a VO administrator is born from a Grid site administrator, which deals directly with a user community (i.e. in-company industrial Grids).

Use Case	GB01 VO Administrator creates a Virtual Organisation
Actors	VO admin., Grid Site admin.
Detection	VO administrators discover Grid resource information.
Assumptions	Grid sites exist and allow creation of VOs.
Preconditions	Grid site administrators must publish their resources or allow requests about them.
Steps	Step 1. VO administrator is constituted from a Grid site administrator or independent entity. Step 2. VO administrator gathers information about available Grid resources for its purposes. Step 3. Grid site administrators provide VO administrator with all required information. Step 4. VO administrator is granted access to the various Grid resources. Step 5. VO administrator composes a directory of services to be offered to users.
Variations	Step 4. If access to Grid resources requires special security actions: proceed to former security protocol. Step 5. If Grid resources are announced in any other way: proceed to announce the resources in the suitable way.
Post-conditions	After the process, a VO exists being administered by a VO Administrator and offering Grid services.
Extends/Includes	N/A
Non-Functional	N/A
Issues	N/A

2.4.2 Visual Analysis of Data (KoDaVis application)

Use Case	GB02 User accesses remote data sets
Description	A scientist wants to perform visual analysis of the data. He or she is using a high-end visualisation facility at his/her own laboratory for the session, or is visiting a remote site with a graphics workstation or may only have his/her laptop at hand for a demo at a conference. A session can be scheduled for a specific time or be spontaneous. In both cases, the scientist starts his visualisation application and requests a data-service from the system on which the data is stored. To get a reliable service, he may also request to reserve network bandwidth between the data service and his local visualisation device. The request would be: “I need data-service for one client for ‘1 hour’ ‘now’ or ‘tomorrow starting at any time between 14:00 and 17:00’. At the same time I need 700 Mbit/s reserved bandwidth between the data-service and my location”.
Actors	Grid User, Grid Site admin., VO admin.
Detection	User logs in the Grid application (KoDaVis) and requests connection to a selected KoDaVis server.
Assumptions	A VO exists (where the KoDaVis server is located).
Preconditions	User belongs to the VO.
Steps	Step 1. User runs KoDaVis client locally. Step 2. User uses KoDaVis client to get access to Grid resources and schedule a data visualisation session. Step 3. VO administrator or middleware provides user with the resource allocation. Step 4. User gets resources instantiated/configured and starts visualisation session. Step 5. User finishes visualisation session and releases Grid resources.

Use Case	GB02 User accesses remote data sets
Variations	Step 3. If resources are not immediately available: VO administrator or middleware provides the user with an alternative schedule or resources availability
Post-conditions	User is able to repeat the remote visualisation of data sets with KoDaVis client using the same workflow described above.
Extends/Includes	N/A
Non-Functional	N/A
Issues	N/A

2.4.3 Collaborative Data Visualisation (KoDaVis application)

Use Case	GB03 Users access remote data sets collaboratively
Description	Scientists at two or more different sites want to collaboratively explore the data. They may be using different hardware and software environments for that purpose. Nevertheless they want to communicate via video or at least audio and need to have a common, synchronised view on the data. The request would be: “We need data-service for three clients for ‘1 hour’ ‘now’ or ‘tomorrow starting at any between 14:00 and 17:00’. At the same time we need 700 Mbit/s reserved bandwidth between the data-service and each of the three locations”. Note: A more advanced request might also include the personal calendars of the scientists and reservation systems for the high-end visualisation facilities to negotiate the earliest possible timeslot, where all the requirements can be met.
Actors	Grid User, Grid Site admin., VO admin.
Detection	Two or more users log in the Grid application (KoDaVis) and request connection to the same data sets in the selected KoDaVis server.
Assumptions	A VO exists (where the KoDaVis server is located).
Preconditions	All user belong to the VO.
Steps	Step 1. Users run KoDaVis client locally. Step 2. Users use KoDaVis client to get access to Grid resources and schedule independent data visualisation session to the same KoDaVis server. Step 3. VO administrator or middleware provides users with the resource allocation, respectively. Step 4. Users get resources instantiated/configured and start independent visualisation sessions to the same data sets in the server. Step 5. Users finish visualisation sessions and release Grid resources.
Variations	Step 3. If resources are not immediately available for one or more users: VO administrator or middleware provides all users with an alternative schedule or resource availability.
Post-conditions	Users are able to repeat the remote visualisation of data sets with KoDaVis clients using the same workflow described above.
Extends/Includes	N/A
Non-Functional	N/A
Issues	N/A

3 Smart Autonomous Network Services

3.1 System-wide Functional View

There are several examples of network services that demand new methods of collaborating and coordinating with other services to create a more complex network service. For example, a current Network Storage Service (NSS) could establish collaborations with other Network Storage Services in order to provide a more complex distributed NSS with added value (for example mirroring, streaming, redundancy, data replication or improving access performance). Another example is the possible collaboration between different database instances in order to provide a distributed network database service with added value (for example redundancy, performance and scalability).

All these composable network services could be improved and controlled by means of an autonomous manager [AUTCOV] that collects data from the Sensor Framework (SF) deployed at the individual servers, and acts as a controller to provide self-optimising, self-healing, self-protecting and self-configuring capabilities to the services provided. This manager could be programmed according to high-level business goals. It would then be in charge of orchestrating and coordinating these underlying network services according to these business objectives [AUTC].

For example, a NSS could be overloaded at a given time. This situation could be monitored by an autonomous manager. In response, the autonomous manager could deploy a read-only mirror of this storage service in a different machine. This deployment could be used, for example, to perform a load balancing of the different requests to reduce the overload of the original service. In this case, the autonomous manager would provide self-optimising capabilities to the storage service.

There are two clear elements that should be considered in this use case:

- An architecture should be available in which the administrator can show the description of the managed NSS, in order to describe the policies that satisfy the business objectives. This architecture would demand that all the NSS elements provide a description of the current capabilities, status and action.
- The autonomous manager should be inserted on the architecture in order to execute the previous policies orchestrating the rest of the NSS involved in the architecture. As a result, self-optimising capabilities will be provided on the network services by means of executing actions on the managed network services. This architecture is shown in Figure 3.1. Therefore, an enforcing module should be inserted on the autonomous manager in order to enable the execution of action from the inferred information [AUTCAR].

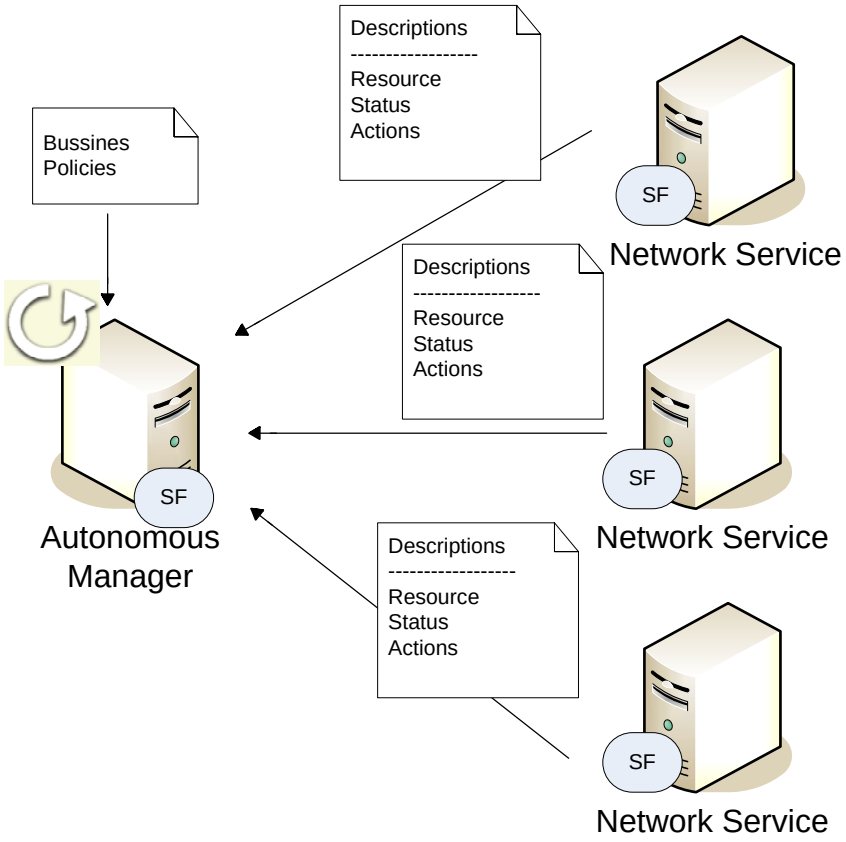

Figure 3.1: Components of a NSS autonomous system

Note that each NSS is providing storage by means of a distributed file system. This kind of dynamic file-system has several advantages by means of the composition of services (such as mirroring, streaming, stripping, parity check, redundancy, fault tolerance, etc). To provide these kinds of services the regular interaction request/response between the third party actor and the NSS should be improved in order to perform autonomous actions to ensure features like healing, optimising or protecting. Moreover, these distributed systems are continuously changing the number of managed volumes, the size of these volumes and the performance provided by each of them. This factor could be improved by means of the collaboration with other NSS elements.

3.1.1 Context Diagram

Figure 3.2 shows the context and components involved in this use-case:

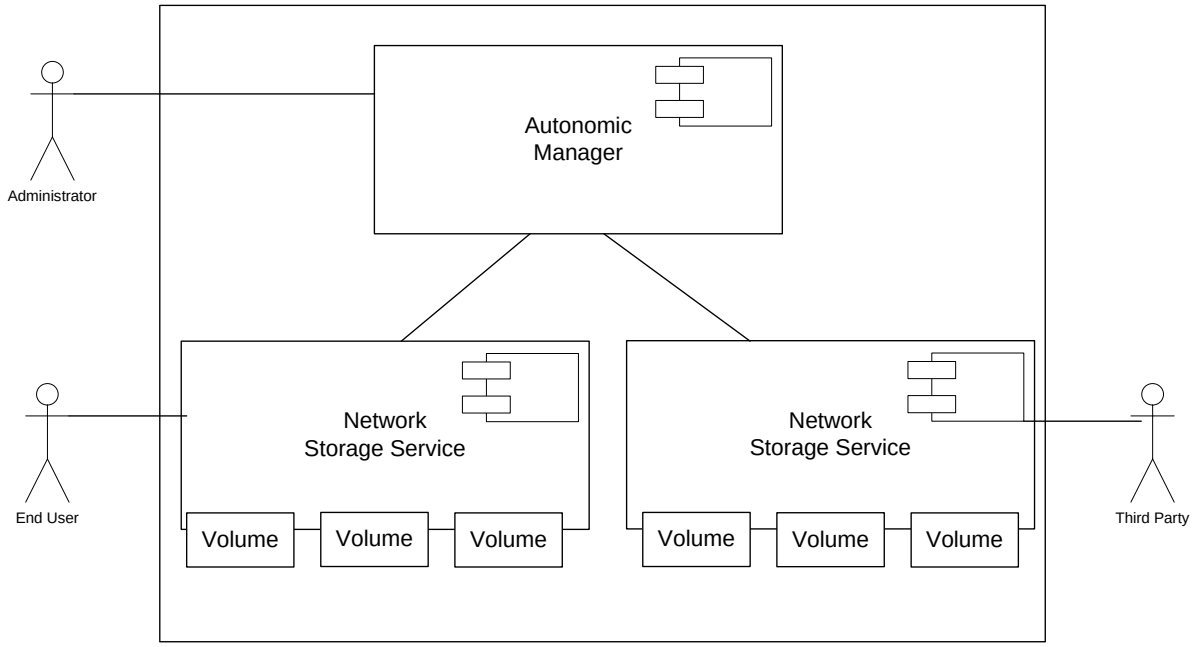

Figure 3.2:Context Diagram for the AC use case

3.2 Actors of the System

Actor	Description
Administrator	In general, this actor is in charge of all the management of the system. In this context, this actor will be in charge of defining the policies which will constitute the autonomic behaviour of the system to achieve self-healing properties.
End-User	This actor will use the NSS. This actor is the real user of the service, and could be an external application/user, represented by means of the End-User actor. Examples of this actor could be a final end-user or an external application/service such as GÉANT cNIS and GÉANT PerfSONAR.

3.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Autonomic Management	Define Policy	**Administrator**	This use case produces the insertion of the policies that will provide the autonomous behaviour in the system.

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
	Start Autonomic Manager	**Administrator**	This use case starts the autonomous manager in order to ensure the application of the policies defined by the administrator.
Network Storage Service (NSS) Management	Manage Volumes	**End-User**	This use case represents the management services of the volumes administrated in the storage services.
	Manage Security	**End-User**	This use case represents the security management related to the access to the NSS.

Figure 3.3 shows the global functional view (including all use cases).

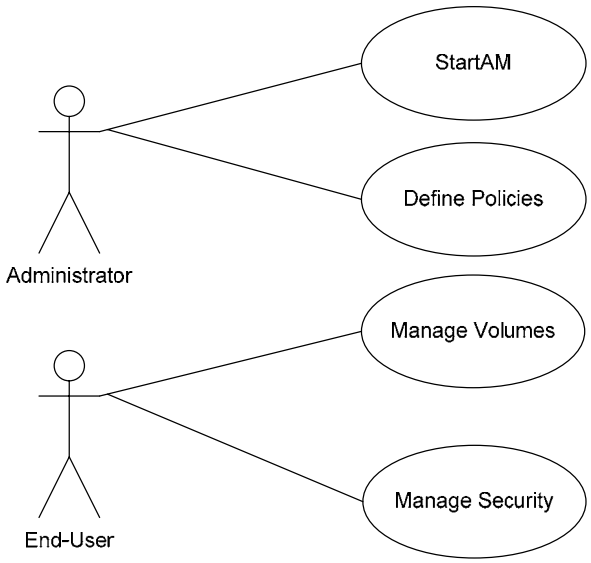

Figure 3.3:Autonomous services use case diagram

3.4 Dynamic View

3.4.1 Define Policies

Use Case	GB04 Define Policies
Description	This use case describes the process by which the administrator defines the policies that will provide autonomous behaviour to the NSS.
Actors	Administrator

Use Case	GB04 Define Policies
Detection	When the software for editing policies is opened
Assumptions	It is assumed that the Administrator has been correctly authenticated and authorised in the system
Preconditions	N/A
Steps	1. Policy Editor retrieves the description of the managed resources and actions related to all the controlled network storages services and shows this information to the administrator. 2. Administrator defines/modifies/manages policies in order to determine behaviour that will be taken into account when the policy antecedent is fulfilled. The intention of these policies is to achieve any of the features related to self-optimising, self-healing, self-protecting, self-configuring. 3. Administrator saves all the policies.
Variations	o In case resources description could not be retrieved: an error message will be shown in the screen.
Post-conditions	All the policies should be saved correctly.
Extends/Includes	N/A
Non-Functional	Policy language should be designed in order to enable the expression of (at least) the behaviour related to the NSS. Security, Privacy, Trust and AAA should be controlled.
Issues	TBC

3.4.2 Start Autonomous Manager

Use Case	GB05 Start Autonomous Manager
Description	This use case represents the starting of the autonomous manager that will control the autonomic behaviour of the NSS.
Actors	Administrator
Detection	When the Administrator starts the Autonomous Manager Service
Assumptions	It is assumed that the administrator has been authenticated and authorised correctly in the system.
Preconditions	Policy has been previously defined by the administrator
Steps	1. Autonomous Manager loads the policies defined previously. 2. Autonomous Manager retrieves the current state of the NSS. 3. Autonomous Manager executed the policies against the status retrieved and obtains a scheduling plan. 4. Autonomous Manager invokes all the actions in the NSS provided in the scheduling plan. 5. Go to step #2.
Variations	o Status of the NSS could not be retrieved: A notification is sent to the Administrator. o An action could not be invoked: A notification is sent to the Administrator.
Post-conditions	NSS is being monitored and controlled by the autonomous manager in order to get any self-healing etc. feature.
Extends/Includes	N/A
Non-Functional	Security, Privacy, Trust and AAA should be controlled
Issues	N/A

3.4.3 Manage Volumes

Use Case	GB06 Manage volumes
Description	This use case represents the process for which an end user manages the NSS.
Actors	End-User
Detection	When an end user actor is trying to access to the NSS.
Assumptions	NSS is running.
Preconditions	N/A
Steps	1. End-User establishes a trust relationship with the NSS (security considerations). 2. End-User performs an API call requesting: creation/deletion/mount/unmount/… of volumes. 3. NSS performs the requested action.
Variations	o NSS could not perform the action: A notification is shown to the user.
Post-conditions	Autonomous Manager will provide autonomic behaviour to this service enabling an added value. For example, mirroring, striping, streaming, parity check, redundancy, replication, fault tolerance...
Extends/Includes	N/A
Non-Functional	Security, Privacy, Trust, AAA might be considered
Issues	Other previous GEANT services (such as cNIS or perfSONAR) could use this service in order to perform the monitoring of the current state of the NSS service.

3.4.4 Manage Security

Use Case	GB07 Manage security
Description	This use case represents the process for which an end user manages the security issues in the NSS.
Actors	End-User
Detection	When an end user actor is trying to control security access to the NSS
Assumptions	NSS is running
Preconditions	N/A
Steps	1. End-User establishes a trust relationship with the NSS (security considerations). 2. End-User defines security considerations. 3. NSS applies the security defined by the actor.
Variations	o NSS could not perform the action. A notification is shown to the user.
Post-conditions	NSS has security control in the access to the service
Extends/Includes	N/A
Non-Functional	N/A
Issues	TBC

4 Digital Repositories

4.1 System-wide Functional View

The expansion of the Internet and ubiquitous access to the web for millions of users has resulted in a variety of new online activities. User-generated content sites include blogs and web forums, social bookmarking sites, photo and video sharing communities, as well as social networking platforms [SOCSRCH].

The exceptional amounts of information in these collaborative sites have resulted in the construction of large repositories of knowledge. This requires new approaches to information access [DRIVER, DSPACE] that are significantly more powerful than the conventional keyword-based methods.

In this use case, a shared digital repository stores and manages a large collection of dynamic web content. In this way, web content can be organised and managed. Data can be collected, disseminated, distributed, catalogued, indexed and controlled with repository methods more efficiently than the traditional methods. One of the essential elements in the deployment of a digital collection involves implementing a technical infrastructure [SDR] that will ensure that its content is incorporated into all search engines. Optimised data storage methods can be applied to offer better search results and attract more users.

The basic technique involves making the contents of the collection data and metadata available to the indexing process and ensuring an easy path into the resources as users click through the results found on the search engine. This makes the management and searching of different web content in the digital repository more efficient.

The architecture comprises the following services:

- Retrieval Service: Through this service a user can query or retrieve data from the digital repositories.
- Submission Service: Through this service the content submitters can submit their data.
- Persistence Service: This service is responsible for the interaction with the various databases and retrieving/and or storing data.
- Composition Service: This service is responsible for the communication between the different components of the digital repository.
- Management Service: Through this service the manager/administrator can manage the digital repository.

4.1.1 Context Diagram

Figure 4.1 shows the context and components involved in this use-case:

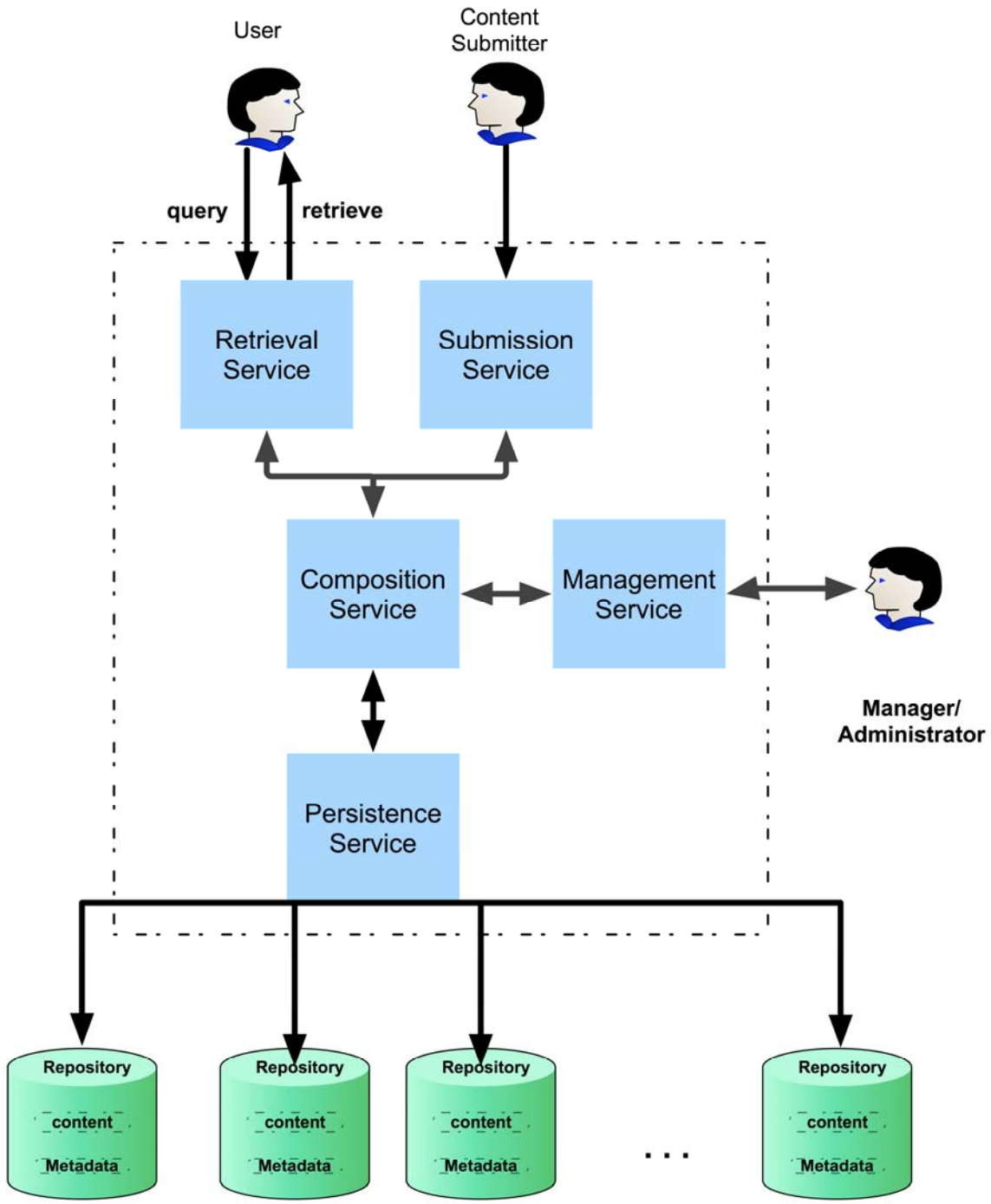

Figure 4.1: Context Diagram for the digital repository use case

4.2 Actors of the System

Actor	Description
End User	The end-user is looking for information through a web interface. Once the requested web object is located, the composition service returns the most relevant objects from the databases, which can be displayed in a readable or downloadable format.
Content Submitter	Content Submitter is responsible for the accurate submission method with specific design patterns. In most cases the content submitter can also be the content producer. The submitted content must be in web-native format that can be viewed immediately (directly in a web browser or with a plug in).
Repository Administrator	The Repository Administrator is in charge of managing the whole repository infrastructure. Through a single point of view (Administrative Interface) the Administrator has an overview of all internal procedures in the Composition Service.

4.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Content Submission	Content Conversion and Submission	Content Submitter, Submission Interface, Administrator	
Information Query/Retrieval	Clients requests and retrieves for specific information	End User, User Interface Administrator	
Repository management	Repository management	Administrator, Management Interface Administrator	

Figure 4.2 illustrates the global functional view (including all use cases).

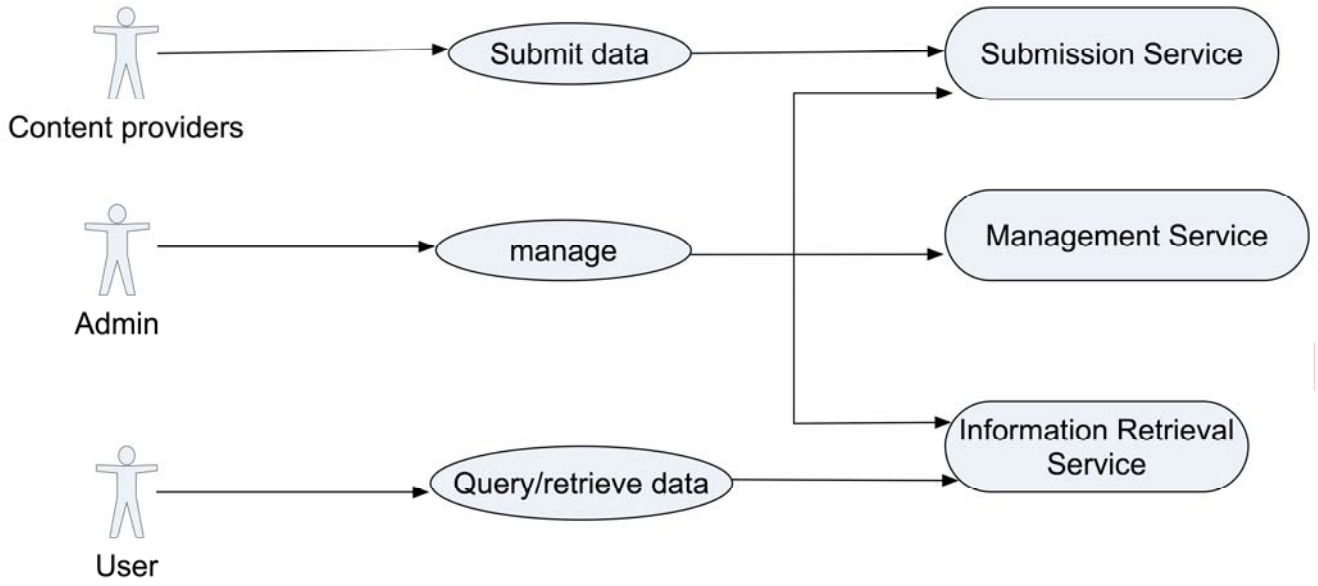

Figure 4.2: Digital repository use case diagram

4.4 Dynamic View

4.4.1 Content Conversion and Submission

Use Case	GB08 Content conversion and submission
Description	The content submitters define specific patterns for all the data to be submitted
Actors	Content Submitter, Repository Administrator, Content Provider
Detection	Content becomes available for submission
Assumptions	The content can be described by the specific patterns
Preconditions	Specific patterns for the description of the content and the metadata must be well defined and available
Steps	Step 1: Content has been transferred from the provider to the submitter, if it is different entity Step 2: Content Submitter converts the specific item of content into a ready-to-submit format with the specific patterns. Step 3: Content Submitter authenticates themselves to the repository through a specific AA server. Step 4: Content Submitter stores the content into the repository.
Variations	None
Post-conditions	The repository will be ready to distribute the content and provide the requested data.
Extends/Includes	None
Non-Functional	Content conversion to a non-crawlable format.
Issues	None

4.4.2 Information Retrieval

Use Case	GB09 Information Retrieval
Description	Clients request information through a web interface. Content must have been submitted to the repository in order to be searchable.
Actors	End Users, UI
Detection	Client decision to access the data repository
Assumptions	Data offered by the Content Producer
Preconditions	Steps 1 to 4 in Use Case GB08 have been accomplished
Steps	Step 1: Client accesses the web interface and requests specific data. Step 2: Repository returns the most accurate data, after advanced searching.
Variations	None
Post-conditions	Client retrieves the most accurate data
Extends/Includes	None
Non-Functional	None
Issues	None

4.4.3 Repository Management

Use Case	GB10 Repository Management
Description	The Administrator creates, edits, deletes, monitors and analyses data and repository procedures
Actors	Administrator, Management Interface
Detection	On event notification
Assumptions	None
Preconditions	None
Steps	Step 1: Administrator receives the event notification, Step 2: Administrator accesses the Management Interface, Step 3: Administrator configures the repository accordingly,
Variations	None
Post-conditions	Repository operating functionally
Extends/Includes	None
Non-Functional	Any repository procedure that does not work
Issues	None

5 Artistic Performances

5.1 System-wide Functional View

The use of Internet technologies to augment artistic works has grown greatly in recent years, as higher bandwidth and more powerful collaboration paradigms became available [AHNET]. In particular, delivery of the content of artistic performances (either live or recorded) and on-line artistic collaboration are becoming more feasible as the highly stringent constraints that these application domains impose on the supporting communication layers can be satisfied.

The nature of artistic performances usually implies high levels of transmission quality in order to guarantee an accurate experience to the audience (and the performers themselves), but also comprehensive security and access control procedures to preserve intellectual property rights (IPR) [OPOB].

Current practice for these events requires ad-hoc deployments and specific supervision by networking specialists, not only to establish the initial set up but also to actively incorporate participants into it, and to monitor the performance of the network services being used.

This use case considers an institution willing to distribute an artistic performance (subject to IPR) to a variable number of sites by means of a multicast group and an encrypted data stream, possibly negotiating in advance parameters for quality of service. Participating sites get the appropriate keys to access the data stream by asserting their rights through a federated digital identity infrastructure. The institution wants to monitor the usage and performance of the distribution process at several points in the network.

The institution system administrator uses the GEMBus infrastructure to allocate the multicast group, establish a path to its root from a content producer, define the authorisation requirements to access to the data stream, and deploy the appropriate measurement points and connect them to the institution collection interface. Any client willing to participate will use the GEMBus infrastructure to establish its rights to access the data, connect to an appropriate router in the multicast tree, and download the key(s) to decrypt the content.

5.1.1 Context Diagram

Figure 5.1 shows the context and components involved in this use-case:

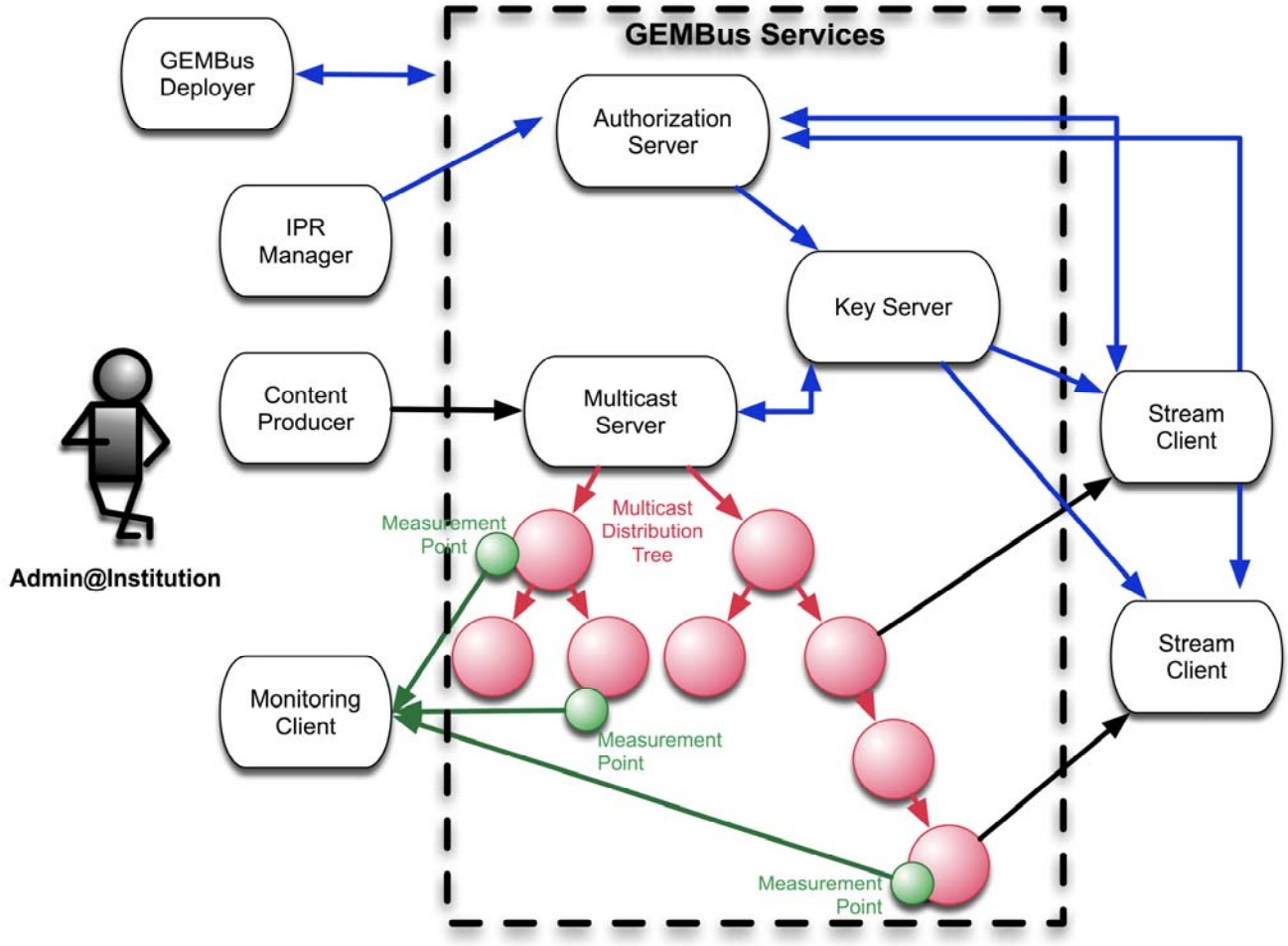

Figure 5.1: Context Diagram for the artistic performance use case

5.2 Actors of the System

Actor	Description
Institutional Administrator	The person(s) in charge of configuring, starting and monitoring the performance using the GÉANT infrastructure through the GEMBus services.
GEMBus Deployment Tool(s)	A set of tools able to collect user requirements and interface with the GEMBus core services, locating the appropriate service endpoints, deploying the appropriate components, and orchestrating their interactions. A typical implementation for such a tool would include a web-based interface adapted to the particular application context.
IPR Manager	Accepts a definition of access rights to a certain set of resources and translates them into authorisation policy statements, able to be used by the GEMBus security services, based on eduGAIN components.
Content Producer	A system producing the data stream that transmits the performance information (typically, audio and video) to the remote audience and/or participants.
Monitoring Client	An element able to collect network usage and performance data from a set of selected points in the network being used to transmit the data stream, based on PerfSONAR services.
Authorisation Server	A component making access-control decisions on the requests to read protected information issued by clients. These decisions are made according to defined policies and requestor attributes, established from its identity within the eduGAIN infrastructure.
Key Server	A system able to store and distribute (upon an authorised request) the keys required to produce and, respectively, to decode an encrypted data stream.
Multicast Server	A system that receives the data stream for the transmitted performance, encrypts it according to the keys collected from the key server, and transmits it through the multicast group that it has previously allocated.
Multicast Group	A set of network resources that transmits the encrypted data stream and makes it available to its potential receivers according to the multicast interaction paradigm.
Measurement Point	An element co-located to a (group of) network resource(s), able to collect and provide access to data related to the resource behaviour and usage.
Stream Client	The intended receiver of the data stream for the performance, able to reconstruct it, show it to the remote audience and/or participant and, if appropriate, send feedback to the content producer and other clients.

5.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Resource location and deployment	Location of service endpoints	Institutional administrator, GEMBus deployment tool	It makes an implicit use of the GEMBus core services
	Service deployment	GEMBus deployment tool, multicast server, key server	
Access control management	Access control management	Institutional administrator, IPR manager, key server	Languages and methodologies for access policy definition are out of scope
Stream distribution	Stream distribution	Content producer, multicast server	
Stream access	Access control	Stream client, key server	
	Stream access	Stream client, multicast group	
Stream monitoring	Monitoring	Monitoring client, measurement point	

Figure 5.2 presents the global functional view (including all use cases).

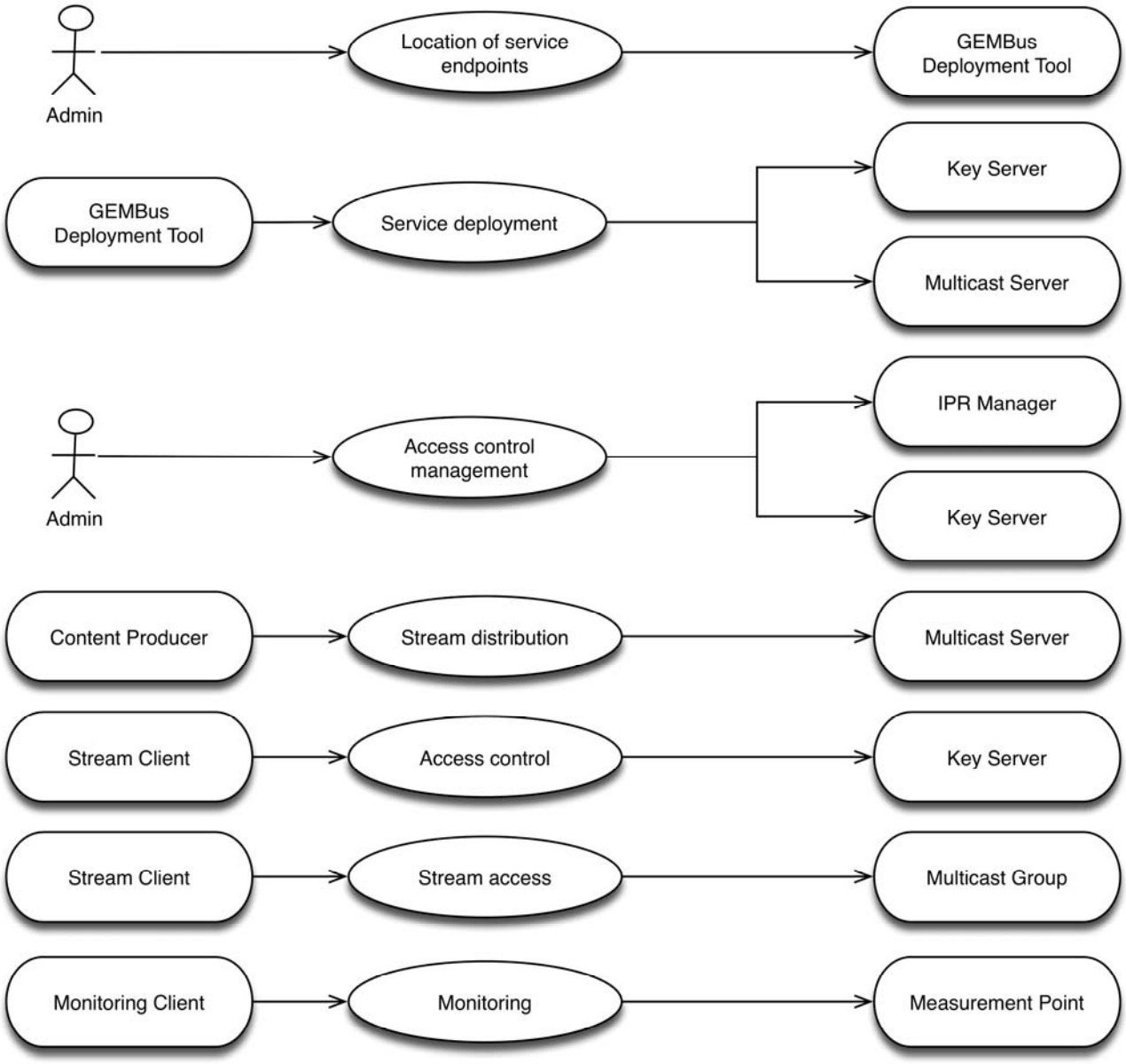

Figure 5.2: Artistic performance use case diagram

5.4 Dynamic View

5.4.1 Infrastructure Deployment

Use Case	GB11 Deployment of the service infrastructure
Description	The administrators at the institution define the characteristics of the event, allocate the necessary resources and deploy the required infrastructure.
Actors	Institutional administrators, GEMBus Deployment Tool(s), IPR Manager, Key Server, Monitoring Client.

Use Case	GB11 Deployment of the service infrastructure
Detection	Event definition becomes available.
Assumptions	Characteristics of the event, including access policies, are clearly defined. GEMBus core services up and running. Monitoring support infrastructure (PerfSONAR) available. Access control infrastructure (eduGAIN) available.
Preconditions	Measurement points can only be deployed at those parts of the network supporting PerfSONAR, but this shall not be considered a condition precluding the deployment of the infrastructure at those points.
Steps	Step 1: The administrators at the institution define the characteristics of the event at the GEMBus Deployment Tool. Step 2: The GEMBus Deployment Tool locates a set of appropriate service access points for: • Accepting the data stream (the Multicast Server). • Distributing keys and applying access control services (the Key Server). • Establishing the PerfSONAR infrastructure required for monitoring. These endpoints are made available to the Content Producer, the IPR Manager, and the Monitoring Client. Step 3: The Deployment Tool contacts the GEMBus orchestration services to interconnect the service access points and make them exchange the necessary information, using a unique ID for the event. Step 4: The above mentioned unique ID is used to build appropriate service endpoints for the clients to access the stream and, optionally, to announce its availability. Step 5: The administrators at the institution define the access policy for the event, and use the IPR Manager to translate it into a federated authorisation policy and transfer it to the Key Server. Step 6: Data about the allocated support infrastructure is made available to the Monitoring Client, which will make the appropriate connections to the PerfSONAR services.
Variations	Step 1: If the access policy is open, then no contact with any Key Server shall be established. Step 5: If the access policy is open, this step shall be omitted.
Post-conditions	The infrastructure will be ready to distribute the data stream and provide monitoring data.
Extends/Includes	None
Non-Functional	Licensing agreements influencing access control policies.
Issues	The way of requesting data for multicast flows from PerfSONAR should be clarified.

5.4.2 Access to the Data Stream

Use Case	GB12 Stream Clients access the data stream
Description	Clients use the unique ID produced in Use Case GB11 to access the data stream, after establishing their rights for it.
Actors	Stream Clients, Key Server, Multicast Group.
Detection	Stream client decision to access the data stream
Assumptions	Data stream is being offered by the Content Producer. Multicast Group is accessible to the Stream Client.
Preconditions	Steps 1 to 6 in Use Case GB11 have been accomplished

Use Case	GB12 Stream Clients access the data stream
Steps	Step 1: Stream Client contacts its eduGAIN endpoint to establish its identity (or the identity of the user it is going to act on behalf of) and gets a credential for the Key Server. Step 2: Stream Client contacts the Key Server, which validates client rights according to the configured authorisation policy. Step 3: Upon positive authorisation, the Key Server provides the Stream Client with the appropriate cryptographic material to decode the data stream Step 4: Stream Client contacts the Multicast Group and proceeds to play the content locally.
Variations	Step 1 to 3: Shall be omitted if the access policy is open. Step 2: The Key Server may require to be contacted prior to Stream Client identity exchange, implying the use of an Identity Provider discovery process. Step 3: If the Stream Client attributes does not satisfy the authorisation requirements, no key will be provided, and an error will be returned instead.
Post-conditions	Stream Client playing (or in disposition to play) the event content.
Extends/Includes	None
Non-Functional	None
Issues	Does it make sense to extend monitoring up to client interfaces, therefore making E2E data available?

6 Collaboration Platforms

6.1 System-wide Functional View

Work practices are evolving from traditional local groups towards a new paradigm of virtual groups, where experts (eProfessionals) have to work together regardless of their geographical location. In this context, these experts use Collaborative Work Environments that enable them to share information and exchange different point of views to reach a common understanding. These Collaborative Working Environments involve several actors:

- Organisations: A Collaborative Work Environment may be established inside an organisation (workgroups) or between different organisations.
- Experts (eProfessionals).
- Underlying technologies, from the simplest ones (like e-mail) to the most advanced and complex Information Systems and Communication Technologies.

Information and Communication Technologies (ICTs) are the main support for these new work practices. ICTs have evolved dramatically over the years, both in architecture as well as in functionality. From the first platforms built over a centralising scheme dominated by the concept of a unique provider, to now where the architecture has moved to that of a “cloud” where, although client-server roles still exist, any element in the platform can be provider and consumer at the same time, and any participant in the “cloud” can provide or consume.

Providers have also changed their way of participating. Against a centralised scheme where information is provided through a well-known and well-located service, now information is no longer the only value of the provider [SWS]. In spite of this, clients also look for services (functionality) on providers. And those services are now distributed so that they no longer need to be in the same provider; they can be found over the network and composition of subsets of services will be used to offer clients a final set of complex business processes [ACSWS].

A basic principle to achieve this organisation is the independence between processes in a way that the service in charge of a single process could be as simple or complex as the activity it is carrying out. In addition, a service can use another service with the only requirement that each service must know the existence of the others, so a mechanism to describe them is needed. This description must have at least the name of the service and the data that is expected to be received and to be returned [MDSWS].

Collaboration platforms deal with a heterogeneous portfolio of tools provided by the different partners, with one point in common; they are oriented towards the eProfessional collaborative activities, in their many forms. As a first attempt to classify them, we note that two types of applications exist:

- Those used in a synchronous way, where tight time constraints exist between the actions occurring.
- Those which are asynchronous, with highly independent interactions (in time).

Due to the fact that this division is also patent in the way that they are actually used, it makes sense that any analysis deals with them separately.

On the other hand, the objective is to construct a more productive environment where the (originally) separated tools can coexist, presenting the eProfessional the possibility to use all of them without even noticing the inner interactions taking place. Therefore, mechanisms to allow the applications to communicate, offer their services and take advantage of the services offered by the others must be ensured.

In the frame of collaborative platforms, Web Services are a fundamental technology over which all the business logic is developed from the bottom, up to the user interface. Those services implement basic functionalities, making them eligible by the business processes that will build much more complex business logics orchestrating them. This conception consolidates the idea of a **CoCoS**[2], as a combination of services offered by one or more applications to achieve a more complex task, concealing most of the complexity that would otherwise appear if they had to be orchestrated manually. Therefore, the ability of orchestrating services to achieve much more complex goals is a key factor in collaborative platforms, but composing services through business processes presents an important problem; the lack of adaptability to any situation.

For example, consider a business process of uploading a document and notifying users. It is represented on the system through an abstract description that will be processed by the Semantic System to execute it using semantic information. It is divided into two different stages:

- Uploading a document: To accomplish this, the Semantic System takes into account the user's profile and characteristics of the document to select the appropriate shared workspace.
- Notifying users: The Semantic System takes into account the user's profile, its context rules that define the preferred media for notification, and context information in order to select the right notification service.

For example, if a notification message includes some kind of attachment, and according to the context information, the user's profile and the user's context rules a mail service must be used, the Semantic System will choose the most suitable to accomplish the task giving all this information, including the characteristics of the attachment. Figure 6.1 illustrates this situation:

[2] Collaborative Composition of Services: describe an orchestration of services to achieve a more complex goal/functionality than the original offered by the services involved.

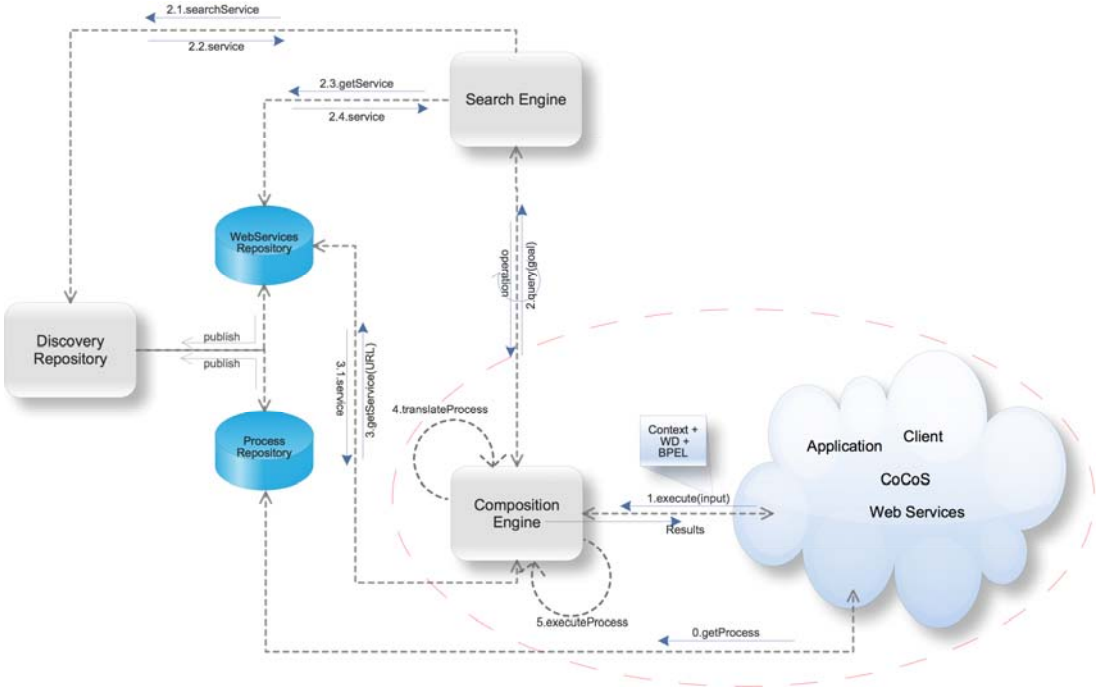

Figure 6.1: General semantic system architecture

The main components are:

- Discovery Repository: Responsible for maintaining the publishing of services available for the dynamic orchestration.
- Composition Engine: Responsible for translating abstract business processes into executable ones and executing them. To accomplish this it will use the information provided by the Search Engine.
- Search Engine: An abstract business process. The services involved are referenced as semantic annotations that represent the functionality needed in each step of the orchestration process, thus, decoupling the underlying services from the business process and making able the dynamic composition. The Search Engine is responsible of locating the most suitable services that fulfil a particular goal using semantic information.

6.1.1 Context Diagram

Figure 6.2 shows the context and components involved in this use-case:

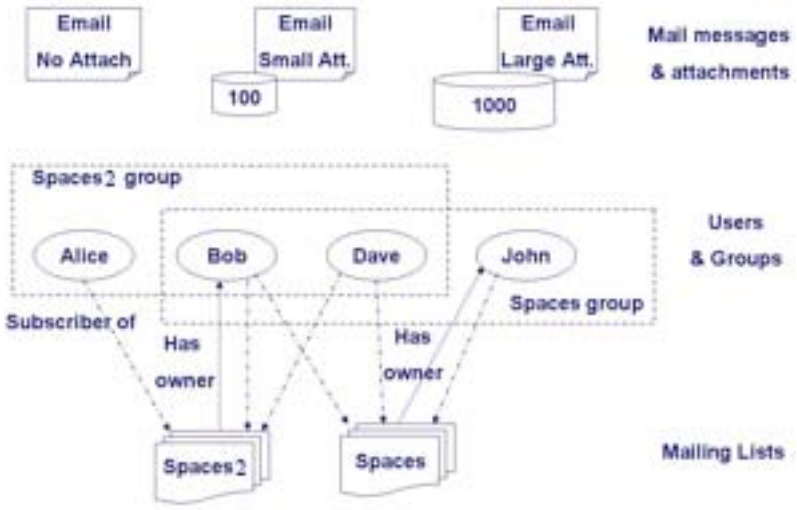

Figure 6.2:: Context Diagram for the collaboration platform use case

Referring to Figure 6.2, if we have the following services:

- Service1-SendMailToMailingList_With_Small_AttachmentLimit:
  - Sender needs to be:
    - Either a subscriber of the Mailing List.
    - Or a member of the group of the Mailing List owner.
  - Attachment size < 100KB.
- Service2-SendMailToMailingList_With_Large_AttachmentLimit:
  - Sender needs to be:
    - Either a subscriber of the Mailing List owner.
    - Or a member of the group of the Mailing List owner.
  - Attachment size < 1000KB.

For example:

If Bob wants to send a message to Spaces2 with attachment = 90 → Service1 and Service2 could be selected, because sender is subscriber to Spaces2 mailing list and message has small attachment.

Bob wants to send a message to Spaces2 with attachment = 200 → Service2 could be selected. Because sender is subscriber to Spaces2 mailing list but message has large attachment.

John wants to send a message to Spaces2 with attachment = 900 → Service2 could be selected. Sender is not a subscriber of Spaces2 mailing list, but is member of Spaces mailing list as well as Spaces2 owner and the message has large attachment.

Alice wants to send a message to Spaces with Attachment = 50 → None. Because sender is not a subscriber of Spaces mailing list and is not a member of the same group as Spaces owner.

The scenario shown as example presents the key use case in eCollaboration environments where the services are orchestrated to achieve the final functionality to the user, supporting the exchange of both information and services between different partners.

6.2 Actors of the System

Since this is a generic use case that takes place on a composite collaboration environment involving the communication between services, actors are not precise but are generalisations.

Actor	Description
Applications	Applications available on the collaboration platform, will communicate with other services to offer functionality to the user.
Web Services	Will offer basic or complex functionalities to other members of the collaboration platform.
Business Processes	Will offer functionality based on the orchestration of Web Services, other Business Processes or both.
Communication Middleware	Part of the platform on the services side, responsible for communication between members of the collaboration platform.
Augmented Middleware	Responsible for the dynamic composition of services using semantic information. It extends the capabilities of the Communication Middleware.

6.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
eCollaboration	Composition of services	**Communication Middleware,** Applications, Web Services	
Semantic eCollaboration	Semantic composition of services	**Augmented Middleware,** Applications, Web Services	

Figure 6.3 presents the global functional view (including all use cases).

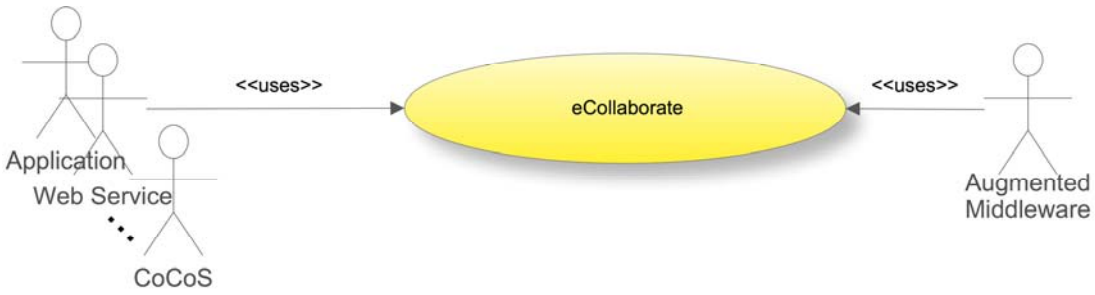

Figure 6.3: Collaboration platform use case diagram

6.4 Dynamic View

6.4.1 Composition of Services

Use Case	GB13 Composition of Services
Description	This is the fundamental Use Case that takes place on a collaboration environment involving communication between services.
Actors	Communication Middleware, Applications, Web Services.
Detection	N/A
Assumptions	A general collaborative platform, services and CoCoS offer functionality through the public interface.
Preconditions	All the actors involved must be available at the time the collaboration is performed.
Steps	This flow is defined over a regular business process where collaboration between different services takes place: From the user interface or another service/CoCoS, an action is triggered that uses one or more services/CoCoS. Those services/CoCoS are executed, which may involve different services/CoCoS as well. Results are returned to origin.
Variations	Does not apply.
Post-conditions	Communication between services done and collaborative goal achieved.
Extends/Includes	No links to other Use Cases in this sense.
Non-Functional	As this is a generalisation of concrete collaboration Use Cases, there are no non-functional circumstances that impact it.
Issues	N/A

6.4.2 Semantic Composition of Services

Use Case	GB14 Semantic Composition of Services
Description	This is an extension of the fundamental Use Case that takes place on a collaboration environment involving communication between services.
Actors	Augmented Middleware, Applications, Web Services.
Detection	N/A
Assumptions	A general collaborative platform, augmented middleware, semantic information available, services and CoCoS offer functionality through the public interface.
Preconditions	All the actors involved must be available at the time the collaboration if performed.
Steps	In the case the interaction between services involve an abstract business process, the overall steps of interaction are as follows: The service/CoCoS that wants to interact with an abstract business process calls the Composition Engine (which is one of the main components of the Semantic System and at the same time is available as a service integrated in the collaboration reference architecture) to execute an abstract process, providing information about the process, for the process and context information. The Composition Engine analyses the abstract process and queries the Search Engine to obtain the most suitable services to be used in the execution of the process.

Use Case	GB14 Semantic Composition of Services
	The Search Engine looks for services on the Discovery Repository. The Discovery Repository returns the services available to the Search Engine. The Search Engine obtains and analyses the services from the Web Services Repository. Then, it uses semantic information to select the most suitable and provides them to the Composition Engine. For every service needed, the Composition Engine retrieves its information from the Web Services Repository while translating the abstract business process into an executable one.
Variations	N/A
Post-conditions	Communication between services done and collaborative goal achieved.
Extends/Includes	Extends Use Case GB13, Composition of Services.
Non-Functional	As this is a generalisation of concrete semantic collaboration Use Cases, there are no non-functional circumstances that impact it.
Issues	N/A

7 Direct Access to PerfSONAR

7.1 System-wide Functional View

PerfSONAR is an infrastructure for network performance monitoring, making it easier to solve end-to-end performance problems on paths crossing several networks. It contains a set of services delivering performance measurements in a federated environment. These services act as an intermediate layer between the performance measurement tools and the diagnostic or visualisation applications. This layer is aimed at making and exchanging performance measurements between networks, using well-defined protocols.

PerfSONAR is a services-oriented architecture. This means that the set of elementary functions have been isolated and can be provided by different entities, called services. All those services communicate with each other using well-defined protocols.

This software offers a powerful environment for network performance monitoring but it still requires a complex setup. The main goal is to offer a very simple interface for obtaining multi-domain network performance results.

7.1.1 Context Diagram

Figure 7.1 shows the context and components involved in this use case:

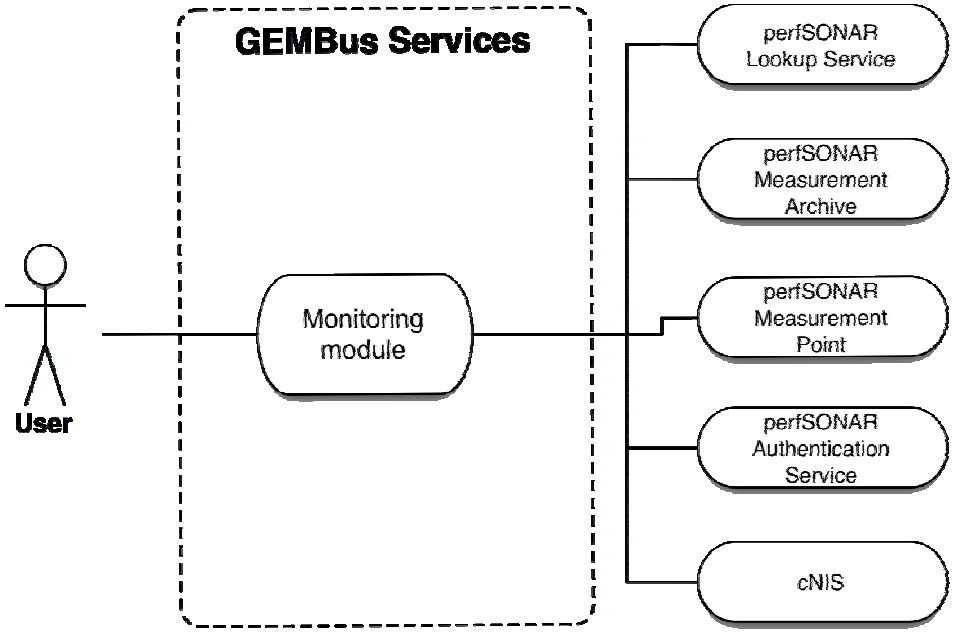

Figure 7.1: Context Diagram for the direct access to PerfSONAR use case

7.2 Actors of the System

Since this is a generic use case that takes place on a composite collaboration environment involving the communication between services, actors are not precise but are generalisations.

Actor	Description
User	Is the person who wants to monitor the performance of a link between two endpoints using the GEANT infrastructure through the GEMBus services.
Monitoring Module	An element able to collect network usage and performance data interacting with perfSONAR services and cNIS.
Lookup Service	The perfSONAR Lookup service (LS) keeps track of which perfSONAR web services are available. The web services can register with the LS at regular intervals to signal that they are running, so that other clients (usually visualisation tools) can then request this information from the LS to find out which services are available.
Measurement Point	The perfSONAR Measurement Point has the capability of creating and publishing monitoring information based upon active and passive measurements.
Measurement Archive	The perfSONAR Measurement Archive retrieves circuit/lightpath status and interface information: link utilisation, link capacity, input errors, output drops…
Authentication Service	The perfSONAR Authentication Service (AS) provides authentication and authorisation to protect perfSONAR web services from unrestricted access.
cNIS	A service able to provide data about the network topology of participating domains.

7.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Network monitoring	Monitoring initialisation	User, Monitoring module	It makes an implicit use of the GEMBus core services
	Monitoring process	Monitoring module	
	Monitoring ending	Monitoring module	

Figure 7.2 presents the global functional view (including all use cases):

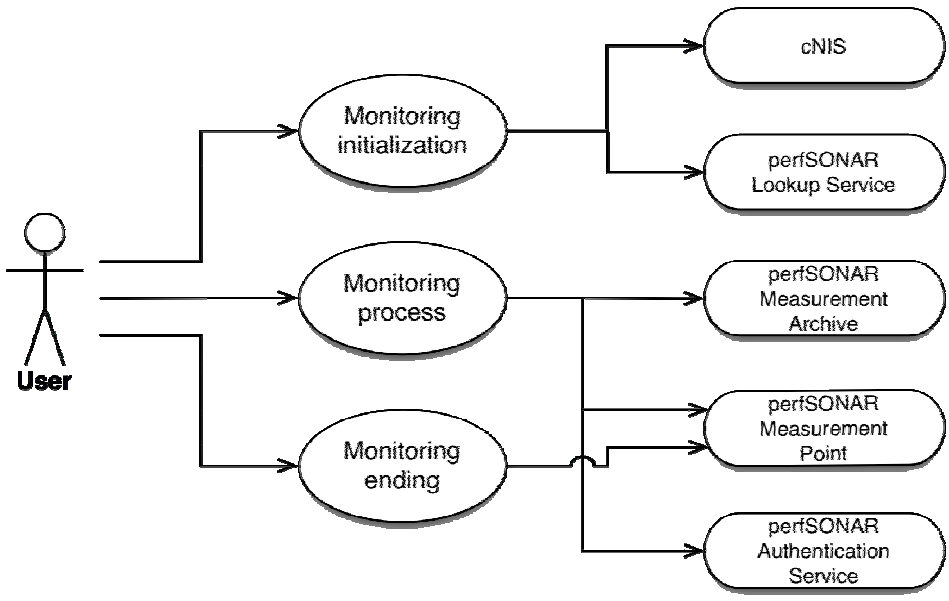

Figure 7.2: Direct access to PerfSONAR use case diagram

7.4 Dynamic View

7.4.1 Monitoring Initialisation

Use Case	GB15 Monitoring initialisation
Description	A user requests information about the performance between two endpoints.
Actors	User, Monitoring module, cNIS, perfSONAR Lookup Service.
Detection	User's request
Assumptions	Monitoring support infrastructure (PerfSONAR) available. Access control infrastructure (eduGAIN) available. cNIS available.
Preconditions	Measurement points can only be deployed at those parts of the network

Use Case	GB15 Monitoring initialisation
	supporting PerfSONAR, but this shall not be considered a condition precluding the deployment of the infrastructure at those points.
Steps	Step 1: User requests performance monitoring between two endpoints: X1 and X2. Step 2: The Monitoring module requests the list of nodes between X1 and X2 from cNIS service. Step 3: The Monitoring module requests the list of Measurement Points and Measurement Archives services available for the obtained list of nodes from the perfSONAR Lookup Service.
Variations	Step 1: If X1 or X2 are not valid endpoints, an error is returned.
Post-conditions	The results contain the list of perfSONAR service elements that may be accessed by the Monitoring module.
Extends/Includes	None
Non-Functional	None
Issues	None

7.4.2 Monitoring Process

Use Case	GB16 Monitoring process
Description	Monitoring module uses the list of perfSONAR services produced in Use Case GB15 to get the performance monitoring.
Actors	Monitoring module, perfSONAR Measurement Point, perfSONAR Measurement Archive.
Detection	Monitoring module decision to access the performance data.
Assumptions	Performance data is offered by perfSONAR MPs and MAs.
Preconditions	Steps 1 to 6 in Use Case GB15 have been accomplished.
Steps	Step 1: Monitoring module contacts the user's eduGAIN endpoint to establish their identity. Step 2: Monitoring module contacts perfSONAR Authentication service in order to know which MPs and MAs the user is allowed to get information from. Step 3: Monitoring module starts the monitoring of the performance in allowed MPs. Step 4: Monitoring module gets performance data from allowed MAs.
Variations	None
Post-conditions	Performance data collected.
Extends/Includes	None
Non-Functional	None
Issues	None

7.4.3 Monitoring Ending

Use Case	GB17 Monitoring ending
Description	Monitoring module uses the list of perfSONAR MP produced in Use Case GB16 to end the performance monitoring.
Actors	Monitoring module, perfSONAR Measurement Point.
Detection	Monitoring module decision to end the performance monitoring.
Assumptions	Performance monitoring was started by perfSONAR MPs.
Preconditions	Steps 1 to 4 in Use Case GB16 have been accomplished

Use Case	GB17 Monitoring ending
Steps	Step 1: Monitoring module contacts perfSONAR MPs in order to end and remove the requested monitoring processes and resources.
Variations	None
Post-conditions	Performance data collected
Extends/Includes	None
Non-Functional	None
Issues	None

8 AutoBAHN/PerfSONAR Integration

8.1 System-wide Functional View

AutoBAHN has been designed to allocate network bandwidth to users/applications, both immediately and in advance. Networking resources in the form of dynamic circuits are allocated, end-to-end, across multiple domains, creating a complex problem of co-ordination and dynamic re-configuration of resources within a number of administrative domains. The granularity of resource reservations in terms of bandwidth and duration is important, together with the required Quality of Service (QoS) parameters.

The AutoBAHN system is based on the Inter-Domain Manager (IDM), a module responsible for inter-domain operations of circuit reservation on behalf of a domain. This includes inter-domain communication, resource negotiations with adjacent domains, request handling, and topology advertisements.

PerfSONAR is an infrastructure for network performance monitoring, making it easier to solve end-to-end performance problems on paths crossing several networks. It contains a set of services delivering performance measurements in a federated environment. These services act as an intermediate layer, between the performance measurement tools and the diagnostic or visualisation applications. This layer is aimed at making and exchanging performance measurements between networks, using well-defined protocols.

PerfSONAR is a services-oriented architecture. This means that the set of elementary functions have been isolated and can be provided by different entities, called services. All these services communicate with each other using well-defined protocols.

The integrated AutoBAHN and PerfSONAR use case will usually occur when someone (the client or another service) wants to dynamically reserve a path with specified parameters (e.g. bandwidth) together with monitoring some of the nodes in the path.

8.1.1 Context Diagram

Figure 8.1 shows the context and components involved in this use-case:

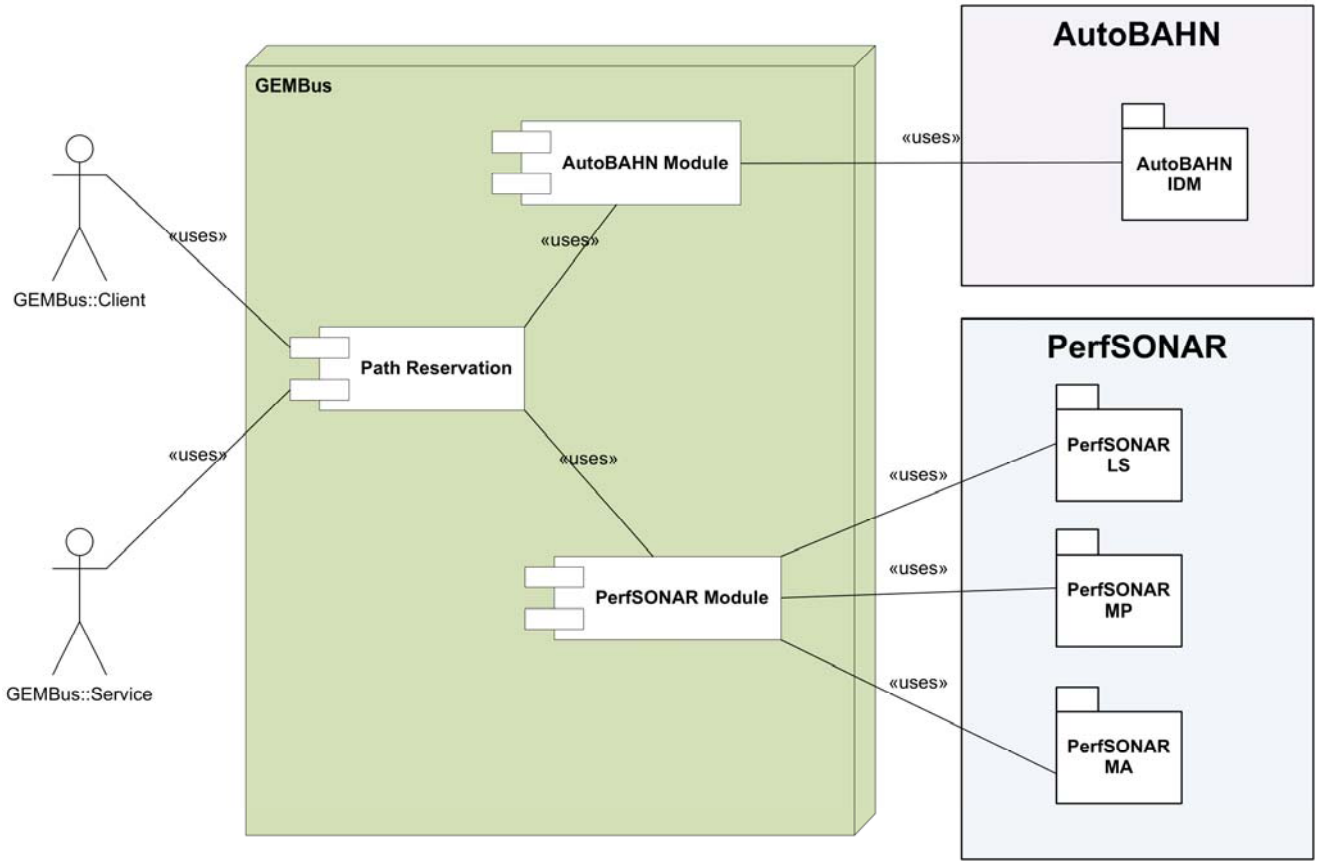

Figure 8.1: Context Diagram for the AutoBAHN/PerfSONAR use case

8.2 Actors of the System

Actor	Description
GEMBus:Client (a.k.a. Client)	Client requests some actions from the system. The action can be path reservation with additional monitoring.
GEMBus:Service (a.k.a. Service)	This is another type of Client. Usually it's another service or module of the GEMBus system that requests some actions from the actors described below. It is sometimes called automated client, which means a client without user interaction. The action can be path reservation with additional monitoring. In the following sections the Service will be treated the same as Client (i.e. if the "Service" is not mentioned explicitly, the "Client" is "Client or Service")
AutoBAHN	All services belonging to the AutoBAHN environment. The AutoBAHN Inter-Domain Manager (IDM) should handle all requests of the system.
PerfSONAR	All services belonging to the PerfSONAR environment. There are three types of such services: Lookup Service (LS) that provides information on other PerfSONAR services; Measurement Point (MP) that can perform monitoring; and Measurement Archive (MA) that stores and manages the measurement results.

8.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Path reservation	Path reservation	**Client**, AutoBAHN	On user's request the Monitoring reservation should be also done after this use case
	Monitoring reservation	**Client**, PerfSONAR	Will be next action done automatically after Path reservation
	Fetching monitoring results	**Client**, PerfSONAR	
Path resignation	Path resignation	**Client**, AutoBAHN	Monitoring resignation should be also done after this use case
	Monitoring resignation	**Client**, PerfSONAR	Will be next action done automatically after Path resignation

Figure 8.2 presents the global functional view (including all use cases).

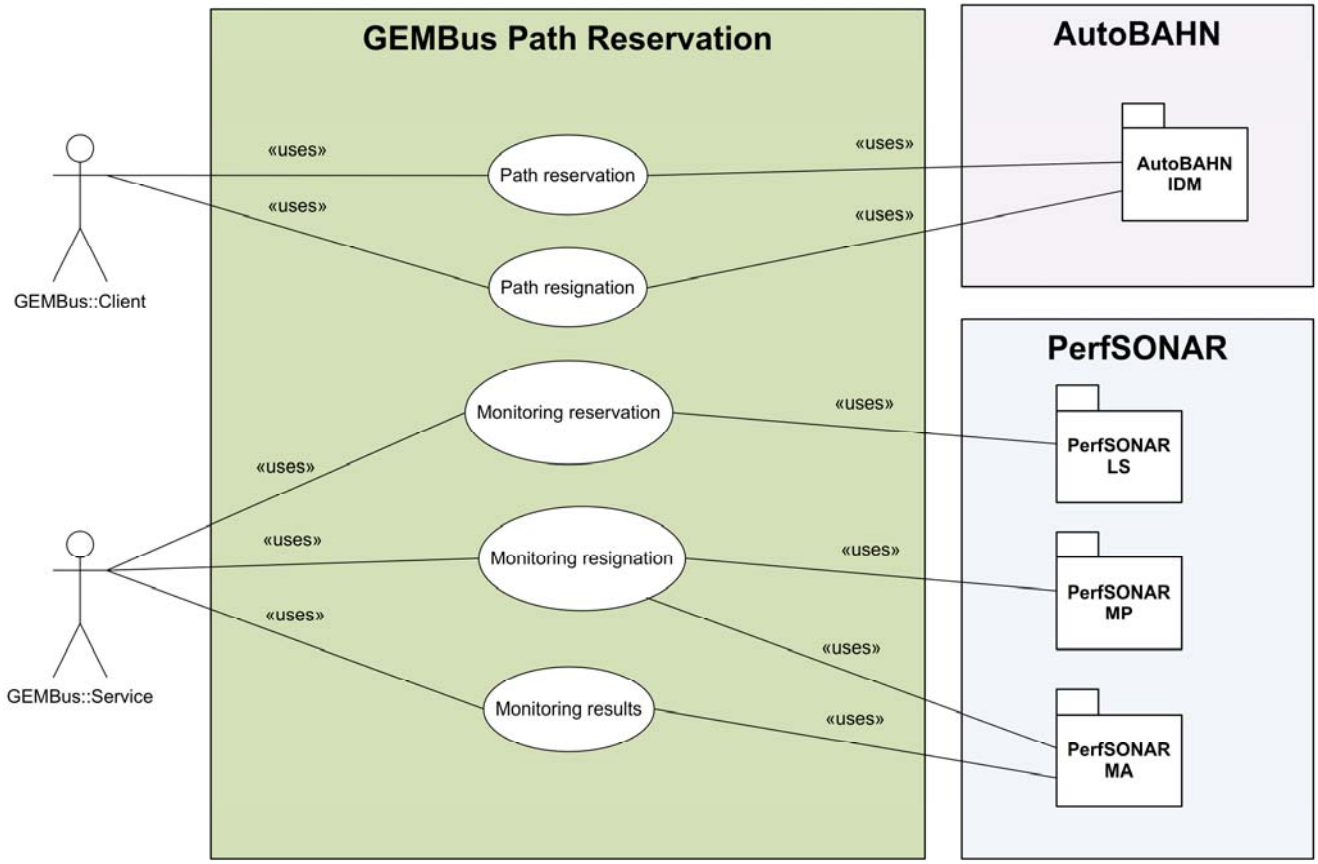

Figure 8.2: AutoBAHN/PerfSONAR integration use case diagram

8.4 Dynamic View

8.4.1 Path Reservation

Use Case	GB18 Path reservation with monitoring
Description	Clients requests the dynamic path reservation from AutoBAHN and wants some of the nodes (or all the nodes if possible) on the path to be monitored by PerfSONAR. There is also a simpler version without the monitoring, and then the monitoring may be omitted. This use case is depicted in Figure 8.3.
Actors	Clients, AutoBAHN, PerfSONAR.
Detection	Client's request.
Assumptions	PerfSONAR and AutoBAHN services up and running. There is at least one PerfSONAR Lookup Service or Global Lookup Service and AutoBAHN Inter-Domain Manager known.
Preconditions	There is a possibility to reserve the path between two requested nodes; there are Measurement Points and Measurement Archives available for monitoring the requested nodes with the requested metrics.
Steps	Step 1: Client requests setting the dynamic path reservation X and perform monitoring all the nodes with parameters Y.

Use Case	GB18 Path reservation with monitoring
	X is a set containing: x_1, x_2 – end nodes t – time p – additional parameters (bandwidth, etc.) Y is a set containing: P – list of all nodes on the path (x_1, p_0, p_1, ... p_n, x_2) with additional information on which of them should be monitored (it may be done by setting a flag on each one) p – additional measurement parameters (required metric(s), etc.) Step 2: Path reservation module contacts the AutoBAHN module and passes X. Step 3: AutoBAHN module contacts AutoBAHN IDMs in order to reserve path resources and get the information on the path topology. As soon as the path is reserved the client is responded with the path details (P) Step 4: Path reservation module tries to schedule measurement and contacts to PerfSONAR module passing P and Y. Step 5: PerfSONAR module contacts PerfSONAR Lookup Service and tries to find all the PerfSONAR Measurement Archives that store measurement results from the node list to be monitored. In addition Measurement Points may be contacted to perform the measurement with specified parameters (Y) on nodes belonging (or close) to the path (P) – if possible and implemented for specified type of metric/MP. The complete path with all MPs and MAs is returned – P'. Step 6: Client is provided with all the details and waits until the path is closed and monitoring is done. Step 7: Client contacts the PerfSONAR module in order to get results from MAs. If available, the Client may contact the specified MA in order to get partial measurement date before the path is closed.
Variations	Step 1: If not all required parameters -> error Step 3: if path can't be established -> error Step 5: if not all specified nodes to be monitored can be monitored -> warning or error Step 7: if no monitoring or monitoring fails -> do not fetch the results.
Post-conditions	Path established and monitoring done, results in a set of PerfSONAR Measurement Archives.
Extends/Includes	None
Non-Functional	None
Issues	None

This use case is depicted in Figure 8.3:

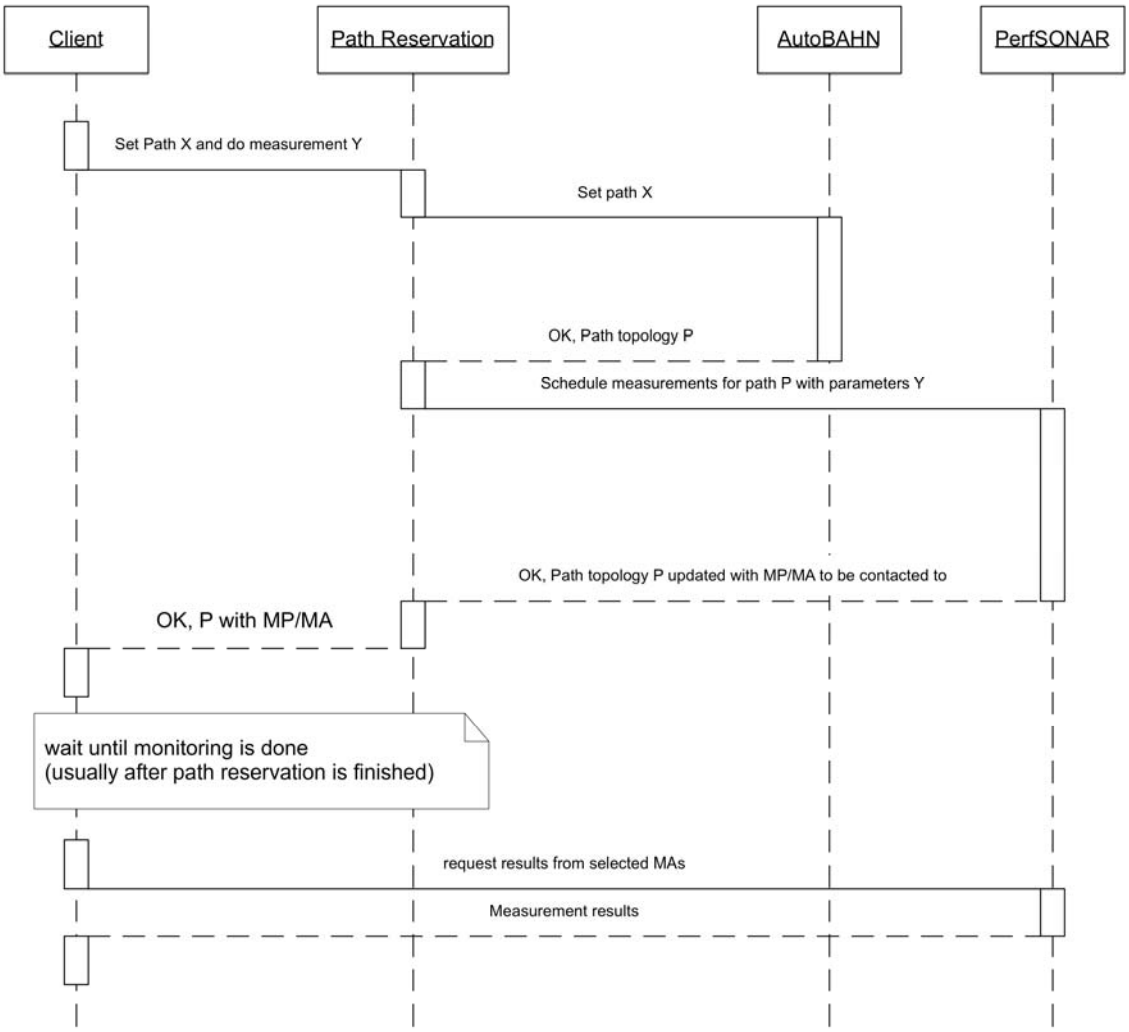

Figure 8.3: Sequence diagram for the reservation with monitoring process

8.4.2 Path Resignation

Use Case	GB19 Path resignation with monitoring resignation
Description	The previously reserved path needs to be removed This use case is depicted in Figure 8.4.
Actors	Clients, AutoBAHN, PerfSONAR.
Detection	Client's request or reservation failed.
Assumptions	PerfSONAR and AutoBAHN services up and running.
Preconditions	Path is reserved, monitoring is reserved.
Steps	Step 1: Client passes the identifier of the reserved path. Step 2: Path reservation module contacts to AutoBAHN module in order to remove the path. Step 4: Path reservation module contacts the PerfSONAR module in order to remove all scheduled measurements from the PerfSONAR MPs.

Use Case	GB19 Path resignation with monitoring resignation
Variations	Don't apply.
Post-conditions	Path is removed.
Extends/Includes	In some cases, when the reservation went wrong, this use case may extend the Path reservation use case.
Non-Functional	None
Issues	None

This use case is depicted in Figure 8.4:

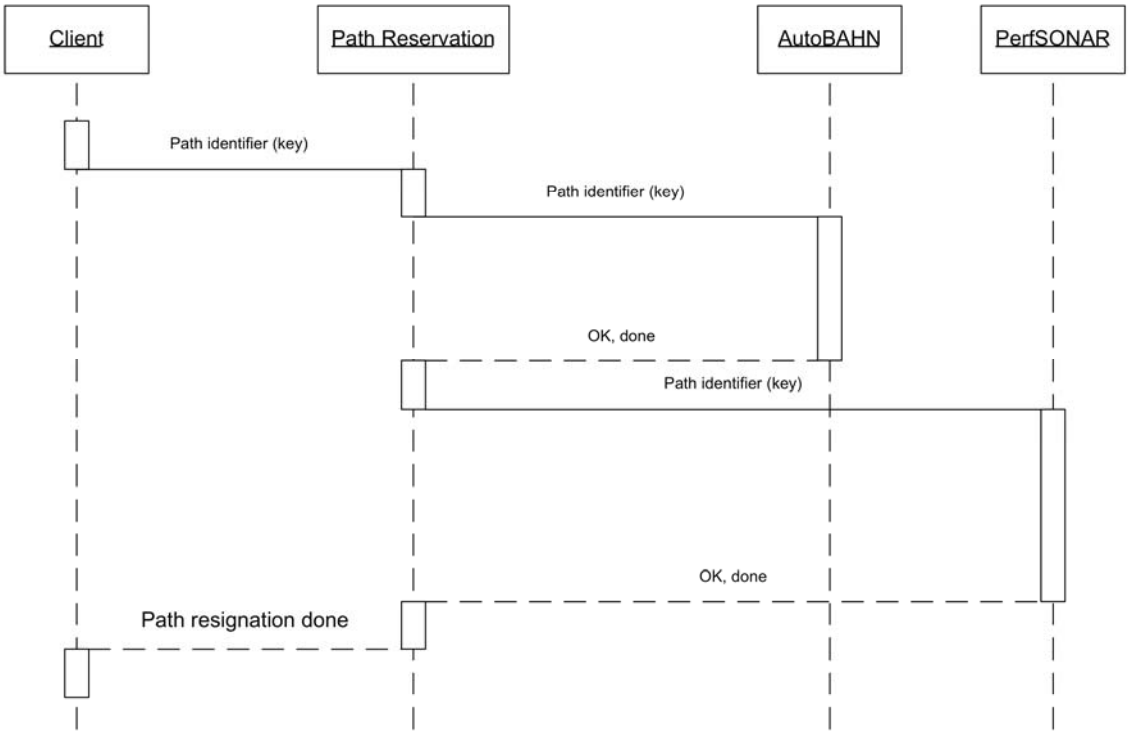

Figure 8.4: Sequence diagram for the resignation process

9 Real-time Collaboration

9.1 System-wide Functional View

Real time collaboration service (like H.323 videoconferencing) helps users to meet more efficiently by reducing the need to travel. However, there is quite a lot of work in organising and technically preparing such meetings [JANETBS].

It is assumed that almost each conferencing system could nowadays be equipped with its own reservation and management system, which usually also supports some user management functions. Such a service forms an island in the administrative domain that limits its usability. Besides the great variety of communication possibilities (H.323, SIP, jabber, Flash based systems, etc.) we should also take into consideration the many groups of users willing to collaborate cross the borders of institutions. And institutions or independent service providers are willing to provide the communication services to a broad range of users.

This use case depicts a modular system that provides a way to prepare and manage meetings for a group of users. The key word is modularity [COUNIVERSE]. We assume that there is a Group management system that can be incorporated to provide membership data. Next there are service modules providing a way to establish the required communication channels, usually from some central services. The central service (i.e. MCU) brings a larger scale multipoint communication possibility that could not be provided by endpoints.

The modular systems should also overcome the usual borders of existing reservation systems; single service orientation. Real world experience has shown that one system could be excellent for one set of channels while completely inadequate for others. For example, audio problems and small definition video can be experienced in the Adobe Connect flash-based service. These channels are provided in good quality by the H.323/SIP system, although they lack other collaboration features (like active desktop/application sharing and others). These two systems together can create interesting and very usable combinations. And there are many more services that could be used side-by-side or even interconnected. Modular design also allows the incorporation of new services as they appear, and the removal of unused services.

There are also two other services tightly coupled to conferencing. These are Recording and Streaming services, which can extend the usage of real-time systems. Even these could be introduced into such a modular system.

The key assumption for success of such a system is a precise definition of API(s) and service descriptions that will allow us to manage the services from outside systems.

9.1.1 Context Diagram

Figure 9.1 shows the context and components involved in this use-case:

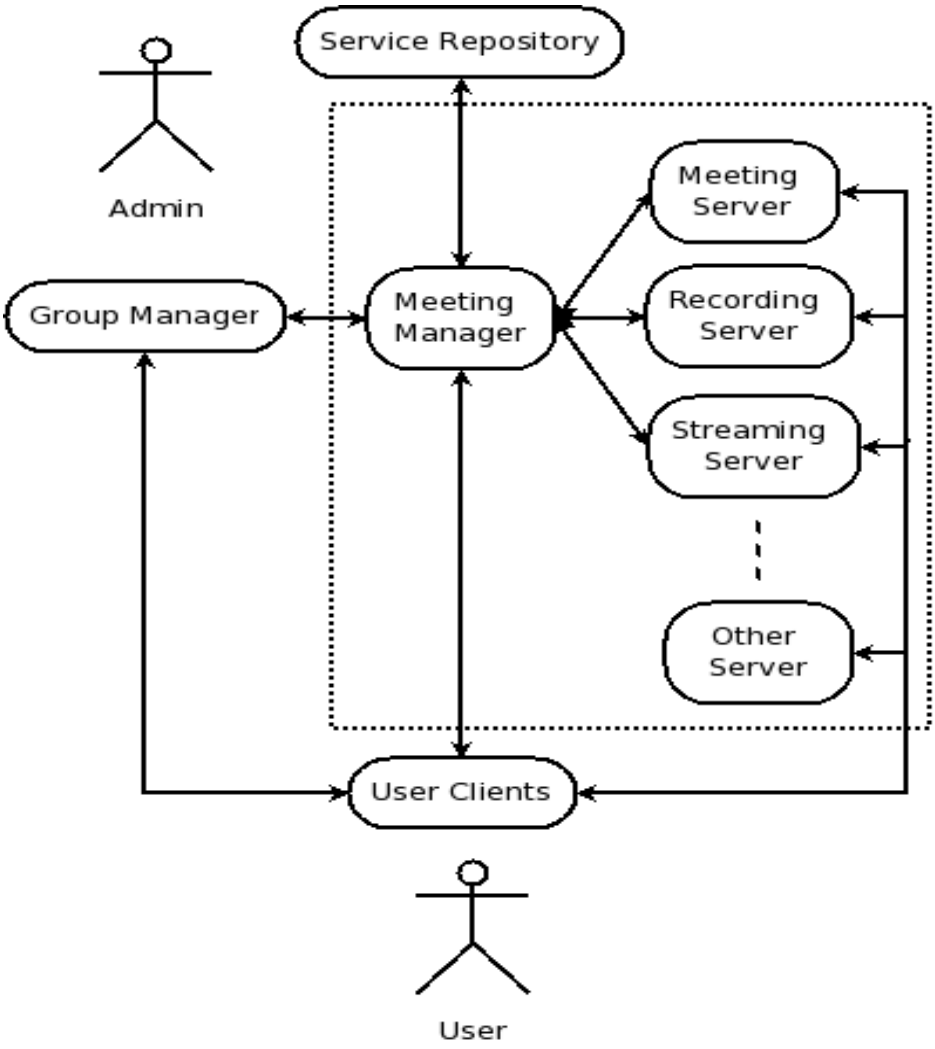

Figure 9.1: Context Diagram for the real-time collaboration use case

9.2 Actors of the System

Actor	Description
Administrator	The Administrator is a manager of the meeting process (creating, changing, etc.). It includes user role management and status monitoring. The Administrator usually will be a member of the group willing to meet, but this isn't mandatory. The Administrator can delegate any right to other users (i.e. Management of running the MCU meeting room).
Service Repository	Definition of available services (and their locations, interfaces and policies).
Meeting Manager	A set of tools able to collect user requirements and interface with the services, locating the appropriate service endpoints, deploying the appropriate components, and orchestrating their interactions. The Meeting manager provides meeting information (service identifiers like numbers, URLs, etc.) to support user interaction (User clients). Meeting manager offers only those services in the repository for which it has the right API and where its service policy allows it to be used by a certain Manager or Group.
Group Management	Provides User identities to the system, together with attributes for authorisation decisions.
Meeting Server	A system that distributes meeting channels between user's clients and other services (Recording, Streaming, etc.). Under a meeting channel can be included audio, video, text, application sharing and many more. A meeting server could be any service that provides meeting channels exchange, for example H.323/SIP MCU, Adobe Connect (Flash-based system), IM (Jabber) server and others. More than one Meeting server for a particular technology could be attached to the Meeting manager. It's up to Meeting manager to choose one according to free time slots, to migrate between them or even to request cooperation (clustering/cascading) of them.
Recording Server	A system that records meeting channels and provides offline access to recordings according to access rights. The recording server is not a mandatory part of the meeting but it is included to illustrate possible needs of users in the meeting. Some of the real world meeting servers could provide recording and streaming service but for the purpose of this uses cases the servers are decomposed.
Streaming Server	A system that provides online and/or on demand streaming of meeting according to access rights. The Streaming server is not a mandatory part of the meeting but is included to illustrate possible requirements of users in the meeting. Another complete use case could be used here as a service provider, and this use case will be consumer of its service (i.e. an artistic performance use case for streaming).

Actor	Description
Other Server	Any other Server/Service that could be used for running or supporting the meeting. A complete use case could be used here as a provider and this use case will be consumer of its service.
User Clients	Producers and receivers of particular meeting streams i.e. H.323/SIP clients, Flash-enabled browsers, IM clients.... User clients are also a web browser used to access the Meeting Manager portal, and a mail client to receive notifications and calendaring application.

9.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Meeting preparation and deployment	Location of services according to user group preferences	Administrator, Meeting manager, Service Repository	According to user group preferences, set of available services to meeting manager
	Meeting deployment	Meeting manager, Meeting server, Recording server, Streaming server, other servers, user clients	
Accessing the Meeting	Accessing the Meeting	Meeting manager, Meeting server, Streaming server, user clients	

Figure 9.2 presents the global functional view (including all use cases):

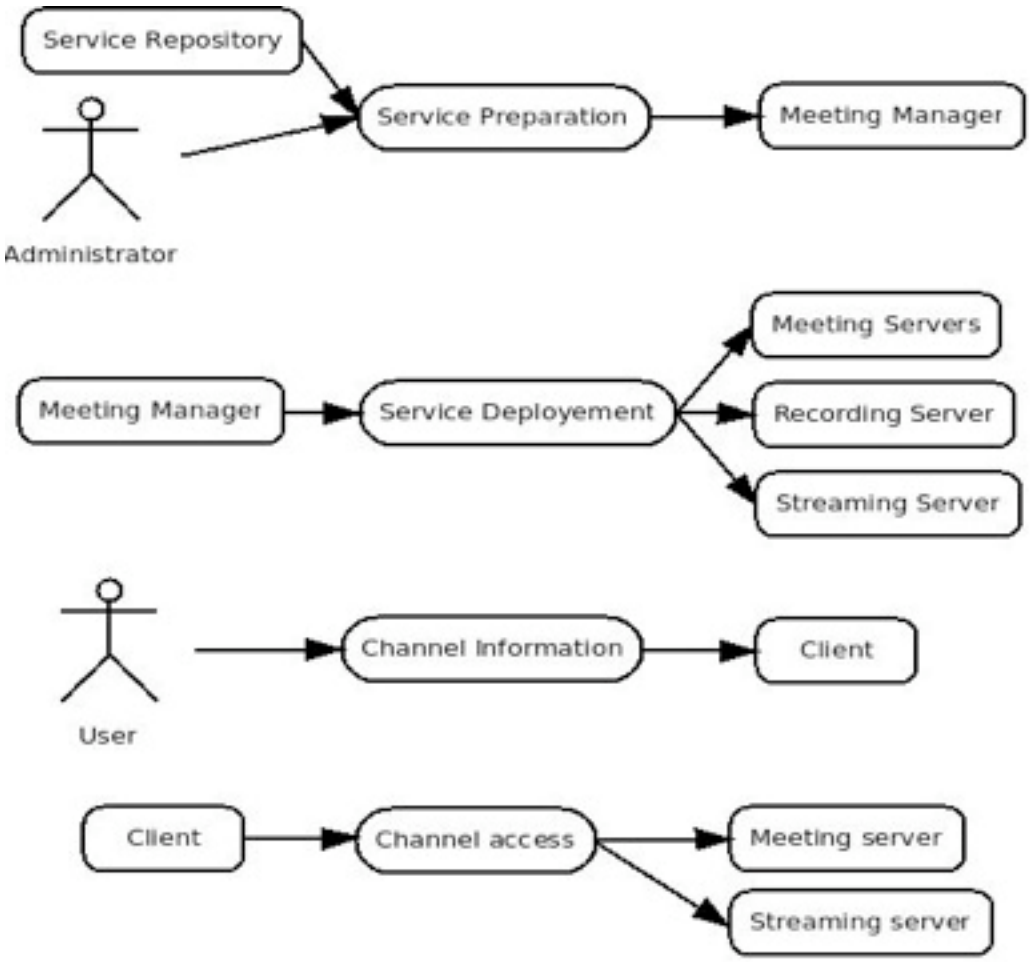

Figure 9.2: Real-time collaboration use case diagram

9.4 Dynamic View

9.4.1 Meeting Deployment

Use Case	GB20 Meeting deployment
Description	Administrator, according to Service policies, Meeting manager defines capabilities/features, user preferences, and meeting purpose and allocates the required channels and extensions. After approval by the group wishing to meet (the date is set), resources are orchestrated together to build the meeting and pass back the status and means for accessing the channels.
Actors	Administrator, Meeting manager, Meeting server, Recording server, Streaming server, other servers, user, user clients.
Detection	Meeting is available.
Assumptions	Meeting manager core service and repository available Group Management Access control infrastructure available.
Preconditions	General availability of services chosen from the repository.
Steps	Step 1: The Administrator defines the meeting channels in the Meeting Manager for the particular user group.

Use Case	GB20 Meeting deployment
	Step 2: The group chooses the date and time and confirms the channels. A defined meeting could be easily reused just by change of date. Or the meeting could be defined as recurrent. Step 3: The Meeting manager locates a set of appropriate services: Meeting server or servers to exchange the streams. Recording server if needed. Streaming server if needed. Manager checks their availability until the set is complete or error state is entered. Manager notices the Administrator whether the systems are available or channel allocation has to be modified. Step 4: The Meeting manager interconnects the set of services as needed (Meeting, Recording and Streaming server or more Meeting servers). Since there can be several independent streams in the meeting, it fetches the access information for each channel. Step 5: The Manager creates or uploads meeting roles and further information, like user's label (where required) into Servers, prepares authorisation attribute mapping between the Group manager and service, and obtains authorisation codes from service, according to particular service constraints. Step 6: Meeting information is provided to the User in the Meeting manager portal. The User is notified by mail and can upload the event into calendars or other announcement systems. Step 7: Users can change a limited set of information in their profiles and change real endpoints to meeting server rooms.
Variations	Step 2: Date and confirmation could be provided directly by the Administrator
Post-conditions	Users will be ready to access the meeting channels using their identities
Extends/Includes	None
Non-Functional	None
Issues	None

9.4.2 Accessing the Meeting

Use Case	GB21 User accesses the meeting room
Description	User gathers meeting channel access information (logs into the Meeting manager portal) that will be used by clients to access the channels. User or portal instructs the clients to access the channels.
Actors	Meeting server, Streaming server, user clients, Meeting manager.
Detection	User's decision to access set of channels defined in meeting room.
Assumptions	User clients are operational and able to access the meeting room channels.
Preconditions	Steps 1 to 6 in Use case GB20 have been accomplished.
Steps	Step 1: User logs into the Meeting manager portal where all the channel access information is presented. Step 2: User chooses the channel (repeatedly) and accesses it by (for example) copying the service identifier into the appropriate client or accessing the URL via a browser. The precise way depends on service definition and manager functionality. Step 3: Client is authorised, if required. For example, service internal protocols authorisation, PIN or by Authorisation service (Group Manager) session and attributes could be used, depending on the particular Meeting Server.

Use Case	GB21 User accesses the meeting room
	Step 4: Power users (delegated by the Administrator) can access out-of-band meeting channels management portals or will be allowed to manage meeting channels directly at the resources depending on the resource functionality
Variations	None
Post-conditions	Clients are exchanging the streams.
Extends/Includes	None
Non-Functional	None
Issues	None

10 Scientific Workflows

10.1 System-wide Functional View

Workflow-based research systems will be considered here in the framework of the Common Language Resources and Technology Infrastructure (CLARIN) project. CLARIN [3] is a large-scale, pan-European collaborative effort to create, coordinate and make language resources and technology available and readily usable. CLARIN offers scholars the tools to allow computer-aided language processing, addressing one or more of the multiple roles language plays (i.e. carrier of cultural content and knowledge, instrument of communication, component of identity and object of study) in the Humanities and Social Sciences.

CLARIN is devoted to building a persistent integrated and inter-operable infrastructure that will facilitate the access and the combination of language resources and tools/web services for those researchers that are working with language material in some form, in particular the humanities and social sciences. Even though CLARIN focuses on language material for the humanities and social sciences, their goal of providing inter-operable resources and services that can be dynamically combined and composed is a direct match with the generic goals of GEMbus; making the CLARIN project an ideal candidate for testing the GEMbus ideas and (later on) implementations, with GEMbus providing the generic technology and CLARIN using them for their specific needs.

The division between generic principles provided by GEMbus and the specific use CLARIN will make of them also shows up in the context diagram and use cases/workflow descriptions further on. While the context diagram describes a CLARIN specific scenario, the individual use cases and workflow descriptions themselves are generic in nature (since the technology for these will be provided by GEMbus).

Every CLARIN service (whether new or existing) will be accessible as a web service, which can be Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) based and should be handled by CLARIN without distinction.

Since language resources can be large, typically the resource itself is not passed in the web service call, but rather the metadata (including provenance) is passed between services, including the (storage) location of the resource to be operated on. Different types of services exist, such as tokenisers, parsers and taggers. Roughly speaking tokenisers divide language resources into logical units (or tokens), while parsers “interpret” these. Taggers provide metadata about language resources and attach these to the resources (“tag”' them). Resources are referred to using identifiers (PIDs) that can be either transient (for resources that only exist

3 http://www.clarin.eu/

temporarily, e.g. for the duration of the workflow) or persistent (for resources that will exist beyond the duration of the workflow) identifiers. These identifiers can be resolved to the location of the resource itself (the mechanism of which is beyond the scope of this document, since it is CLARIN-specific).

Profile matching is used to determine whether a service (instance) is able to process a resource, in other words: metadata for the resource and the description of the service are checked to see whether they are "compatible". After every processing step the metadata for the resource is updated by the service that processed it, so that profile matching can be repeated for the next step. Categories used in the metadata are for example ResourceFormat, AnnotationType, ApplicationDomain, MediaType, etc.

Workflow systems are used to link individual services together in a meaningful way, and should support IF/THEN/ELSE (and similar) constructs.

CLARIN relies heavily on metadata (metadata is used for all activities, such as browsing for - and describing - resources, services, tools and workflow operations), the infrastructure used for working with metadata is standardised within CLARIN, and is called the CLARIN metadata infrastructure (CMDI). The CMDI contains descriptions of all these different types of elements. The use of CMDI is CLARIN specific and its use is "hidden" behind the WebService interfaces, therefore it will not show up in the descriptions of the use cases. It is mentioned here since it aids in understanding the way CLARIN services and resources work together.

In the Language Resources and Technology (LRT) field, as in any scientific field, provenance (source, origin or history of a resource) is important to be able to reproduce results. This means that provenance is also part of the metadata stored in the CMDI.

Instead of the explanation of the processes a number of new concepts, words (and buzzwords) are used. It seems that this is a consequence of a terse compression from the full CLARIN document.

With the whole system in place, users should be able to process language resources in their own virtual workspace. For example with editors or by having different web services operate on the resources, storing the (possibly large) resulting resources either transiently or persistently, possibly using fast connections when needed.

For the use cases only a generic work flow scenario will be addressed, since that is where the involvement of AAI is most prominent and solutions there will also provide solutions for other possible use cases (such as browsing for resources, tools or services, and constructing work flows from individual services). This approach also makes sure that the requirements on GEMbus itself are generic in nature, rather than CLARIN specific CLARIN specific requirements have to be addressed by CLARIN itself, making use of the generic services provided by GEMbus.

10.1.1 Context Diagram

Figure 10.1 shows the context and components involved in this use-case:

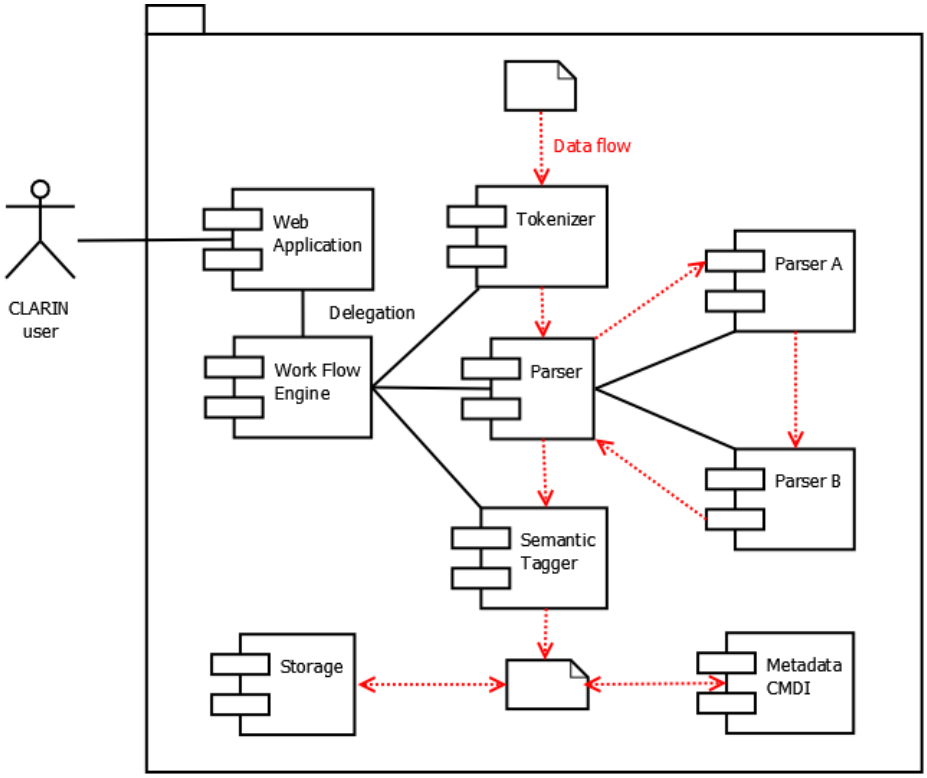

Figure 10.1: Context Diagram for the CLARIN workflow use case.

This figure shows an example of the (generic) CLARIN workflow scenario that will be used for the use cases with respect to AAI. Every link between the different components has a potential link with an AAI infrastructure, but to avoid cluttering the diagram too much these links have been omitted from the figure. In essence every step from one entity to another (typically executing a web service) will be done on behalf of the user (delegation). Since researchers in the LRT field can use resources and services from all over the world, this implies that a con-federative approach is needed. The “generic” service invocation is shown in Figure 10.2.

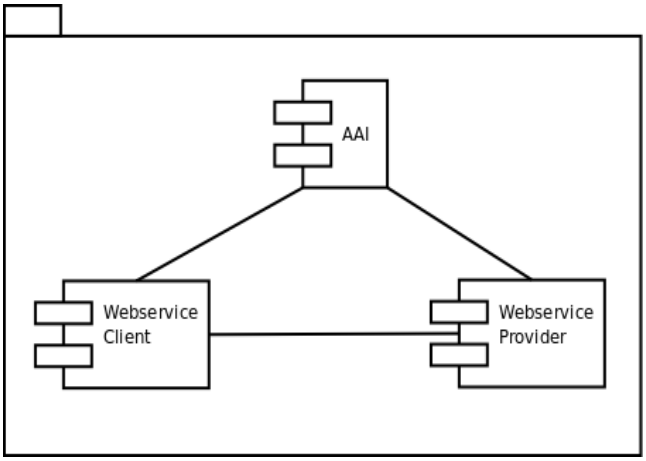

Figure 10.2: Generic service invocation.

10.2 Actors of the System

Actor	Description
Web Application	The front-end web based service that allows a user to create, modify, delete, and execute work flows.
AAI	The generic AAI infrastructure; typically a (con)federation but this also depends on the way the use cases are implemented.
Workflow Engine	Can be considered the “back-end” for the web application. Takes care of the actual execution of a (previously stored) workflow.
Web Service	A generic web service actor, the generalised form of any web service that may play a part in the use cases.
Metadata/CMDI	CLARIN MetaData Infrastructure (CMDI), which is responsible for storing all metadata related to resources and services, including provenance.
Storage	The actual storage for resources. Metadata (including provenance) is stored in the Metadata/CMDI.

10.3 Functional View: Use Cases

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
Workflow management	Execute Workflow	**Work Flow Engine**, Web Application, AAI, CMDI, Storage	
	Invoke Web Service	**Work Flow Engine**, Web Service, AAI, CMDI, Storage	
	Compose Workflow	**Web Application**, Work Flow Engine, CMDI, Storage	
	Service Discovery	**CMDI**	

Feature	Use Case	Main Actor, Secondary Actor(s)	Comments
	Resource Discovery	**CMDI**	

Figure 10.3 presents the global functional view (including all use cases).

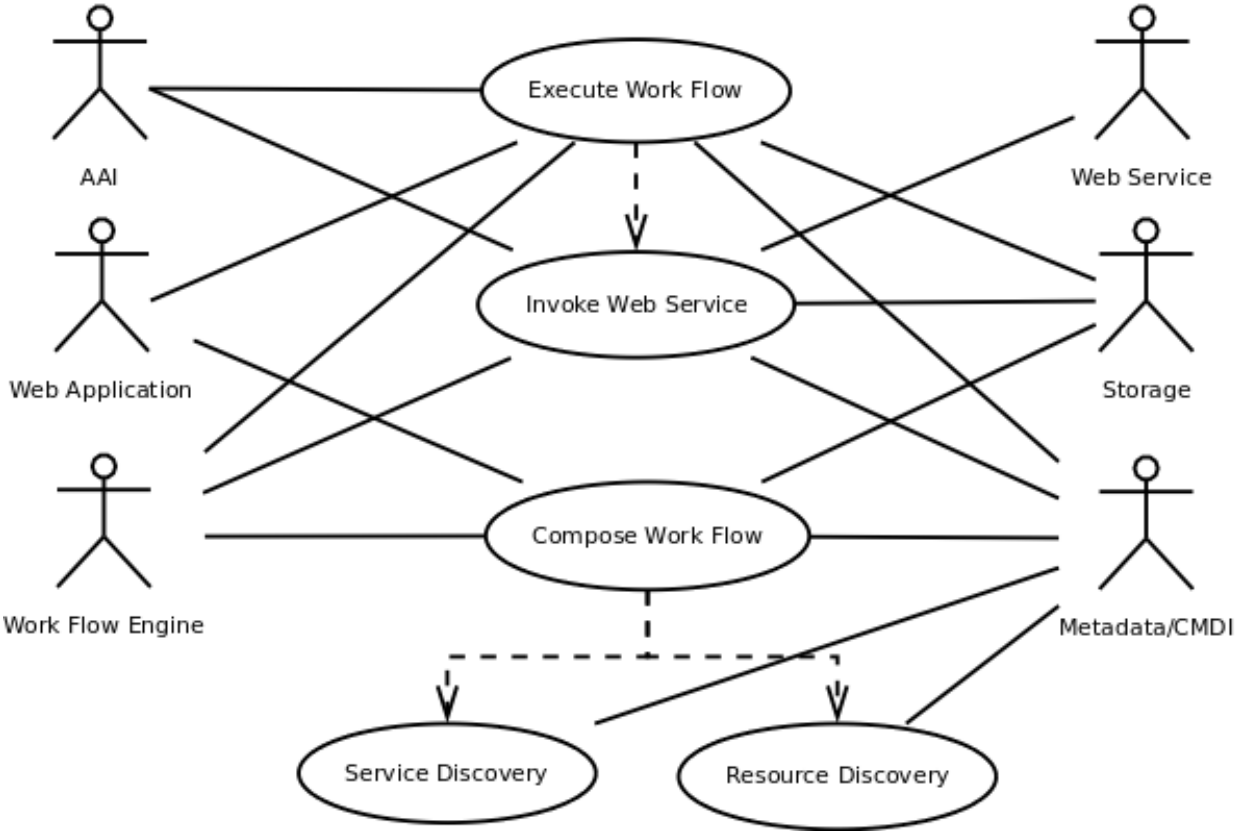

Figure 10.3: CLARIN workflow use case diagram

10.4 Dynamic View

10.4.1 Execute Workflow

Use Case	GB22 Execute workflow
Actors	Work Flow Engine, Web Application, AAI, Metadata/CMDI, Storage.
Detection	The Web Application triggers the Work Flow Engine to start the execution of a Work Flow.
Assumptions	The user is authenticated (and authorised) to use the web application and (by delegation) the WFE. A previously defined Work Flow is available to the WFE (in storage).
Preconditions	User is authenticated, a specific work flow (WF) is selected by the user for execution as well as the resource to operate on (to be used as input to the start of the WF).

Use Case	GB22 Execute workflow
Steps	Step 1: The Web Application triggers the WFE to start the execution of a workflow by passing the WF reference and the Resource reference to operate on to the appropriate WFE function. If the Authentication and Authorisation Infrastructure is a Web Service, this is done as use case "Invoke Web Service". Step 2: The Work Flow Engine retrieves the WF specification from storage (see "Invoke Web Service"). For every step in the WF, the WFE: Step 3: Checks preconditions. Step 4: Invokes the individual Web Service ('Invoke Web Service'). Step 5: Stores results and updates CMDI. After the WF finishes, the WFE: Step 6: Returns a reference to the resulting resource back to the Web Application.
Variations	Error during WF execution (TBC).
Post-conditions	The WF has finished, (reference to) result and metadata is returned to the Web Application.
Extends/Includes	Includes "Invoke Web Service".
Non-Functional	
Issues	Is transforming inputs/outputs that may be needed between steps done beforehand while composing the workflow, or are transformation needs determined at run-time? (Assumption is the former).

10.4.2 Invoke Web Service

Use Case	GB23 Invoke Web Service
Description	The actual invocation of a Web Service by a Web Service client (here typically the WFE or the Web Application).
Actors	Work Flow Engine, Web Service, AAI, CMDI, Storage.
Detection	WS clients invokes a Web Service.
Assumptions	An AAI service or front-end (middleware) is available for performing various AAI functions (checking tokens, providing tokens for use with web services).
Preconditions	The WS client has some sort of valid token for the user.
Steps	Step 1: The WFE requests an appropriate token for calling the Web Service from the AAI. Step 2: The WFE constructs the request to the Web Service. Step 3: The WFE executes the request. Step 4: The WFE evaluates the return data, updates the metadata if needed.
Variations	
Post-conditions	The Web Service executed successfully and returned the results.
Extends/Includes	Execute Work Flow.
Non-Functional	
Issues	

11 Conclusions

This document presents the definitions of several use cases for GEMBus. These use cases cover what is believed to be a wide range of application domains and are described according to a well-structured (if not fully formalised) procedure.

A specific subset of these use cases will be applied in the next phase of GEMBus development; the definition and selection of its supporting ESB framework, together with the interface requirements that individual services and its multi-domain nature impose. Furthermore, this validation process will lead to the availability of prototypes for the selected use cases, and the foundations for the implementation of all of them.

Once these foundations are available, the GEMBus team will have the tools to proceed to the implementation of the use cases described here, plus those that arise along the lifetime of the project, according to the incremental procedures described in the GN3 proposal. The formal structure established in this document for descriptions will constitute the framework for any further analysis of proposed use cases to be considered by the GEMBus team.

References

[ACSWS] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull and Massimo Mecella. “Automatic Composition of Transition-based Semantic Web Services with Messaging”, in Proceedings of the 31^st VLDB Conference, Trondheim, Norway, 2005.

[AHNET] Arts-Humanities.net
http://www.arts-humanities.net/

[ARGON] C. Barz, M. Pilz, T. Eickermann, L. Kirtchakova, O. Wäldrich, W. Ziegler,Co-Allocation of Compute and Network Resources in the VIOLA Testbed. CoreGRID Technical Report TR-0051. www.coregrid.net

[AUTC] A.G. Ganek, T. A. Corbi, "The dawning of the autonomic computing era", IBM Systems Journal, Vol 42, No 1, 2003.

[AUTCAR] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess, and J. O. Kephart, "An Architectural Approach to Autonomic Computing", International Conference on Autonomic Computing (ICAC’04).

[AUTCOV] M. Parashar and S. Hariri, "Autonomic Computing: An Overview", Springer Lecture Notes in Computer Science, volume 3566.

[COUNIVERSE] CoUniverse, a network layer for Ultragrid
https://www.sitola.cz/CoUniverse/index.php/Main_Page

[DRIVER] DRIVER, "Digital Repository Infrastructure Vision for European Research II", D2.1
http://www.driver-support.eu/linkspubs/driverpubs.html

[DSPACE] Dynamic Digital Repository: DSpace
http://www.dspace.org/

[GRID2] Ian Foster, Carl Kesselman, "The Grid 2, Blueprint for a New Computing Infrastructure"; Morgan Kauffman publishers, 2004.

[IST-PHOSPHORUS] IST Project PHOSPHORUS, 6th Framework Program of the European Union.
http://www.ist-phosphorus.eu/

[JANETBS] JANET booking service.
http://www.ja.net/services/video/jvcs/bookingservice/bookingservice.html

[KODAVIS] Collaborative Data Visualisation (formerly KoDaVis). Grid application.
http://www.viola-testbed.de/index.php?id=kodavis

[MDSWS] Rao, J., et al., A Mixed Initiative Approach to Semantic Web Service Discovery and Composition: SAP’s Guided Procedures Framework, in The IEEE Intl Conf on Web Services (ICWS’06). 2006.

[OGSA] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, J. Von Reich, "The Open Grid Services Architecture, Version 1.0"; Open Grid Forum, July 2006.
http://www.ogf.org/documents/GFD.80.pdf

[OGSAUC] I. Foster, D. Gannon, H. Kishimoto, Jeffrin J. Von Reich, "Open Grid Services Architecture Use Cases"; Global Grid Forum, 2004. http://www.gridforum.org/documents/GWD-I-E/GFD-I.029v2.pdf

[OPOB] Xavier Carreras, “Òpera Oberta (Open Opera): The Opera at the University”, PAPWS, Trieste, July 2009. http://www.garr.it/eventiGARR/papws/doc/carreras_0709_papws.pdf

[SDR] Krystyna Marek, "Scientific Digital Repositories", OGF Barcelona 2008 http://research.yahoo.com/publication/author/Zha

[SOCSRCH] Agichtein, E.; Gabrilovich, E.; Zha, H., "The Social Future of Web Search: Modeling, Exploiting, and Searching Collaboratively Generated Content", IEEE Data Engineering Bulletin, Volume 32, Issue 2, p.52-61 (2009)

[SWS] McIlraith, S.A., T.C. Son, and H. Zeng, Semantic Web Services. IEEE Intelligent Systems, 2001.16(2): p. 46-53

[UNICORE] Uniform Interface to Computing Resources. www.unicore.eu

[VIOLA] VIOLA Test-bed, Germany. www.viola-testbed.de

Glossary

AAA	Authentication, Authorisation and Accounting
AAI	Authentication and Authorisation Infrastructure
ADS	Additional Services
AM	Autonomous Manager
API	Application Programming Interface
AS	Authentication Service (perfSONAR)
ASP	Application Service Provision
CLARIN	Common Language Resources and Technology Infrastructure
CMDI	CLARIN Metadata Infrastructure
cNIS	Common Network Information Service
DRIVER	Digital Repositories Infrastructure Vision for European Research
ECMWF	European Centre for Medium-Range Weather Forecasts
ESB	Enterprise Service Bus
GEMBus	GÉANT Muti-domain Bus
GPE	Grid Programming Environment
HD	High Definition
HLA	High Level Architecture
ICT	Information and Communication Technology
IDB	Incarnation Database
IDM	Inter-Domain Manager (AutoBAHN)
IM	Instant messaging
IPR	Intellectual Property Rights
KoDaVis	Collaborative Data Visualisation
LRT	Language Resources and Technology
LS	Lookup Service
MA	Measurement Archive
MCU	Multipoint Control Unit
MP	Measurement Point (perfSONAR)
MSS	MetaScheduling Service
NSS	Network Storage Service
OGSA	Open Grid Services Architecture
PID	Persistent Identifiers
QoS	Quality of Service
REST	Representational State Transfer

SF	Sensor Framework
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol'
TS	Target System
TSI	Target System Interface
TSS	Target System Service
UML	Unified Modelling Language
UUDB	Unicore User Database
UI	User Interface
URL	Uniform Resource Locator
VO	Virtual Organisation
WF	Work Flow
WFE	Work Flow Engine
WSDL	Web Service Description Language
XML	Extensible Markup Language