NEXTGRID ARCHITECTURAL CONCEPTS

David Snelling
Fujitsu Laboratories of Europe Limited, Hayes, Middlesex, UB4 8FE, United Kingdom
david.snelling@uk.fujitsu.com

Ali Anjomshoaa, Francis Wray
EPCC, University of Edinburgh, Edinburgh, EH9 3JZ, United Kingdom
ali@epcc.ed.ac.uk, f.wray@epcc.ed.ac.uk

Achim Basermann
NEC Europe Limited, C&C Research Laboratories, D-53757 Sankt Augustin, Germany
basermann@ccrl-nece.de

Mike Fisher
BT Group Chief Technology Office, London, EC1A 7AJ, United Kingdom
mike.fisher@bt.com

Mike Surridge
IT Innovation Centre, Southampton, SO16 7NP, United Kingdom
ms@it-innovation.soton.ac.uk

Philipp Wieder
Central Institute for Applied Mathematics, Research Centre Jülich, 52425 Jülich, Germany
ph.wieder@fz-juelich.de

Abstract This paper outlines the conceptual model of the NextGRID architecture. This conceptual model consists of a set of architectural principles and a simple decomposition of the architecture in order to facilitate common understanding of the architecture and its development.

Keywords: Grid Architecture, Service Level Agreement, Service Grid.

1. Introduction

The NextGRID project vision is of future Grids, which are economically viable; in which new and existing business models are possible; in which development, deployment and maintenance are easy; and in which the provisions for security and privacy give confidence to businesses, consumers and the public.

The goal and primary output of NextGRID is to define the architecture of the Next Generation Grid. This will prepare the way for the mainstream use of Grid technologies and their widespread adoption by organisations and individuals from across the business and public domains. In addition to the design of architectural Grid concepts, the NextGRID architecture will facilitate the development of key middleware components, application support mechanisms, know-how and standards that underpin the Next Generation Grid.

2. NextGRID Architectural Principles

The NextGRID architectural principles define the overall characteristics of the NextGRID architecture and outline its key components. These principles define the *personality* of the NextGRID architecture.

The primary architectural principles of the NextGRID project are:

Service Level Agreement Driven Dynamics: All interactions in NextGRID are predicated by a Service Level Agreement (SLA) that is dynamically created and aims to ensure that the relationship between a provider and a consumer is well defined and understood. This SLA based approach applies to all service interactions, thereby providing a uniform framework for the management and operation of all Quality of Service (QoS) aspects.

Service Construction and Composition: As a dynamic Grid infrastructure, NextGRID provides extensive capabilities for service construction and composition. This includes traditional interface composition, various forms of workflow-enabled orchestration, and support for the dynamic extension of service capabilities.

Minimal Service Infrastructure: All services operating in a NextGRID environment can expect to find a minimal service infrastructure. This infrastructure is manifested as a set of capabilities, such as service lifetime management or service registries, which are either available in the environment or exhibited by peer services.

2.1 Service Level Agreement Driven Dynamics

A successful NextGRID architecture will have a number of stakeholders, ranging from the large multi-national enterprise organisations, down through the large nationally based enterprises, service providers, small and medium sized enterprises, academic institutions and individual end users. Interactions will most likely involve a combination of these parties.

A SLA covers the entire lifecycle of the interaction with a service provider, from the negotiation of the QoS that the consumer can expect, through to the deployment, execution and monitoring of the service to decommissioning.

2.1.1 Overview of Service Level Agreements. NextGRID believes that SLAs should be used to build relationships between service providers and con-

sumers. Neither the service provider nor the consumer will gain a significant advantage by violating a SLA. The customer will not get the service they require, and the provider's reputation will be damaged. It is proposed, therefore, to have a framework that is less focused on monitoring of every element of every transaction in isolation, but is rather more focused on providing an overall level of service in terms of the business being carried out.

We believe that a SLA is a key component to be considered at all stages in the lifecycle of a service provision. The policies for managing the service, the mechanisms for monitoring it, and the acceptable quality of service terms to offer to a consumer should be produced at the same time as the service is designed and developed. This ensures that the required information is available to be able to guarantee the QoS levels necessary, such that a consumer will consider entering into an agreement with a provider to use a service.

2.1.2 SLA Structure and Contents. A SLA exists between two parties, the service provider and the consumer. By building a robust and non-ambiguous SLA framework, the need for trusted third parties, who provide independent verification of monitoring information to give confidence to the consumer, can be reduced and replaced with the provider and consumer performing their own monitoring in a mutually trusting way.

Therefore, a considerable amount of work in NextGRID has been focusing on the structure of the SLA, so it can provide all the information that other components require, in a standard, structured way that allows for automated and more economic processing. We see the SLA as containing not only information relating to the specific guarantees offered on the performance of the service, what we categorise as *dynamic terms*, but also relating to the commercial due diligence terms, which we categorise as *static terms*.

Static terms describe the policies in place in the environment in which the service will be deployed and executed. They are less likely to change between many SLAs between two parties. In dynamic terms, we identify higher level terms, which are closer to those understood by consumers or applications. Guarantees are offered on these terms.

Service levels must be defined in terms of the value delivered to the customer. It would be a bad idea to reveal what computational resources would be used to deliver a service, as these suggest a much lower value to the customer. Of course, the service provider has to know how to manage their resources to deliver the specified results, and what the business-level consequences will be if they experience a resource shortfall.

To make this work, mapping mechanisms are needed as shown in Figure 1: to translate business-level objectives defined in a SLA into resource management policies that can be applied at the technical level within the service provider's environment, and to translate technical-level monitoring information into busi-

ness level consequences that can be compared with a SLA, and used to provide meaningful feedback to the consumer.

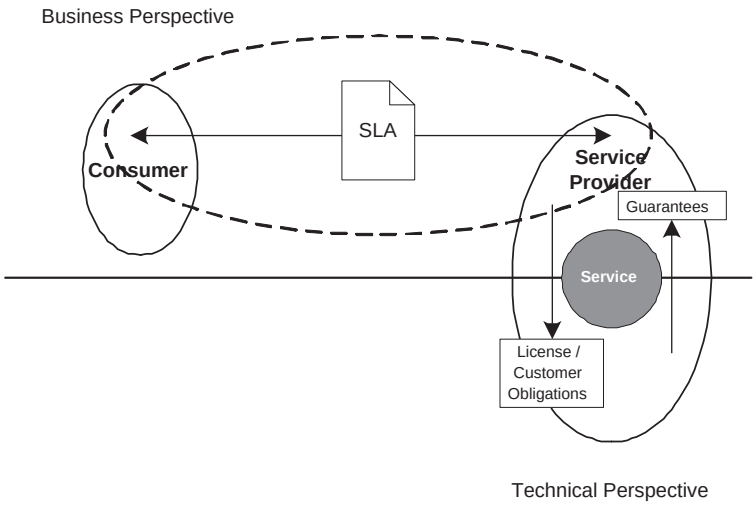

Figure 1. Business level SLAs and technical resource management are related, but logically separated into a business perspective and a technical perspective, respectively.

2.1.3 Protocol. Negotiation of a SLA should be as flexible as possible, but at the same time aligned with the negotiated service's lifetime. It is counterproductive to use a protocol needing a longer time span to negotiate than is expected for performing the requested service.

To keep the negotiation effort as low as possible, NextGRID employs a discrete offer protocol: the service provider offers the service customer some services (e.g. Services A, B and C), from which the service customer has to choose one. There is no scope for negotiation as the parameters of the offered services are fixed. In a symmetric fashion, the customer may also make the offer and have it accepted or rejected by the provider.

2.2 Service Construction and Composition

The NextGRID architecture is intended to support rapid and dynamic federation of resources to support user communities. Architecturally, we assume that applications may be constructed by composing NextGRID services, each of which has a set of common properties and behaviours. When executing applications, we can assume that certain core infrastructure services or properties are available in the environment of the application. A key requirement is that such federation mechanisms should result in architecturally self-similar struc-

tures that are themselves amenable to NextGRID composition rules, leading to an environment that enables recursive service composition.

The basic modes of service composition are:

Resource sharing: arises when the consumer of a service shares it with another consumer. Resource sharing is strictly a federation between consumers. It makes the consumers part of a related set of interactions as seen by the service provider. Resource sharing is very important for business Grids.

Resource orchestration: arises when a consumer of two services asks them to interact in some fashion. This process effectively combines resources from two service providers to meet the needs of the common consumer.

Resource encapsulation: arises when a service provider delivers a service to a customer through a third party service provider, with no direct interactions between the third party service provider and the consumer.

2.2.1 Implications for SLAs. Resource sharing and orchestration both involve the creation of new bilateral relationships with a service, which are initiated by an existing consumer. Every bilateral relationship should be governed by a SLA. Our investigations suggest that it should be possible to automatically infer the terms of a new SLA from the terms of original SLAs in place with the consumer. Resource encapsulation does not impose requirements on individual SLAs, but has implications for the overall SLA architecture.

Figure 1 shows that there should always be a mapping between the terms of a SLA related to a service, and the technical management policies and actions needed to deliver that service. The view of encapsulation as a resource pattern then becomes useful in the design of SLAs and for SLA management mechanisms. Instead of using a single mapping mechanism directly from the business level to the resource level, one can introduce intermediate level services and simplify the mappings at each stage.

Figure 2 shows an example of this approach, in which 4 distinct levels are identified. Here the communication (and agreement) between a service consumer and a service provider is on the business level. Instead of mapping directly to the fabric (computational resource, disk space, networks, etc), this service is provided by encapsulating other services, each encapsulation being governed by its own SLA. The management policies specify the requirements to be met by SLAs from the layer below and the monitoring and corrective action to be used to detect and recover from any breaches of those SLAs.

2.3 Minimal Service Infrastructure

The key aspects of a minimal Grid infrastructure lead to a minimal set of expected Grid service behaviours. These aspects are:

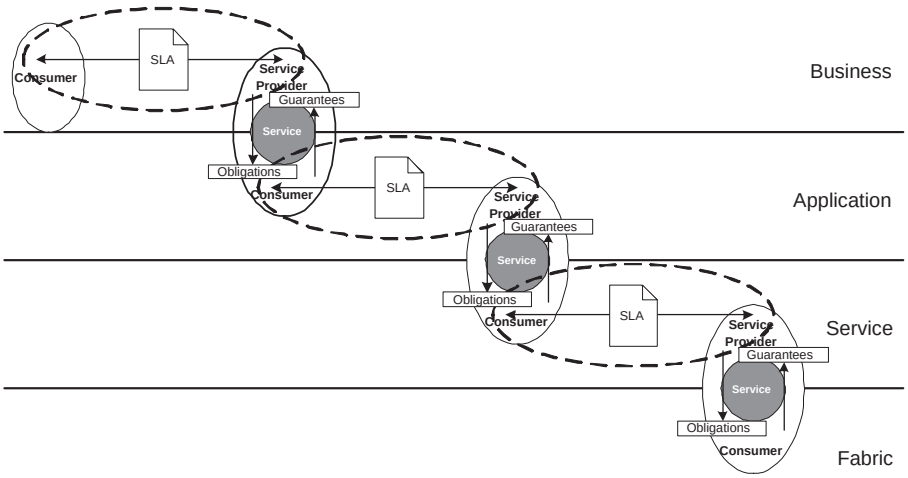

Figure 2. SLAs and different service levels.

Communication – *protocols* and *languages* through which NextGRID components communicate;

Behaviour – *interfaces* which dictate service behaviour are implemented (actually inherited) by all NextGRID components;

Management systems – those service management systems, e.g. for Naming and Addressing for service discovery, which are always available to Grid users and services; and

Schemas – schemas that underpin NextGRID concepts.

With respect to behavioural interfaces, it is a design requirement for NextGRID services to expose a minimal behavioural interface. The minimal Grid behaviour implemented by all NextGRID entities is largely driven by the degree of basic management functionality required by all services. Information discovery and service introspection provide the requirements for some of this basic management functionality. These behaviours are now being described in a document as a NextGRID Basic Profile.

3. NextGRID Architectural Decomposition

In order to help understand and build the architectural vision of the NextGRID project, some form of system decomposition is necessary. Frequently, systems can be decomposed into a *layered* architecture, where each layer communicates only with its adjacent layers. However, increasing complexity of Grid systems has resulted in the erosion of this simple approach, with some aspects of the system (e.g. security and messaging) spanning all layers of the architecture.

The NextGRID architecture is decomposed into four concepts, as follows:

> **Schemas:** Components of a system need a set of common schemas to communicate. The primary schema categories are: **Message schemas:** describing the contents of messages; **Naming and Addressing schemas:** providing data structures (based on WS-Addressing [1]) to address and access services; **Security schemas:** defining the format for policy and token contents and the basis for token and policy languages; **SLA schemas:** defining the negotiation and agreement languages for QoS agreements; **Service Description schemas:** defining the service discovery framework; **Activity schema:** providing the language to describe activities (e.g. programme executions and Web Service invocations); and **Query schemas:** providing the infrastructure for searching service and information registries.

These schemas are the glue that ties the various systems, which constitute the other three concepts of the NextGRID architecture, as follows:

> **Management Systems:** These components provide the minimal support for the NextGRID architecture to operate, but do not define any operational functions. They are approximately parallel to the basic schema categories discussed above. The bulk of the NextGRID architecture is concerned with these systems.
>
> **Functional Systems:** These components provide the conceptual framework for any functional activities that can be carried out. Their detailed definition is not part of the NextGRID architecture. They can be roughly categorised in terms of their relationship to data, and their functions exhibit some commonality in terms of cost-per-performance prediction. They are served by NextGRID Management Systems.
>
> **Orchestration Systems:** These components manage the dynamic composition of services, facilitated by orchestration systems ranging from simple service invocators, through to complex workflow processing engines.

Figure 3 depicts this decomposition and some of the interactions expected between the components.

3.1 Management Systems

3.1.1 Naming and Addressing. A naming service should be autonomous, scalable, distributed, secure, reliable, trusted, and have global scope. Desirably, the naming scheme (and a name resolution service) should also be fast, efficient, extensible and support internationalisation. The NextGRID Naming Service will be a combination of the Handle.net [2] system and a Web Services front end based on the WS-Naming [3] profile.

The operational capabilities of the naming service include: (1) creation of a contextual and unique name; (2) verification of a user selected name for

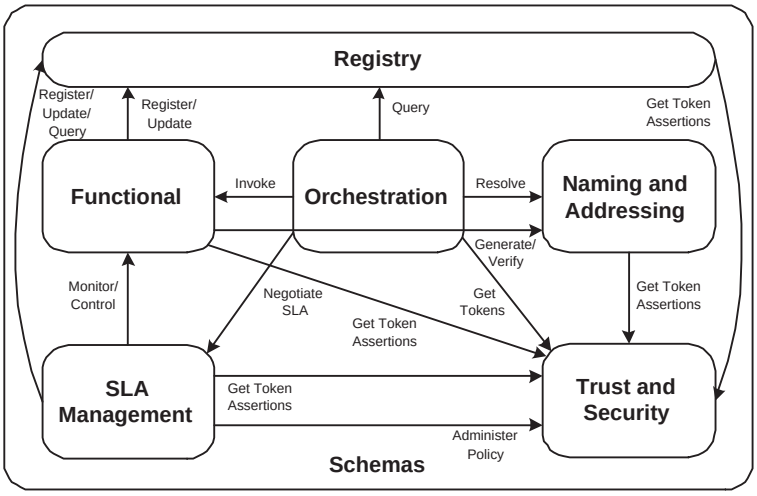

Figure 3. Overview of NextGRID Component Model and basic interactions.

uniqueness; and (3) access to the registry of addresses and aliases for a given name.

Use-cases for naming and addressing reveal several actors. Firstly, a name creator, who either requests or validates a name for an entity and then registers that name with some information (e.g. address or alias) pertaining to that name. The other primary actor is the address (or information) finder, who uses the name as input to query a registry for information about (e.g. address of) the named entity.

3.1.2 Security Facility. NextGRID provides dynamic authorisation and claims based security. The Security Tokens and Dynamic Authorisation services are simple services that are easy to create and operate, but their combination enable services to decide dynamically, on a request by request basis, whether a certain action or request is permitted.

There are two services that are central to the security facility. These are: **The Token Manager:** a Policy Decision Point that provides security access tokens based on policy information pertaining to the entities to be accessed and the claims made by a requestor; and **The Policy Manager:** provides interfaces for administrating the policy that governs access to a service.

The fundamental characteristic that makes these services unique to NextGRID is the emphasis placed on dynamic decision-making and policy management. NextGRID security and access policy can change dynamically throughout the lifecycle of a SLA based interaction between two entities.

3.1.3 SLA Management. The NextGRID SLA management system is autonomous. Once instantiated the system needs to include capabilities for negotiation of new SLAs, and for providing support for SLAs currently in effect. The latter includes monitoring running SLAs for QoS; accounting for SLA execution during and on completion; and enforcement of post-execution requirements, e.g. penalties and bonuses. These need to take place autonomously from the service provider's perspective or, if desired, by using trusted third parties. It is hoped that using trusted third parties for SLA management can be avoided in the NextGRID architecture through employing sufficient trust anchors.

3.1.4 Registry. In dynamic Grid environments, service endpoints cannot be hard-coded into applications. Rather, the location of available services, which meet the immediate needs of a consumer, must be found dynamically at application run-time. Service registries in NextGRID allow clients to search for required services among a set of available services. Multiple registries are used to support different environments and can exist in hierarchies for scalability.

3.2 Functional Systems

The NextGRID functional systems consists of a set of components that provide the conceptual framework for any functional activities that can be carried out using the NextGRID architecture. These functional systems can be described in terms of their relation to data.

3.2.1 Data Access. Access to data will be made available through data services. Data in all forms including, streams, sequences, files, images, traces, databases and archives provide input to analyses and models used by businesses, researchers, designers and decision makers. Abstractly, data access can be described as data where it is now.

3.2.2 Data Transfer. Like for data access, data transfer to and from data resources will be made available through data services. Additionally, the long-term goal will be to support a multitude of data transport protocols which provide both file transfer and remote movement of data as part of another operation. Such operations include database query or update processing, reading individual elements within a file, or consuming streams of data from live sources, e.g. scientific instruments, online market tickers, etc. Abstractly, data transfer can be thought of as the movement of data in space.

3.2.3 Data Processing. All computation in a NextGRID architecture can be thought of as falling within the conceptual model of data processing. This includes simple data transformations, such as compression or encryption, or more complicated scenarios, such as multi-part queries and in general the transfor-

mation of raw data into information or knowledge. A major aspect of this NextGRID functional system is the description of data through various types of metadata. Abstractly, data processing can be described as the movement of data in meaning.

3.2.4 Data Storage. As data and information are generated on a Grid, the issue of data storage must be addressed. Storage in the context of Grids has a wider remit than in conventional contexts. The need for replica management, distributed coherency, pre-processing to reduce transfer bandwidth requirements, and security and integrity constraints all add up to create a more complex problem. Abstractly, data storage can be described as the movement of data in time.

3.3 Orchestration Systems

Orchestration systems manage the dynamic composition of services in the NextGRID architecture. Dynamic composition of NextGRID services is facilitated by orchestration systems ranging from simple service invocators, through to complex workflow processing engines. The work on this aspect of the decomposition of NextGRID is just beginning to have an impact on the architecture, and few details are available at this stage.

Acknowledgments

The work presented in this paper is the result of the efforts of the NextGRID project consortium. The efforts of all consortium members involved in this work is duly acknowledged.

This work has been supported by the NextGRID project and has been funded by the European Commission's IST activity of the 6th Framework Programme under contract number 511563. This paper expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this paper.

References

[1] W3C, *Web Services Addressing (WS-Addressing)*, August 2004: http://www.w3.org/Submission/ws-addressing/

[2] The Handle System: http://www.handle.net/

[3] Andrew Grimshaw and David Snelling, OGSA-Naming Working Group, *WS-Naming Specification (Draft)*, Open Grid Forum, 4 December 2006.