GEMBus Based Services Composition Platform for Cloud PaaS

Yuri Demchenko[1], Canh Ngo[1], Pedro Martínez-Julia[2], Elena Torroglosa[2], Mary Grammatikou[3], Jordi Jofre[4], Steluta Gheorghiu[4], Joan A. Garcia-Espin[4], Antonio D. Perez-Morales[5], and Cees de Laat[1]

[1] University of Amsterdam, Amsterdam, Netherlands
{y.demchenko,c.t.ngo,delaat}@uva.nl
[2] Dept. Information and Communication Engineering, University of Murcia, Murcia, Spain
{pedromj,emtg}@um.es
[3] University of Athens, Athens, Greece
mary@netmode.ntua.gr
[4] Distributed Applications and Networks Area, i2CAT Foundation, Barcelona, Spain
{jordi.jofre,steluta.gheorghiu,joan.antoni.garcia}@i2cat.net
[5] RedIRIS, Madrid, Spain
antonio.perez@rediris.es

Abstract. Cloud Platform as a Service (PaaS) provides an environment for creating and deploying applications using one of popular development platforms. This paper presents a practical solution for building a service composition platform based on the GEMBus (GEANT Multi-domain Bus) that extends the industry accepted Enterprise Service Bus (ESB) platform with automated services composition functionality and core services to support federated network access to distributed applications and resources, primarily targeted for GEANT research and academic community. The ESB is widely used as a platform for SOA and Web Services based integrated enterprise solutions. However in existing practices ESB design is still based on manual development, configuration and integration. GEMBus with its extended functionality and orientation on distributed resources integration can be considered as a logical choice for creating cloud PaaS services composition and provisioning platform. The paper describes Composable Services Architecture that creates a basis for automated services composition and lifecycle management and explains how this can be implemented with GEMBus. The paper describes the combined GEMBus/ESB testbed and provides an example of the simple services composition.

Keywords: Cloud Platform as a Service, Services Composition, Composable Services Architecture, GEMBus (GEANT Multi-domain Bus), Enterprise Service Bus (ESB).

1 Introduction

Cloud computing [1, 2] defines three basic service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Cloud PaaS

F. De Paoli, E. Pimentel, and G. Zavattaro (Eds.): ESOCC 2012, LNCS 7592, pp. 32–47, 2012.

provides an environment for creating and deploying applications using one of popular development platforms such as current available on the market Windows Azure, Google App Engine, VMware Foundry, SaleForce.com's Force.com, Flexiant's Flexiscale, or specialised proprietary enterprise platforms.

Customers use PaaS services to deploy applications on controlled, uniform execution environments available through the network. IaaS gives a way to bind hardware, operating systems, storage and network capacity over the Internet. The cloud based service delivery model allows the customer to acquire virtualized servers and associated services. We will first discuss how a distributed service-oriented infrastructure can ease the deployment of service instances in the cloud, and how it can facilitate the usage of Cloud infrastructural services as well.

The paper introduces the GEMBus, GEANT Multi-domain Bus, being developed in the GEANT3 project JRA3 Task 3 Composable Network Services [3, 4]. GEMBus uses SOA paradigm to provide a framework to define, discover, access and combine services in the federated GÉANT multi-domain environment. It intends to span over different layers, from the infrastructure up to application elements. The GEMBus architecture is based on a general framework for composable services, founded on the industry adopted Enterprise Service Bus (ESB) [5] and extended to support dynamically reconfigurable virtualised services. GEMBus facilitates the deployment of services, supports the composition of services (spanning different management domains) and enables the automation of a particular task of business process.

The paper refers to the Composable Services Architecture (CSA) proposed by the authors that provides a basis for flexible integration of component services [4, 6]. The CSA provides a framework for the design and operation of composite services, provisioned on-demand. Since it is based on the virtualisation of component services, which in its own turn is based on the logical abstraction of the (physical) component services and their dynamic composition, it does naturally fit in the cloud distributed virtualization philosophy.

GEMBus is an interoperability and integration platform that extends the functionality of traditional enterprise-wide service-oriented architectures to a distributed multi-domain environment - therefore enabling them to be located within the cloud. It acts as enabler for new services that can be deployed in the cloud using a well-defined API, as well as integrating enterprise services and cloud based services. In this way, GEMBus intends to provide a middleware platform to support the integration of cloud-based applications, services and systems.

The paper is organised as follows. Section 2 provides general motivation for combining SOA and cloud technologies to build the advanced community oriented cloud PaaS platform. Section 3 describes Composable Services Architecture, on which the GEMBus is based, and the services lifecycle management model. Section 4 describes the general architecture of GEMBus and section 5 extends on the GEMBus component services. Section 6 provides information about GEMBus implementation status and GEMBus/ESB testbed. And finally, section 7 discusses future development of the GEMBus as a prospective cloud PaaS platform.

2 Clouds and SOA for Services Composition

There are two main directions in which mutual influence in the evolution of cloud infrastructures and Service-Oriented Architecture (SOA) [7] can translate into benefits for the maturity and usability of both technologies. First of all, service deployment and operation can greatly benefit from a supporting cloud infrastructure able to transparently provide elastically and on-demand computational and storage resources. On the other hand, cloud infrastructure services are essentially service-oriented and therefore suitable to take advantage from supporting services, such as messaging, security, accounting, composition and therefore simplifying their integration into business processes. In any of the above directions, multi-domain issues have to be considered from the beginning: service deployment in any cloud infrastructure beyond enterprise limits, as well as access to cloud interfaces out of those limits require mechanisms spanning several management domains. Other, more complicated use cases like collaborating services supported by different infrastructures, or access to different cloud providers imply much more complicated settings although they are clear application environments in the short term, if not already required.

PaaS service provisioning model [1] suggests that besides actual platform for deploying services, the PaaS platform provides also a number of automated services management functions such as remote automatic deployment, configuration of underlying platform/infrastructure services, elastic/dynamic computing and storage capacities resources allocation (by PaaS platform provider), usage statistics/accounting, and platform security such as firewalling, intrusion detection, etc.

Definition of PaaS brings benefits of creating the community oriented platform, in particular for adopted for GEANT Research and Education community in Europe. It can provide a basic set of infrastructure services and usage templates, in particular, allowing integration with campus networks. On the other hand, moving to PaaS service model will require devoted operational support facilities and staff.

3 Composable Services Architecture (CSA)

Composable Services Architecture (CSA) provides a framework for cloud based services design, composition, deployment and operation [6, 8]. CSA allows for flexible services integration of existing component services. The CSA infrastructure provides functionalities related to Control and Management planes, allowing the integration of existing distributed applications and provisioning systems, thus simplifying their deployment across network and cloud infrastructures.

CSA provides also a basis for provisioning distributed composite services on-demand by defining composable services lifecycle that starts from the service request and ends with the service decommissioning. CSA is based on the virtualisation of component services that in its own turn is based on the logical abstraction of the component services and their dynamic composition. Composition mechanisms are provided as CSA infrastructure services.

3.1 CSA Functional Components

Fig. 1 shows the major functional components of the proposed CSA and their interaction. The central part of the architecture is the CSA Middleware (CSA-MW), which supports message exchange through a normalized container that provides seamless access to a set of general infrastructure services supporting reliable and secure delivery and operation of composite services:

- A Service Lifecycle Metadata service (MD SLC) that stores service management metadata and code.
- A Registry service that holds information about service instances.
- Security services that ensure the proper operation of the infrastructure.
- Logging mechanisms able to provide operational information for monitoring and accounting purposes.

It must be noted that both logging and security services can be also provided as component services that can be composed with other services in a regular way.

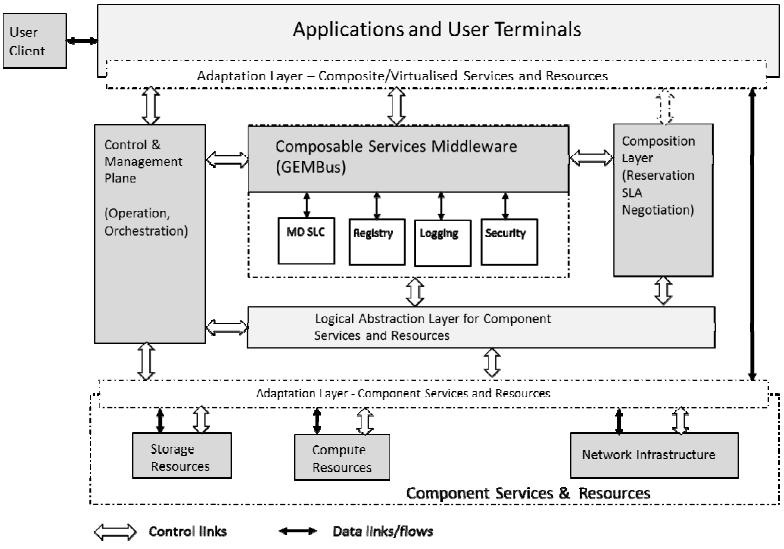


Fig. 1. Composable Service Architecture and main functional components

The Logical Abstraction Layer (LAL) defined by CSA eases service relocation across highly distributed infrastructures that can span different management domains, enabling service developers to simply fit them to satisfy the requirements to make them able to be seamlessly deployed in the cloud, as shown in "Component Services & Resources" in the diagram above. Composite services offer compatible interfaces through the Service Composition layer, which in a simple case can be provided by standard workflow management systems adapted through the Logical Abstraction Layer.

3.2 Service Provisioning Workflow and Service Lifecycle Management

While this architecture provides a good basis for creating and composing services, making them suitable to support and make advantage of the dynamical re-configurability associated with cloud infrastructures also requires them to rely on a well-defined Services Lifecycle Management (SLM) model. Most of existing services development and lifecycle management frameworks and definitions are oriented towards rather traditional human-driven services development and composition [9, 10]. Dynamically provisioned and re-configured services will require re-thinking of existing models and proposing new security mechanisms at each stage of the provisioning process.

The proposed service lifecycle includes the following main stages, depicted in the diagram (Fig. 2) below:

- **Service Request.** It relies on service metadata and supports SLA negotiation, described in terms of QoS and security requirements.
- **Composition/Reservation.** It provides support for complex reservation process in potentially multi-domain, multi-provider environment. This stage may require access control and SLA/policy enforcement.
- **Deployment.** This stage begins after all component resources have been reserved and includes distribution of the common composed service context (including the security context) and binding the reserved resources.
- **Operation.** This is the main operational stage of the on-demand provisioned services.
- **Decommissioning.** It ensures that all security contexts are terminated, and data are cleaned.

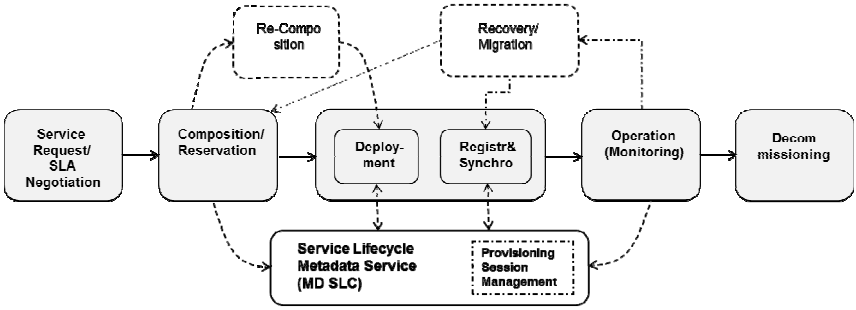


Fig. 2. Services Lifecycle Management Diagram

To take advantage of a distributed infrastructure, two additional stages can be initiated from the Operation stage based on the running service state, such as its availability or SLA requirements from its user composite services:

- Re-composition or re-planning, allowing incremental infrastructure changes.
- Recovery/Migration, can be initiated both by the user and the provider. It may also require re-composition/modification

It is important to mention that the implementation of the proposed provisioning workflow requires a number of special services to support consistent provisioned (on-demand) service life cycle management such as Service Lifecycle Metadata Service, Service Repository and Service Monitor, that should be implemented as a part of the CSA middleware.

Defining different lifecycle stages allows using different level of service presentation and description at different stages, and addressing different aspects and characteristics of the provisioned services. However, to ensure integrity of the service lifecycle management, consistent service context management mechanisms should be defined and used during the whole service lifecycle, including the corresponding security mechanisms to protect integrity of the services context. The problem here is that such mechanisms are generically stateful, what imposes problems for a SOA environment, which is defined as generically stateless. The MD SLC functional component shown in Figure 2 is intended to services lifecycle metadata.

4 GEMBus

The GEMBus framework, being developed within the GEANT project, aims to build a multi-domain service bus for the GEANT community to provide a common platform for integration of the existing and new GEANT services. With the GEMBus as a development and integration platform, new services and applications can be created by using existing services as building blocks. The foundation of the GEMBus framework includes the necessary functionality to create composite (composed) services and effectively use the widely accepted Service Oriented Architecture (SOA) to building autonomic and manageable services using the provided mechanisms for composition, adaptation, and integration of services [11, 12]. The GEMBus uses the federation approach and mechanisms for services integration and operation that is natively applicable for such multi-domain environment as GEANT community. Federation preserves management independence for each party as long as they obey the (minimum) set of common policies and technological mechanisms that define their interoperation. Metadata constitute the backbone of such federations, as they describe the components provided by each party and their characteristics in what relates to the federated interfaces.

To facilitate the GEMBus based services integration and interoperability with other cloud platforms, the core functionalities provided by GEMBus are accessible and managed via the OCCI (Open Cloud Computing Interface) [13].

The main goal of GEMBus framework is to define, discover, access, and combine network services of different layers (infrastructure, platform, service). Thus, the framework will expose to the application elements both the infrastructure-level and service-level elements. It provides the basis for a federated, multi-domain service bus designed to integrate any services within the GEANT community, and provide flexible negotiation of service provision capabilities interactively on mutual basis, avoiding centralized service deployment and management.

Fig. 3 shows the intended use of GEMBus in a cloud infrastructure. A common service registry and a service repository provide the metadata backbone for the federated SOA, together with the deployable service code. Service instances are deployed within the supporting infrastructure from the repository, allowing for re-composition, modification and migration. Description and metadata information of the deployed instant services are updated in the common registry and populated among all participating entities (services and applications). The GEMBus core supports the CSA LAL through a common messaging infrastructure plus supporting infrastructure services for security, accounting and composition.

The GEMBus core is constituted by those elements that provide the functionality required to maintain the federation infrastructure, allowing the participant SOA frameworks to interoperate in accordance with the principles previously described. The GEMBus core comprises two types of elements, combined to provide the functional elements described below according to the functionalities of the service frameworks connected to GEMBus:

- The core components that form the federation fabric, enforcing its requirements in regard to service definition and location, routing of requests/responses and security. These elements are implemented by specific software elements and by extending and profiling the service frameworks to be connected.
- A set of core services that provide direct support to any service to be deployed in GEMBus, such as the STS or the Workflow Server described below. These core services are invoked by the core elements as part of their functions. They can be called from the code implementing any service deployed in GEMBus. Furthermore, as any other service taking part in the infrastructure, they are suitable to be deployed anywhere, and integrated within composite services.

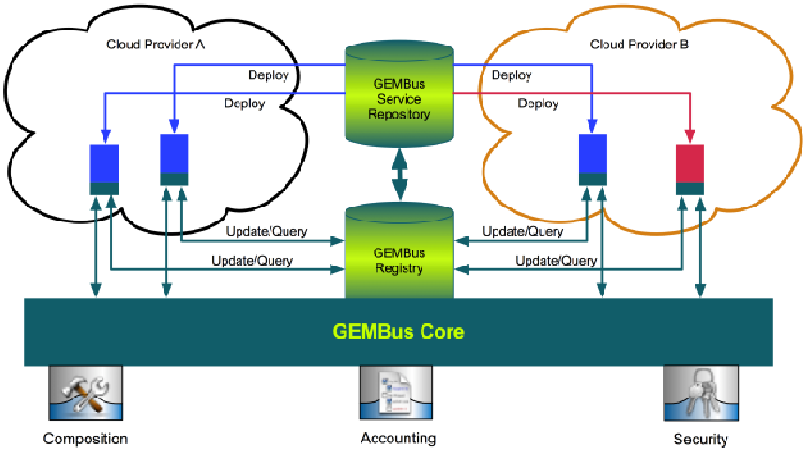


Fig. 3. GEMBus as platform for cloud services integration

GEMBus includes a set of core services that jointly constitute the GEMBus composable service platform and can be used to support user defined services:

- Federated Service Registry: stores and provides information about GEMBus services.
- Service Repository: stores service bundles, thus allowing their deployment via GEMBus.
- Composition Service: enables services composition and supported by the orchestration engine.
- Security Token Service: issues, verifies and translates security tokens to allow the authentication of requesters in a federated, multi-domain environment.
- Accounting Service: provides configurable and aggregated access to the GEMBus login service to support monitoring, auditing, diagnostics, and troubleshooting.
- Messaging service that provides underlying infrastructure for composable services interaction, integration and QoS management.

5 GEMBus Component Services

5.1 Composition Service

Being based on SOA principles, the ecosystem around GEMBus comprises a group of loosely coupled, reusable, composable and distributed services, mainly coming from the GÉANT community. Therefore, there is a need for a feasible and reliable way to compose those services to build up more complex and smarter services. This functionality is provided by the composition engine, a core service that enables GEMBus to aggregate multiple general services, as well as other composed services, into new services and applications. In summary, GEMBus allows to use existing services as building blocks of other (bigger) services with additional functionality that extends the aggregated functionality provided by the individual services.

To achieve this objective, the GEMBus framework follows existing procedures and standard SOA mechanisms, and extends them to support multi-domain operations, in particular using Business Process Execution Language (BPEL) [14] and available ESB-based workflow execution engines to enable services orchestration. They are connected with a specific description and control tool (first prototype implemented as an Eclipse plug-in) based on Business Process Modeling Notation (BPMN) [15], which is a graphical representation for Business Processes Modeling. BPMN is aimed to fill the gap between the different stakeholders that take place from the analysis of a business process to the implementation and management. It also provides a mapping to the underlying constructs of execution languages (BPEL).

The most important feature of the composition aspect offered by GEMBus is the possibility to compose any kind of services that implement standard Web Services API. For instance, the infrastructure services available in cloud IaaS may be composed with other computing services by using the OCCI standard. GEMBus

allows directly consuming the services that support OCCI by connecting them to other services through the composition engine.

In addition to the orchestration service, a workflow management system will be provided by integrating the Taverna [16] environment, a cross-platform set of tools enabling users to compose not only web services, but also local Java services (Beanshell scripts), local Java APIs, R scripts and import data from Excel or in CVS format. The GEMBus composition service allows integration of other workflow engine. Thus it is dynamically extended to meet the requirements of the specific applications and services that rely on GEMBus.

5.2 Security Token Service

The GEMBus architecture bases its operation on the Security Token Services (STS) defined in WS-Security [17] and WS-Trust [18] as a security mechanism to convey security information between services that can also be easily extended to the federated security required by the GEMBus composable services. The STS makes it possible to issue and validate security tokens such as SAML [19], JWT [20], and X.509 certificates [21]. It also supports services (identity) federation and federated identity delegation.

In the GEMBus STS, different elements support token issues and validation, as shown in Fig. 4. The Ticket Translation Service (TTS) is responsible for generating valid tokens in the system according to the received credentials, renewing and converting security tokens. Token validation is performed by the Authorisation Service (AS), which can also retrieve additional attributes or policy rules from other sources to perform the validation.

The TTS mostly relies on external identity providers that must verify the identity of the requester based on valid identification material. To support a large amount of services, the application of different authentication methods must be ensured. This must include the support of currently standardized authentication methods as well as methods incorporated in future. In this respect, there will be a direct usage of the eduGAIN identity federation service [22], or TERENA Certificate Service (TCS) [23] and other accredited identity federation service.

The AS is responsible for checking the validity of the presented tokens. In this case, the requester is usually a service that has received a token along with a request message and needs to check the validity of the token before providing a response. Checks carried out on the token can be related to issue date, expiration date or signature(s). If the token is valid, the AS provides an affirmative answer to the service. This process can also be associated with more complex authorisation processes that involves additional attributes request and authorisation policy check.

The interaction between services in GEMBus is based on message exchanges. Whether deployed inside GEMBus or running as an external service, the STS can be used in a service composition to transparently provide its capabilities.

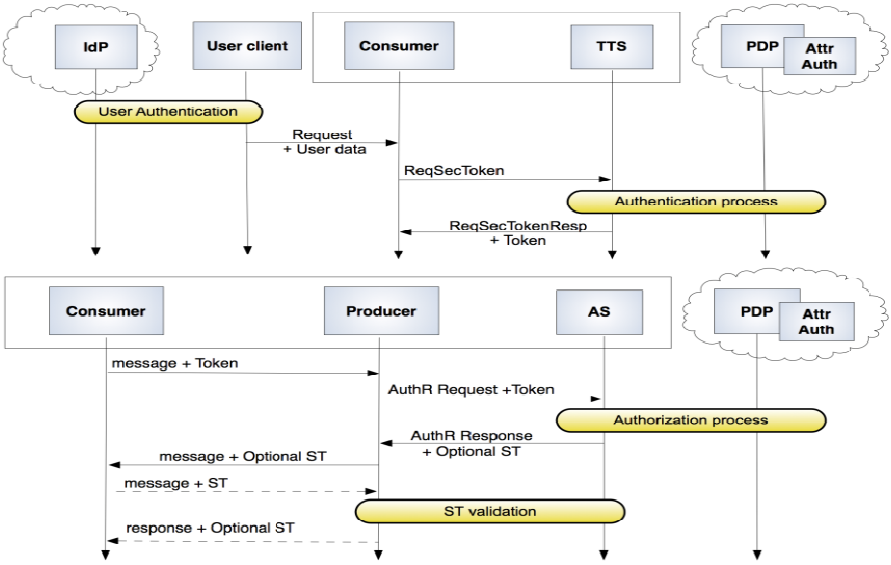


Fig. 4. Authentication and authorization processes in GEMBus security services

5.3 GEMBus Accounting Service

The GEMBus multi-domain nature requires specific mechanisms for producing and processing meaningful accounting information. The Accounting Service is deployed at every participating ESB; the collected data are stored locally, within each service instance.

The Accounting Module consists of the following main building blocks (see Fig. 5):

1. **Data Collection:** this block is in charge of collecting basic, "raw" data about the services. Data collection is done at the ESB level. This is an asynchronous operation, as it is triggered each time a service is being called. In particular, the services integrated in GEMBus use the message-oriented middleware infrastructure to communicate. The function of this block is to capture every message exchanged between services, as those messages are precisely the source of information to evaluate GEMBus services behavior and performance.
2. **Data Storage:** this component stores the raw data. This operation occurs in conjunction with data collection; in other words, the collection operation is always followed by the operation of storing the collected raw data locally. Consequently, this is an asynchronous operation as well.
3. **Data Processing:** this block computes the metrics of interest related to each service. In order to fulfill this operation, the Accounting Service must be aware of the metrics it should compute for each of the services registered with the system. Which metrics are appropriate for a particular service or how to define them is out-of-scope of our work. Instead, we assume the metrics are either part of a basic set provided by the Accounting Service itself, or they are specified by

the service provider. Data processing occurs asynchronously; by that we refer to the fact that the raw data is processed only when a request is received. Additionally this module performs aggregation of accounting data from all available domains.

4. **Information Reporting:** this component is responsible for producing a report in a human-readable format. Once the data processing has been concluded, the information reporting block produces a report containing a complete view, including data obtained across all available ESBs.

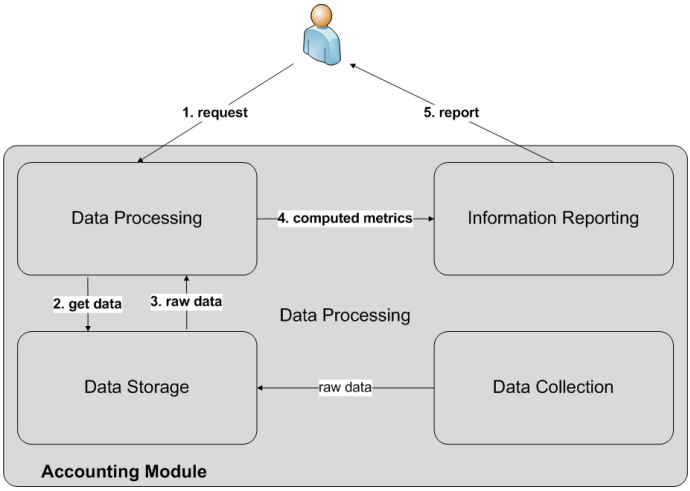


Fig. 5. Accounting service architecture

In order to support the multi-domain nature of GEMBus, the Accounting Service must allow inter-domain data aggregation. To this end, each Accounting instance informs the Registry on the services about which it has collected raw data. Next, when a request about a specific service is received at one of the Accounting Services, it will query the Registry for the list of other Accounting instances with data on that service. Once the Registry returns the list, the Accounting will contact the other instances to obtain the data of interest, which will further be aggregated with the local data and the resulting report will be presented to the issuer of the request.

Let us give a short example to explain the workflow in the Accounting Service, also illustrated in Fig. 5. We consider a situation when a system user or system administrator from domain A/ESB A would like to get information on a certain service, denoted by Service X. In the following, we use the term “local” to refer to the Accounting System receiving the request, and the term “remote” to refer to other instances that have collected data about the service under consideration.

The user/administrator issues a request to the local Accounting Service (Step 1 from Fig. 6). Next, the Accounting Service from domain A obtains from the Registry the list of remote instances (i.e. the Accounting Service from domains B and C) with information on Service X (Steps 2 and 3), and contacts each of them (Steps 4 and 5).

At the same time, the Data Processing component from the Accounting Service in domain A retrieves from the Data Storage the raw data collected locally , and it computes the metrics of interest. Once the reports from the remote Accounting instances arrive, all data are aggregated and passed to the local Information Reporting component. This block will further produce a final report, to be delivered to the system user or system administrator who issued the request in the first place (step 6).

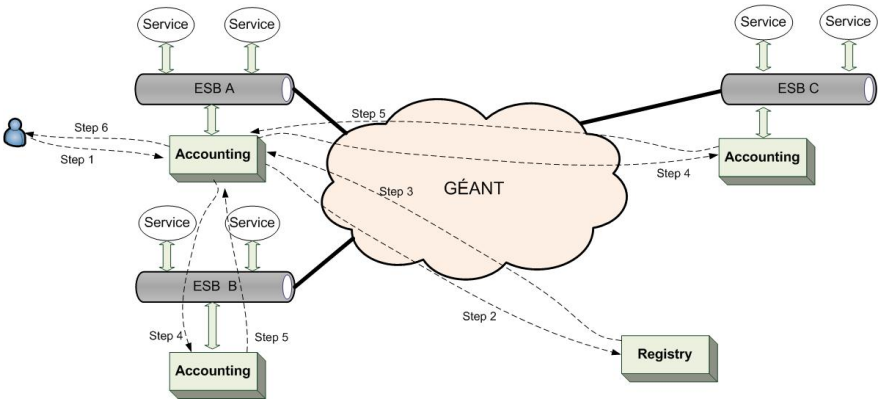


Fig. 6. Workflow in the Accounting Service

6 Testbed for ESB Based PaaS Platform

The proposed solutions and GEMBus/ESB based platform for services composition have been implemented as a cloud PaaS tested at University of Amsterdam. The testbed provides a facility for testing technologies and developing them as an open cloud PaaS platform.

Fig. 7 shows the testbed structure and implementation details. The lower layer infrastructure uses OpenNebula Virtual Machines (VM) management environment [24]. Each VM runs a Fuse ESB [25] instance that may host one or more services that can be provided and deployed as OSGi bundles [26].

Services interconnections is realised based on such ESB functional components as Message Broker (based on Apache ActiveMQ [27]) and Message Router (based on Apache Camel). Component services can be deployed in ESB environment using VM's with preinstalled and pre-configured ESB instances. Final services interconnection topology can be created by pre-configuring ESB instances or dynamically changing their configuration after deployment and during run-time, what is supported by ESB functionality.

Communication between GEMBus domains is done either over underlying transport network infrastructure or using dedicated network infrastructure provisioned as Network as a Service (NaaS). In the latter case NaaS can controlled via dedicated GEMBus service. Current testbed implementation uses only underlying transport network infrastructure.

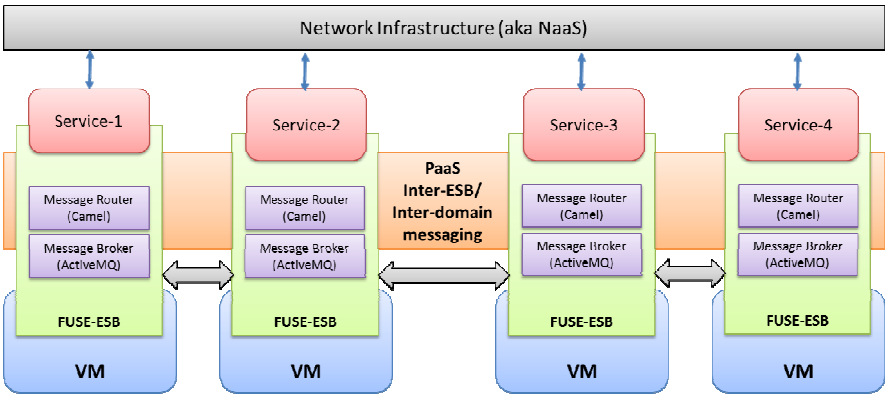


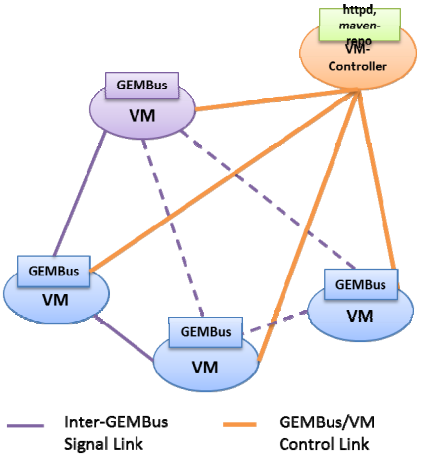
Fig. 7. Testbed for GEMBus/ESB based services composition as a cloud PaaS

In the proposed testbed, the deployed services which can be classified into two types. The first one is data generator service type, which automatically generates data wrapped inside messages on a regular basis. The second service type is data processor, which can process incoming data from different sources. The configurable routing mechanism allows to define data flows from data generator services to data processor services. All services are deployed in separated VMs that can be distributed between different hosting computers and physical locations. They can be dynamically connected into single logical topology by network of brokers as shown in Fig. 8. Theoretically, the testbed could allow to deploy as many services as possible, not only in existing VMs, but also by extending network of brokers when provisioning new VMs.

Fig. 8 provides graphical illustration of the services topology realization using Message Router and network of Brokers. The listing provides example of Message Broker configuration in bean.xml for a simple demonstrator shown in Fig. 9.

Current testbed configures that service data generators produce data messages every second. The service data processor receives data messages from different sources and processes them to produce new data and send them to the visualization output. Problems relating to message transport such as delay and message ordering can be illustrated by the visualization result. For example, with the service data generators to produce sin signal and square signal, and the service processor realising an addition function engine, the output is a combination of sin and square signals. The demonstrator helped to reveal the importance of services synchronisation and controlling message sequencing in a distributed environment due to possible communication delays and even changed sequence of messages arrival.

The ongoing testbed development addresses discovered services synchronisation and load balancing issues which are particular important for industry oriented applications. Some known approaches deal with these problems are using message processing mechanisms available in ActiveMQ Message Broker and Camel Normalised Message Router, but their dynamic deployment and configuration in multi-domain environment remain a subject of the future research.




```
<route>
    <from uri="jms:S1_Out"/>
    <to uri="jms:S3_Out"/>
</route>
<route>
    <from uri="jms:S2_Out"/>
    <to uri="jms:S3_Out"/>
</route>
<route>
   <from
uri="jms:queue:S3_Out"/>
   <to
uri="bean:logger?method=log”/>
</route>
```

Fig. 8. Message Router and network of Brokers and example of bean.xml configuration

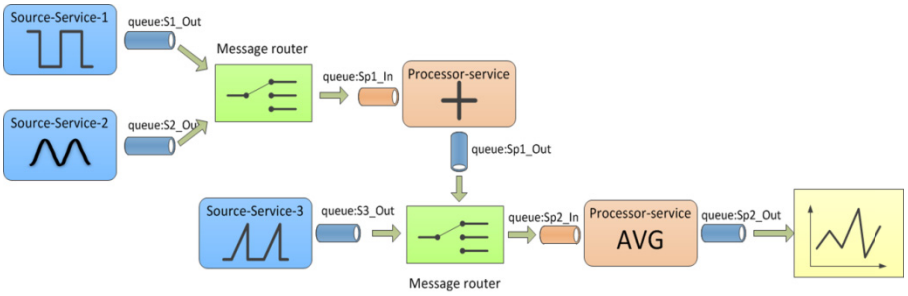


Fig. 9. GEMBus/ESB services composition Demonstrator.

7 Conclusion and Future Development

In this paper we described the architecture and initial implementation of the GEMBus framework as a generic SOA service bus implementing the proposed in the GEANT project Composable Services Architecture (CSA). Besides serving as a middleware platform for CSA, the GEMBus offers composition and orchestration services for automated services creation, deployment, and execution. The paper also describes how the GEMBus can be used as a cloud PaaS platform.

It is important to mention that GEMBus is being developed as a part of the GÉANT project, so it inherits a wide base of already developed services and the user community of all European National Research and Education Networks (NREN) which is currently positioned as a federated community of independent networks and service operators. GEMBus provides all the necessary functionality to integrate services from different domains and resolve the inter-domain issues. The future work will include the deployment of a GEANT wide GEMBus testbed to be provided as a

cloud PaaS service that will allows creating new services and applications and integrate them with the basic GEANT infrastructure services such as AutoBAHN [24] and PerfSONAR [25] which correspondingly provide the bandwidth on-demand service and the multi-domain monitoring service for the GÉANT network. This will facilitate exchange of the community developed services and disseminate best practices among GEANT members.

Finally, it is planned that the research results presented here will be contributed to the Open Grid Forum Research Group on Infrastructure Services On-Demand provisioning (ISOD-RG) [30], where the authors play an active role.

Acknowledgement. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007-2013) under Grant Agreement No. 238875 (GÉANT).

References

[1] NIST SP 800-145, A NIST definition of cloud computing, http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

[2] NIST SP 500-292, Cloud Computing Reference Architecture, v1.0, http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf

[3] GEANT Project, http://www.geant.net/pages/home.aspx

[4] GN3 Project JRA3 Task 3 Composable services, http://www.geant.net/Research/Multidomain_User_Application_Research/Pages/GEMBus.aspx

[5] Chappell, D.: Enterprise Service Bus. O'Reilly (June 2004)

[6] Grammatikou, M., Marinos, C., Demchenko, Y., Lopez, D.R., Dombek, K., Jofre, J.: GEMBus as a Service Oriented Platform for Cloud-Based Composable Services. In: Proc. 3rd IEEE Conf. on Cloud Computing Technologies and Science (CloudCom 2011), Athens, Greece, November 29-December 1 (2011) ISBN: 978-0-7695-4622-3

[7] OASIS Reference Architecture Foundation for Service Oriented Architecture 1.0, Committee Draft 2 (October 14, 2009), http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf

[8] Generic Architecture for Cloud Infrastructure as a Service (IaaS) Provisioning Model, Release 1. SNE Techn. Report SNE-UVA-2011-03 (April 15, 2011), http://staff.science.uva.nl/~demch/worksinprogress/sne2011-techreport-2011-03-clouds-iaas-architecture-release1.pdf

[9] Demchenko, Y., van der Ham, J., Ghijsen, M., Cristea, M., Yakovenko, V., de Laat, C.: On-Demand Provisioning of Cloud and Grid based Infrastructure Services for Collaborative Projects and Groups. In: The 2011 International Conference on Collaboration Technologies and Systems (CTS 2011), Philadelphia, Pennsylvania, USA, May 23-27 (2011)

[10] TMF Service Delivery Framework, http://www.tmforum.org/servicedeliveryframework/4664/home.html

[11] Martinez-Julia, P., Lopez, D.R., Gomez-Skarmeta, A.F.: The gembus framework and its autonomic computing services. In: Proceedings of the International Symposium on Applications and the Internet Workshops, pp. 285–288. IEEE Computer Society, Washington, DC (2010)
[12] Martinez-Julia, P., Marin Cerezuela, A., Gomez-Skarmeta, A.F.: A service oriented architecture for basic autonomic network management. In: Proceedings of the IEEE Symposium on Computers and Communications, pp. 805–807. IEEE Computer Society, Washington, DC (2010)
[13] GFD.183: Open Cloud Computing Interface – Core. Open Grid Forum, http://ogf.org/documents/GFD.183.pdf
[14] OASIS Web Services Business Process Execution Language (WSBPEL), http://www.oasis-open.org/committees/wsbpel/
[15] Business Process Modelling Notation (BPMN), http://www.bpmn.org/
[16] Taverna, http://www.taverna.org.uk/
[17] WS-Security, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
[18] WS-Trust, http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html
[19] Cantor, S., et al.: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 (SAML Core). OASIS Standard (2005)
[20] Jones, M., et al.: JSON Web Token (JWT) Network Working Group, Internet Engineering Task Force (IETF) (December 2011), http://tools.ietf.org/html/draft-jones-json-web-token
[21] Lawrence, K., Kaler, C.: Web Services Security: X.509 Certificate Token Profile 1.1. Web Services Security (WSS) (November 2006)
[22] eduGain – Federated access in GEANT services network, http://www.geant.net/service/edugain/pages/home.aspx
[23] TERENA Certificate Service, http://www.terena.org/activities/tcs/
[24] OpenNebula, http://opennebula.org/
[25] FUSE ESB Platform, http://fusesource.com/products/enterprise-servicemix/
[26] OSGi Service Platform Release 4, Version 4.2, http://www.osgi.org/Download/Release4V42
[27] Apache ActiveMQ Performance, http://activemq.apache.org/performance.html
[28] AutoBAHN Bandwidth on-demand provisioning tool, http://www.geant.net/service/autobahn/pages/home.aspx
[29] PerfSONAR Multidomain monitoring service for GEANT service area, http://www.geant.net/service/perfsonar/pages/home.aspx
[30] Open Grid Forum Research Group on Infrastructure Services On-Demand provisioning (ISOD-RG), http://www.ogf.org/gf/event_schedule/index.php?event_id=17