Generic Service Model Specification Technical Report

Date:	23 June 2005
Author(s):	A. Tsalgatidou, G. Athanasopoulos, M. Pantazoglou, V. Floros, E. Koutrouli,

TABLE OF CONTENTS

1. INTRODUCTION

Our era has been marked by a shift in the way of thinking and acting across many domains such as business, science and community. The cornerstone in this new way of thinking is the notion of service. In general a service may be conceived as an action that is offered by somebody and may be used by anybody else. Although usually this action is performed upon a resource this is not mandatory. However, when it’s applied on a resource this resource could be a tangible object e.g. a hard disk or a ticket, or an intangible object such as a word document.

This shift has an effect on systems and the way they are developed. Service Oriented Development (SOD), which has emerged with the advent of services, is regarded as the future trend in distributed system development; its focus is on the use of services as system’s constituent parts. SOD emerged as an evolution to the component-based development and distributed object oriented computing and among its goals is to promote the loose coupling of the system’s parts in a far better way than component and object oriented technologies.

Nonetheless, the diversity of requirements stemming from the business, science and community domains has given rise to various service oriented technologies. Each of these technologies addresses specific needs of its related domain. Hence, web services support mainly business oriented systems, grid services support scientific oriented systems and p2p services support community oriented systems (e.g. Instant Messaging).

Despite the compliance of all service oriented technologies (e.g. web, grid and p2p services) with the same paradigm, they adhere to different models and they have different characteristics and different properties. Moreover, their heterogeneity spawns across other aspects such as architecture, supported protocols and standards, infrastructure, semantics and quality of service (QoS).

W3C has tried to mitigate the problem by establishing a service model [w3c 2004] according to which a set of roles and operations must be provided. The set of roles consists of the service provider, service requestor and service broker whereas the set of operations consists of publish, discover and invoke. However, all these concepts have either been extended[1] (e.g. grid services) or partly ignored[2] (e.g. p2p services) by the service technologies besides web services.

This diversity makes the integration of different services a strenuous task. In order to remove this burden from system developers, a generic service model incorporating appropriate features and properties of all service oriented technologies needs to be provided. This model will facilitate the specification of any type of service as well as the mapping and/or association of service features of one technology to the other.

[1] E.g. Grid services have annotated the w3c model with the introduction of the Resource element

[2] E.g. P2P services such as JXTA services don’t inherently support the notion of operation or service interface.

This report will try to specify a generic service model which will accommodate features and properties of all service technologies addressed by the SODIUM project. In doing so, we will investigate existing protocols and standards that are currently used. For each service type we will provide conceptual models that describe features and properties that have been incorporated by each protocol and standard. These conceptual models will be the basis for the identification of similarities and differences among the investigated services. The outcome of this process will be used as input for the specification of a generic service model, which will be consequently evaluated against services stemming from the pilot applications of the SODIUM project.

In order to proceed with the specification of a generic service model we need to define some high level requirements that will guide us through.

1.1 Requirements for generic service model

As we have stated before the provided generic service model will accommodate the the concepts of each type of service that will be tackled by the SODIUM project. Besides this, the generic service model should also satisfy some high level requirements which will embellish it with extra features and added value.

Therefore, the generic service model should have the following features:

- **Generic**: The specified model should be generic enough so that it can support the modelling of all types of services. The range of supported types of services consists of web, grid and p2p services only. However, it may span across other types, such as Jini services which are not currently addressed by the SODIUM project
- **Abstract**: The model should incorporate all common concepts of the addressed service types. These concepts should be described in an abstract way that will enable their mapping to the concepts used by each type of service.
- **Extensible**: The model should be easily extended with features and properties that are not currently addressed by the existing service technologies. This should be performed through appropriate extension mechanisms
- **Modular**: Related information and properties should be grouped in independent modules, thus allowing the easier modification and extension of specific information. Appropriate extensions may be provided through the use of specialization and/or generalization mechanisms
- **Expressive**: The model should accommodate features and properties that support all service operations such as, discovery and composition. This feature will be endowed to the languages that will be based upon it
- **Simple**: The generic service model should be simple enough so that it will provide for all types of users and tools that will use it

These requirements will act as a placeholder and will drive us through the specification of the generic service model. They will merely serve as guidelines and not as rigid requirements that must be completely satisfied. Moreover, since some of them contradict e.g. simplicity and expressiveness, appropriate tradeoffs will have to be made.

2. STATE OF THE ART IN SERVICE ORIENTED TECHNOLOGY

2.1 Introduction

The definition of a service and its fundamental components constitutes a hotly debated issue. Each of the addressed technologies tries to inflict its own perspective on the subject. Nonetheless, despite the existing differences, they share some common features and properties.

A thorough investigation of contemporary service oriented technologies that are addressed by the SODIUM project, i.e. web services [w3c 2004], grid services [OGSI][TCFFGK 2002] and p2p services [JXTA][Edutella], reveals that as services we may regard self-described software systems, which interact with their clients over the network through messages (see Figure 1). The description of a service facilitates its clients in identifying the messages that can be exchanged and the interactions that may be conducted with a service.

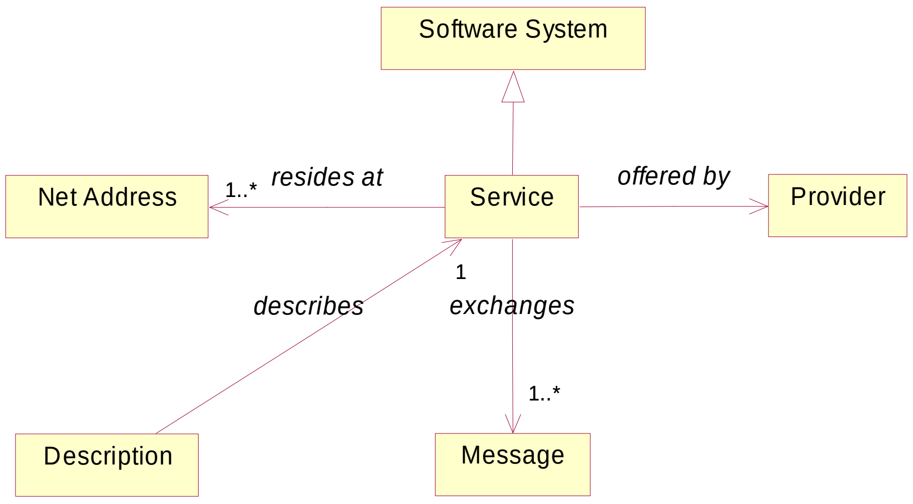

Figure 1 : Abstract service mode

Furthermore, since a service is a software system, it's endowed with some extra properties such as those depicted in Figure 2.

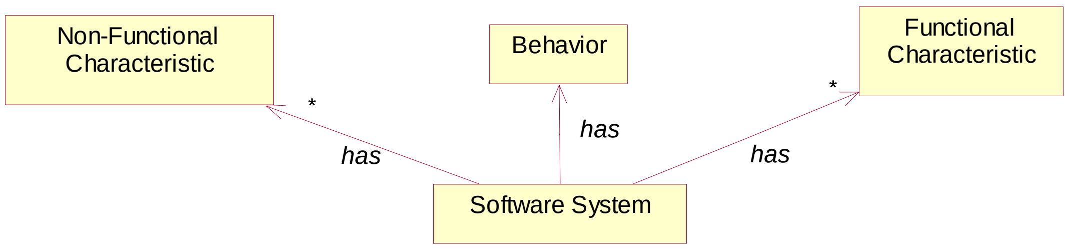

Figure 2: Service properties

Therefore, according to Figure 2 a service has functional and non-functional characteristics as well as a behaviour, which is described in terms of its functional characteristics (e.g. supported operations).

This abstract model, which is presented in Figure 1, has been extended with special properties from each of the addressed service technologies (see Figure 3). As it will be illustrated in the following the properties that have been incorporated by web, grid and p2p services have created a gap among these service types. Though, some of them have tried to bridge this gap e.g. web and grid, there are still differences among them.

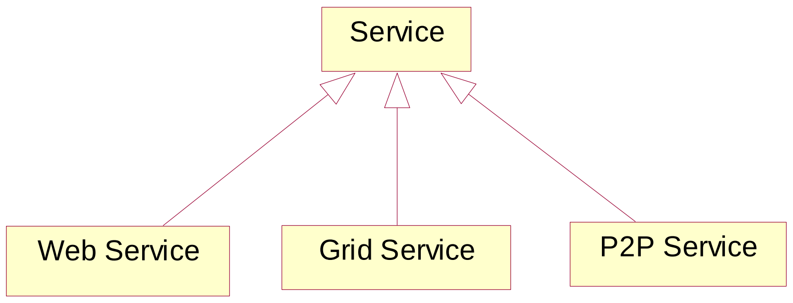

Figure 3: Service extensions

Another important issue that has cropped up lately with the advent of the Semantic Web initiative [BerHenLa01] is the annotation of service descriptions with semantic information that will enable the automated execution of operations such as discovery, invocation and composition. For each of the addressed service types there have been research efforts and proposals which investigate how their semantics can be accommodated. Yet most of these approaches are based on a small set of protocols and frameworks that have been originally applied in web services [OWL 2004][OWL-S/DAML-S][RDF 2004].

Apart from semantics another important issue that has raised considerable momentum recently is quality of service (QoS). There are many research efforts and protocols that try to address the specification, monitoring, measurement and management of QoS. Yet as in the case of semantics the most promising proposals are those that have been originally applied in web services [WS-QoS][WSLA][WSOL].

Therefore, in order to construct the generic service model we first have to look into each type of service and identify its properties and characteristics. In the following sections we will dig into the frameworks, protocols and standards that have been leveraged by each of the addressed types of services.

2.2 Web Services

There are many definitions for what constitutes a web service. The UDDI consortium [UDDI] defines web services as "*self contained, modular business applications that have open, Internet-oriented, standards-based interfaces*", whereas W3C states that web services are "*applications identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artefacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols*" [W3C 2004]. Among the goals of web services, as it has been stated in [AlCaKuMa 2003], is to enable interoperability among software systems and to leverage enterprise-application integration (EAI).

A common misconception that many people have is that web services are web interfaces to components. However, web services aren't just that, one of their differences is that web services expose higher level business logic whereas components tend to expose lower level business objects and processes. An extended list of differences can be found in [Szyp 2003], [Vogels 2003], [CKMTW 2003] and in [GKS 2002].

In contrast to these misconceptions, web services are an exact instantiation of the service oriented model that is specified by W3C [w3c 2004]. They adhere to the set of roles and operations that have been identified by the W3C's model and they have also managed to establish a standardized protocol stack. SOAP [SOAP], WSDL [WSDL 2001] and UDDI [UDDI v2 DS] are the most well known standards used for the execution of a basic set of operations i.e. invocation, description and discovery.

In practise, though, web services have been regarded as services that abide by WSDL [WSDL 2001], SOAP [SOAP] and UDDI [UDDI] standards. The protocol stack and model that were introduced by W3C and other major vendors (IBM, Microsoft, SUN, SAP, etc) have been widely accepted. As it is presented in Figure 4, the layered structure that was used for the web service protocol stack, tackles high level aspects such as service composition, transactions, etc. based on the use of lower level protocols.

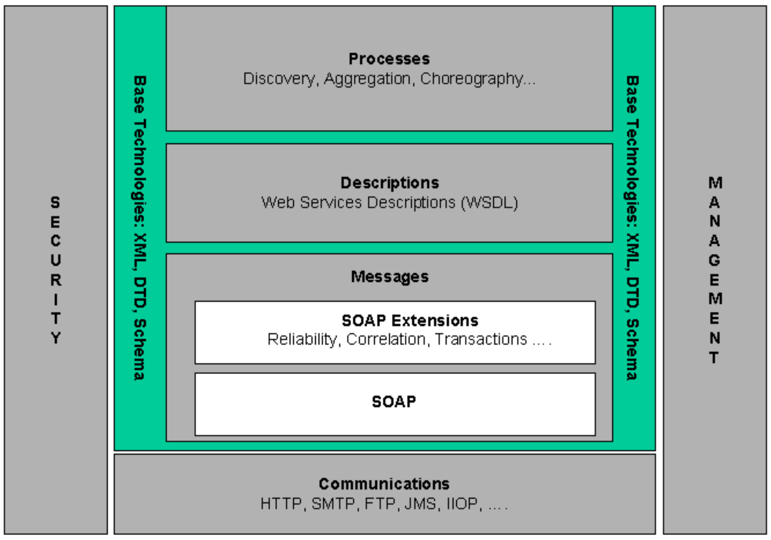
SECURITY
Base Technologies: XML, DTD, Schema
Processes
Discovery, Aggregation, Choreography...
Descriptions
Web Services Descriptions (WSDL)
Messages
SOAP Extensions
Reliability, Correlation, Transactions ....
SOAP
Base Technologies: XML, DTD, Schema
MANAGEMENT
Communications
HTTP, SMTP, FTP, JMS, IIOP, ...

Figure 4: W3C Web Service Architecture Stack

In the following we describe the set of standards and protocols that are used for supporting the basic layers of the W3C's web service architecture stack. These protocols provide mainly syntactic constructs and contribute in the web service's syntactic information model. An exhaustive look into existing standards, protocols, frameworks and research initiatives related to the Service Oriented Computing is provided in [BHATBVKL04].

2.2.1 Basic Web Service Model

A web service's syntactic information model comprises properties and features that address basic aspects such as invocation, description and discovery as well as a set of extended features that address composition, transactions, security, reliability, policies, etc. An ample of protocols has been proposed for addressing these features. However, currently only WSDL [WSDL 2001], SOAP [SOAP] and UDDI [UDDI] have been standardized and widely accepted.

Nonetheless, the plethora of protocols that have been proposed and the number of properties and features that each one of them introduces has increased the complexity of the web service model and has brought a lot of confusion and misconception to the web service community. This has also been the argument of Wernel Vogels who in [Vogels 2003] has established a minimal model.

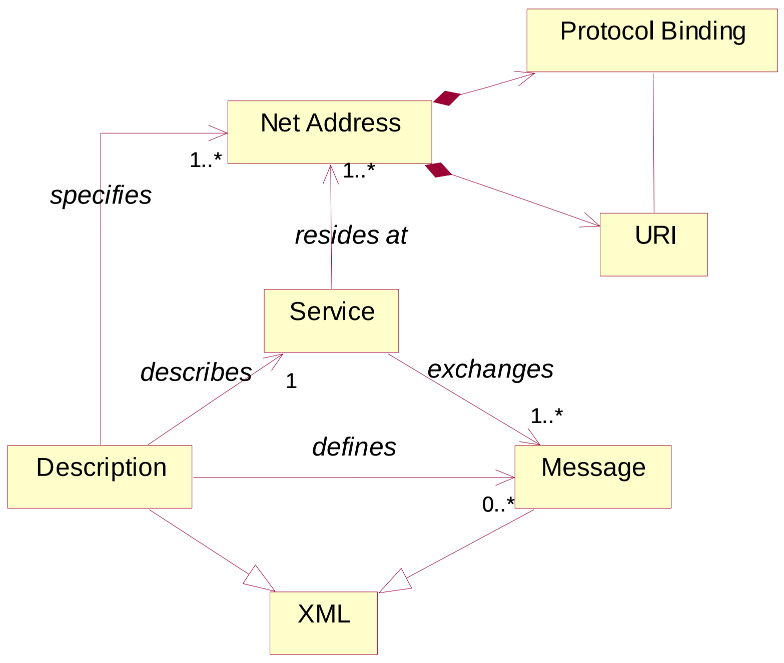

Figure 5: Minimal Service Model

According to this model (see Figure 5) a web service is a software system, which resides at a specific network address exchanges messages with its consumers and is described by a description document. The cornerstone of this model is that both the service description and the exchanged messages are XML documents. The service description document specifies the format of the messages that will be exchanged and the network address that will be used for the service invocation, which is composed of a protocol binding and a URI part.

This model though, doesn't suffice for the provision of tasks such as discovery, and composition. In the following sections we present a range of protocols that support the description, discovery and invocation of a web service. The set of standards that are addressed within the following sections comprises the Web Service Description Language (WSDL), the Simple Object Access Protocol (SOAP) and the Universal Description and Discovery Interface (UDDI).

2.2.1.1 Web Service Description Language (WSDL)

The Web Service Description Language (WSDL) is an XML-based language used for describing functional properties of Web services. It aims at providing self-describing XML-based definitions that applications, as well as people, can easily understand. Thus, as it is illustrated in Figure 6 a WSDL Document is a realization of the Description element presented in Figure 5.

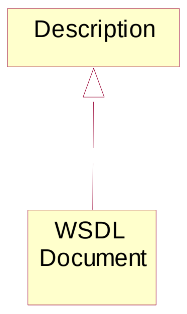

Figure 6: Web Service Description structure

The WSDL standard is currently under revision currently and a new version WSDL 2.0 [CGMSW] has been proposed as an update to the existing WSDL 1.1 [WSDL 2001] specification. Yet, since the WSDL 2.0 specification hasn't been finalized and no commercial tools are supporting it, we will base our investigation on the existing WSDL 1.1 specification.

A WSDL document consists of two parts, namely abstract and concrete. This twofold structure provides for the reuse of service descriptions. These two parts, as it is presented in Figure 7, relate to each other, since elements of the concrete part have references to elements of the abstract part.

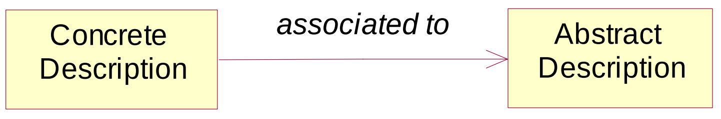

Figure 7: WSDL structure

According to WSDL1.1 (Figure 8), a service consists of a collection of message exchange endpoints (or ports). A port provides a description of an implementation binding and refers to an abstract description of a service port type (service interface). The abstract description of a service contains: (i) definitions of the messages which are exchanged by the service (i.e., input and output messages) and (ii) signatures of service operations.

The implementation binding provides the means to map abstract operations into concrete service implementations. It essentially contains information about the location (URI) of a protocol binding, the transport protocol that will be used for the message exchange (e.g., SOAP over HTTP) and mappings between the abstract description of messages and the underlying communication protocol message types (i.e., how interactions with service occur over SOAP). Figure 8 presents the conceptual model of the elements that are introduced by the WSDL specification.

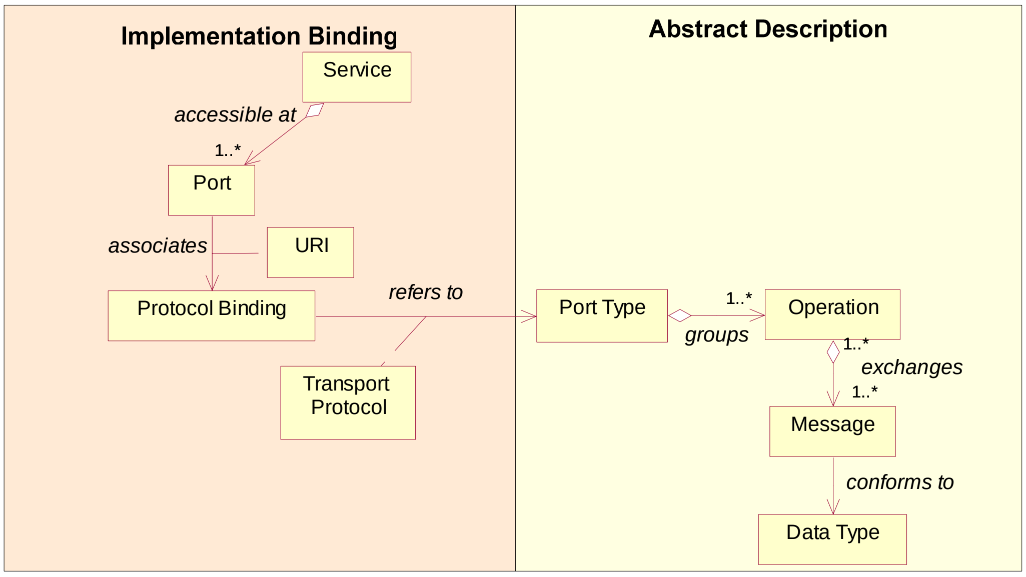

Figure 8: WSDL conceptual model

2.2.1.2 Simple Object Access Protocol (SOAP)

SOAP provides an XML-based protocol for structured message exchanges. It relies on existing transport protocols such as HTTP and SMTP and it features document-based communication among Web services. Document-based communication allows the integration of loosely coupled services.

The SOAP protocol version which is presented in this section is SOAP v1.1 [SOAP 1.1]. Yet, the latest version of the SOAP is version 1.2 [SOAP] which still hasn't been finalized and accepted.

SOAP provides for the formatting of the messages that are exchanged among a web service and its clients. Thus, a soap message is the realization of the Message element that is presented in Figure 5. Figure 9 illustrates the relation among the Message element presented in Figure 5 and the SOAP Message element.

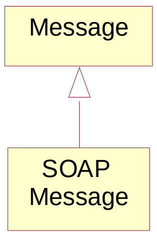

Figure 9: SOAP Message

According to SOAP a message consists of an Envelop element which may be further decomposed in a Header part and a Body part. The header part is optional whereas the body part is mandatory. Both the header and body parts may be composed of multiple blocks which convey information related to the underlying infrastructure or the service respectively.

The elements contained by the header and body parts are named header and body entries respectively. Instead of a body entry a message's body part may contain a single fault entry, which withholds information related to errors that may occur during the message exchange or the message processing. The structure of a SOAP message is illustrated in Figure 10.

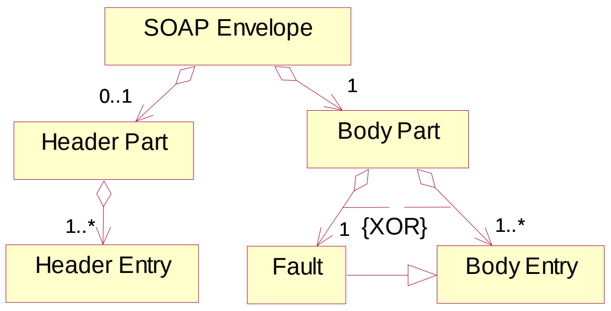

Figure 10: SOAP Message Structure

2.2.1.3 Universal Description Discovery and Integration (UDDI)

UDDI [UDDI] is a specification of an XML-based registry for Web services. It defines an interface for advertising and discovering Web services and it provides three types of information: white pages, yellow pages, and green pages.

The UDDI specification is also under update and a new version has been proposed and accepted by the OASIS standardization body. Nevertheless, UDDI v3.0, which is the latest version, hasn't outnumbered the existing UDDI v2.0 installations. Thus, our investigation will be based upon the UDDI v2.0 [UDDI v2 DS].

The UDDI-supported service publication and discovery mechanism is mainly based on the use of SOAP. The structure of the UDDI protocol and its constructs is illustrated in Figure 11.

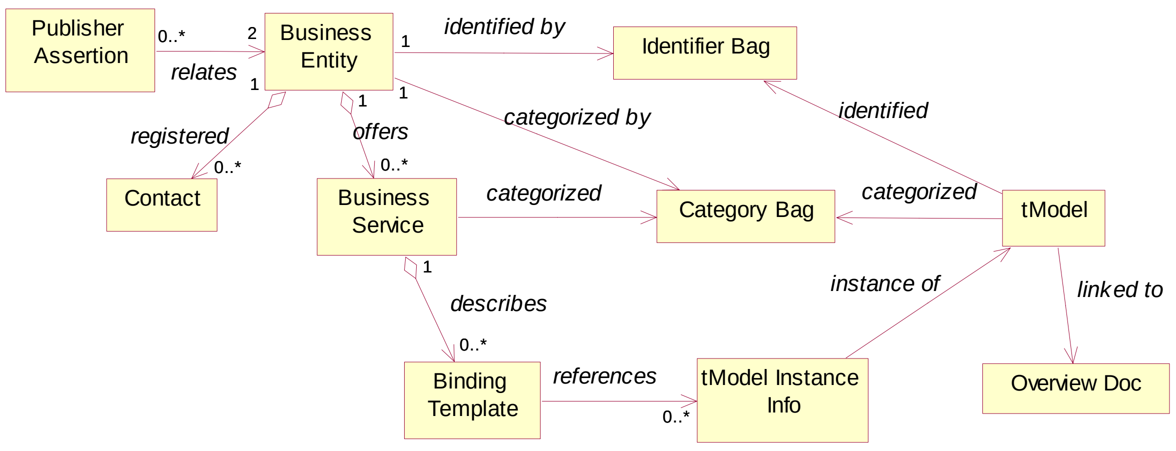

Figure 11: UDDI structure

According to [UDDI v2 DS] a BusinessEntity construct represents the organization that provides a service. Such an organization may be associated to other organizations via the

PublisherAssertion element. A BusinessEntity construct may convey extra information such as identification or classification info, which is associated to a BusinessEntity element via an IdentifierBag or CategoryBag element.

A BusinessService, which represents the service that is offered by a business entity may be classified according to a specific taxonomy with the use of a CategoryBag element. Extra information about a BusinessService element may be provided through the BindingTemplate elements which convey a service's technical details. Technical information about a service is described via tModelInstanceInfo elements.

A tModelInstanceInfo element is an instance of a tModel element, which provides descriptions with information about a service's behaviour, the conventions it supports as well as the protocols and standards it conforms to. However, a tModel is a general container of any type of information, which can be related to identifiers and classified according to specific taxonomies. In addition, a tModel element may be associated to external documents through the OverviewDoc element.

2.3 Grid Services

The term Grid refers to "a system that is concerned with the integration, virtualization, and management of services and resources in a distributed, heterogeneous environment that supports collections of users and resources (virtual organizations) across traditional administrative and organizational domains (real organizations)" [FKT 2001]. In practice a Grid is a software infrastructure that handles the details of resource sharing among distributed environments that reside under different administrative domains. Traditionally Grid resources are accessed using well defined, standard protocols and are governed either by local or distributed policies. Practice has proven that programming in a Grid environment is a tedious and ad-hoc process. No common agreed programming model exists and in any case the diversity of applications that can be developed makes difficult the construction of such a general-purpose abstract model.

During the recent years there has been a shift from the API/Protocol oriented Grid programming model to a Service-oriented approach. In "The Physiology of the Grid:An Open Services Architecture for Distributed Systems Integration", co-authored in 2002 by the two Grid computing pioneers Ian Foster and Carl Kesselman, a Service Oriented Architecture was prescribed where shared resources are described and accessed using open Service-based interfaces [OGSA]. This architecture known as OGSA (Open Grid Services Architecture) has been initially materialized by the OGSI (Open Grid Services Infrastructure). OGSI adopted the Web Services technology and extended it in the areas were it was considered inadequate for developing Grid applications, namely stateful and transient interactions, life-cycle management and notifications [OGSI][FKNT 2002].

However, the criticisms that the OGSI framework received from the web service community led to its refactoring and the introduction of the WSRF [CFFFGMST 2004] specification. WSRF realizes the notion of grid service that has been specified in OGSA through the segregation of the service and resource concepts.

2.3.1 Web Services Resource Framework

WSRF provides an alternative approach for accessing and manipulating disperse distributed computing resources using a Service Oriented Architecture. WSRF was introduced after the critics that OGSI received from the web services community. In short the introduction of OGSI raised controversy and anxiety among the members of the web service community. OGSI proposed a tightly-couple, overloaded and object-oriented extension of web services that conflicts with the rest of the web services specifications and tools. The critics that OGSI received led to the so called refactoring of the standards and the introduction of the Web Services Resource Framework (WSRF) [CFFFGMST 2004].

WSRF defines a set of five web services specifications that together with the WS-Notification and the WS-Addressing specifications provide similar functionality to that of OGSI, which is compatible with the existing WS tools and in accordance with the common WS specification definition philosophy. These specifications are: *WS-ResourceLifetime* [WS-ResourceLifetime], *WS-ResourceProperties* [WS-ResourceProperties], *WS-RenewableReferences* , *WS-ServiceGroup* [WS-ServiceGroup] and *WS-BaseFaults* [WS-BaseFaults]. Except from WS-RenewableReferences all other specifications have reached a stable state and have been released to public.

In the context of WSRF a grid service has been defined informally as “a Web service that is designed to operate in a Grid environment, and meets the requirements of the Grid(s) in which it participates”. Grid resources are exposed and controlled by web services using the so called implied-resource pattern. WSRF specifications facilitate i) the implicit association between web services and grid resources, ii) the definition of the resource life-cycle, iii) the specification of static information, iv)the grouping of multiple resources and v) notifications on resource state changes.

Although WSRF is inline with the rest of the web service specifications the notion of WS-Resource has raised once again controversy in the web services community which remains skeptical regarding this proposal. According to web service purists it is questionable why a service has to use an explicit construct to represent state even if this relationship is enforced implicitly. Indeed WS-Resource alters the semantic perspective of web services; a resource is an artifact based on a web service that can be queried, composed and executed. It extends the operation centric view that current web service technologies support with a resource centric approach. It remains to see the value and applicability of this philosophy in real world applications.

2.3.1.1 WS-Resource and the Implied Resource Pattern

Central to WSRF is the notion of WS-Resource. WSRF defines a WS-Resource to be the implicit coupling between a web service and a stateful computing resource that the web service manages (Figure 12). This coupling is implicit in the sense that the web service consumer does not have direct access neither to the resource state nor to the methods that manage this state. A resource is instantiated, manipulated, managed and finally destroyed as a result of a web service operation invoked by a client, or to be more precise by the exchange of XML messages between the client and the service. It is on the service programmer to decide up to what extend the relationship between the web service and the associated resources is exposed to the service consumer.

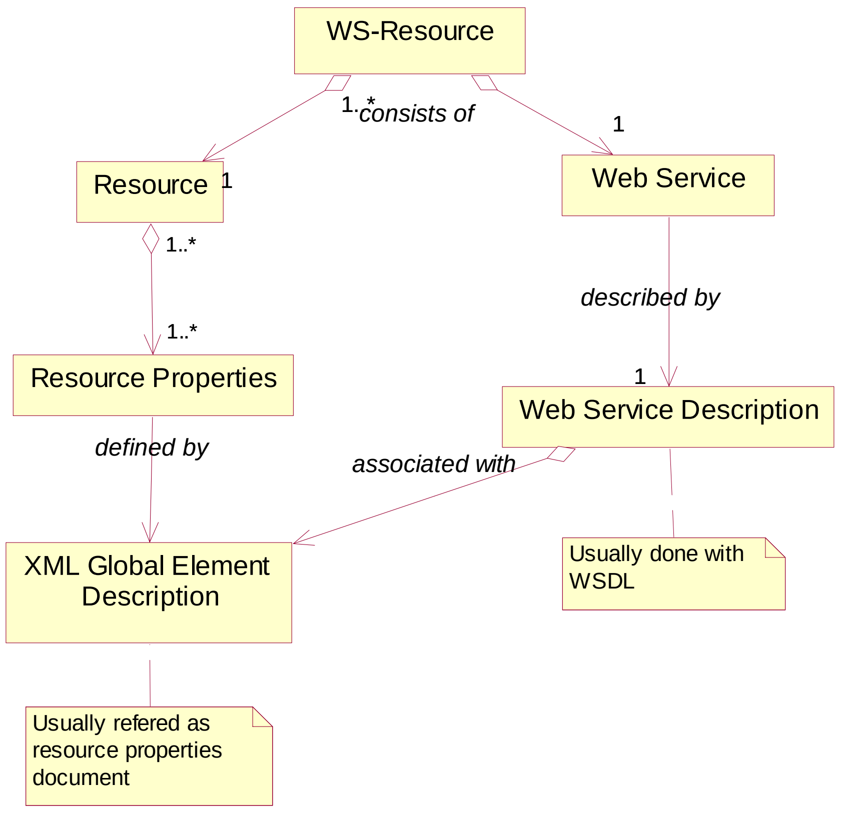

Figure 12 : WS-Resource and the implied resource pattern

2.3.1.2 Resource Properties

The state of a WS-Resource is defined in terms of a resource properties document, which is an XML document that is associated with the web service description (the WSDL document). The resource properties document defines the data types and the values of a resource's properties that can be viewed and modified by a client through the provided service interface [WS-ResourceProperties].

According to the specification the state of a WS-Resource is exposed through a set of properties. The definition of these properties is included in an XML resource properties document defined using XML schema. Each resource property is represented as an XML element within the WS-Resource properties document. Service clients can retrieve the type of a WS-resource as part of the portType definition of the web service. Moreover they can manipulate the value of these properties (thus modify the resource state) by message exchanges that read, modify and query this XML document.

2.3.1.3 Addressing

For the addressing of a WS-Resource, WSRF takes advantage of the WS-Addressing [WS-Addressing] specification endpoint reference construct. An endpoint reference is used to represent the address of a web service deployed at a given network endpoint. According to WS-Addressing an endpoint reference may also contain metadata associated with the Web service. WSRF takes advantage of this capability. The implied resource pattern defines a conventional use of WS-Addressing in which a stateful resource is treated as an implied input for the processing of a message send to a Web service. The endpoint reference construct is used to identify the stateful resource to be used in the execution of all message exchanges performed by this web service.

This type of endpoint reference is referred to as a *WS-Resource-qualified endpoint reference*. WSRF uses a WS-Resource-qualified endpoint reference to represent a “network-wide pointer” to a WS-Resource. A WS-Resource-qualified endpoint reference may be returned as a result of operations such as a Web service message request to a factory service which instantiates and returns a reference to a new WS-Resource, from the evaluation of a search query on a service registry or as a result of some application-specific Web service request.

When an endpoint reference becomes invalid for some reason (for instance the resource it represents is unavailable), service providers have the option to use alternate resources to continue service provision. In this case endpoint references should be renewed appropriately at the client-side. The WS-RenewableReference [CFFFGMST 2004] specification defines WS-Policy assertions for the purpose of decorating endpoint references with information necessary to retrieve a new endpoint reference in the event the reference becomes invalid.

2.3.1.4 Lifecycle

Issues of resource lifecycle are addressed by the WS-ResourceLifetime specification [WS-ResourceLifetime]. More specifically it defines details of a WS-Resource instantiation and destruction.

Contrary to OGSI, WSRF does not enforce a Factory pattern approach for the instantiation of new resources (see Figure 13). A WS-Resource may be created by some out-of-band mechanism, or through the use of a WS-Resource factory. A WS-Resource factory is any Web service capable of instantiating WS-Resources. The result of a WS-Resource factory operation is typically an endpoint reference of the newly created WS-Resource. This endpoint reference is usually conveyed to the service consumer or can be stored in WS-Resource registries were it can be queried and retrieved from other clients.

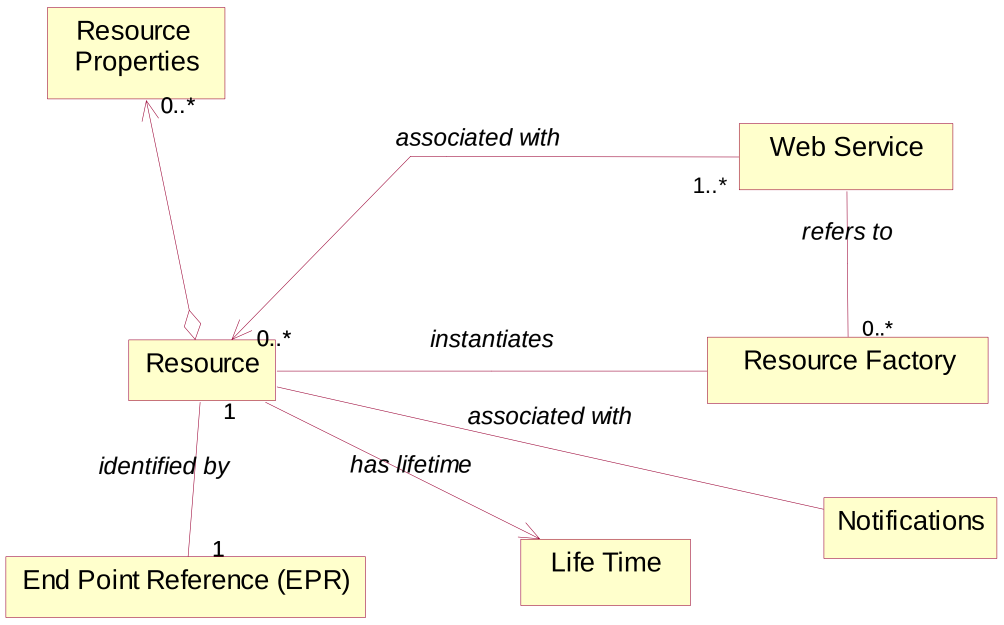

Figure 13 – Conceptual view of resource instantiation

WS-ResourceLifetime defines two lifetime management approaches: immediate and scheduled destruction.

In the immediate destruction approach the service requestor explicitly requests the termination of a resource instance by sending an appropriate request to the web service, together with the WS-Resource qualified endpoint reference. The web service that participates in the WS-Resource takes the endpoint reference and identifies the specific resource to be destroyed. Upon the destruction of the resource the web service sends a reply to the requestor with a message that acknowledges the completion of the request. Any further message exchanges with this WS-Resource will return a fault message.

Besides the ability to destroy a WS-Resource immediately, WS-ResourceLifetime defines the means by which a WS-Resource may be scheduled for termination at a future time. Using a WS-Resource-qualified endpoint reference, a service requestor may first establish and subsequently renew the scheduled termination time of the WS-Resource. When that time expires, the WS-Resource may be self-destroyed without the need for a synchronous destroy request from a service requestor. A requestor may periodically update the scheduled termination time to adjust the lifetime of the WS-Resource.

2.3.1.5 ServiceGroups

The WS-ServiceGroup [WS-ServiceGroup] specification defines the mechanisms used for representing and managing heterogeneous by-reference collections of Web services. This specification can be used to organize collections of WS-Resources, for example to build registries, or to build services that can perform collective operations on a set of WS-Resources.

2.3.1.6 Composition

Since in principle WSRF is based on common web service standards WS-Resources can be composed based on the operations they expose. In addition to operation composition, the service programmer may also aggregate the WS-Resource properties defined in the WS-Resource properties documents of the various constituent portTypes to yield the final, complete WS-Resource property document declared with the final composed portType.

2.3.1.7 Notification

WSRF exploits the family of WS-Notification specifications to define mechanisms and interfaces that allow clients to subscribe to topics of interest, such as resource property value changes for a WS-Resource. More specifically the WS-BaseNotification specification, describes the basic roles, concepts, and patterns required to allow a subscriber to register interest in receiving notification messages from a notification producer. A notification may concern anything, a change in the value of a resource property, some other internal modification in the state of the notification producer, or some other “situation” within the environment. A subscriber registers interest in receiving notification messages on one or more topics by issuing a “subscribe” message. In response, the subscriber receives a WS-Resource-qualified endpoint reference to a “subscription” WS-Resource. The subscription WS-Resource models this relationship between the subscriber and the producer, and it uses WS-ResourceProperties and WS-ResourceLifetime to help manage this relationship.

2.3.1.8 Base Faults

Part of WSRF is also the WS-BaseFaults specification. WS-BaseFaults[WS-BaseFaults] defines a base fault type which is used to return faults in a Web services message exchange. The specification is not constrained to WS-Resource faults and is generic enough so as to be used in any web service interaction. Hence WS-BaseFaults is used by all other WSRF specifications as a consistent mechanism for returning faults.

2.4 P2P Services

The term “peer-to-peer” refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner. These resources encompass computing power, data (storage and content), network bandwidth, and presence (computers, human, and other resources). The critical function can be distributed computing, data/content sharing, communication and collaboration, or platform services. Decentralization may apply to algorithms, data, and meta-data, or to all of them. This does not preclude retaining centralization in some parts of the systems and applications if it meets their requirements. Typical P2P systems reside on the edge of the Internet or in ad-hoc networks.

The term “p2p service” though, hasn’t kindled a consensus currently. On the contrary there are a lot of misconceptions and hype around this term. In order to provide a definition of a p2p service we will briefly describe two classification schemes that will guide us through. The criterion for the first scheme is the level of granularity that each service provides. Thus under this scheme we have the following categories:

- *Elementary Services*: services supporting the P2P network formulation and other basic functionalities (e.g. discovery of nodes, data or other resources, message exchange, network structure, etc.)
- *Coarse Services*: services exposing coarse business logic, such as file sharing and instant messaging.

For the second classification scheme the criterion is based on the service provider. Therefore we have the following categories:

- *Node Services* offered by each node of the P2P network. Such services are mostly elementary services
- *P2P Network Services* offered by the whole P2P network, e.g. shared data space
- *Intermediate Services* offered by groups of P2P nodes to other nodes or to external clients with respect to the P2P network.

Based on the first classification scheme we may define a coarse p2p service as "*the provision of resources or the execution of tasks of one or more (temporarily provider) peers on behalf of one or more (temporarily user) peers in a P2P network. A human user or, alternatively, a program (e.g. an electronic agent acting as a user), is associated with one peer node in the network, and is assumed to have an objective that can be fulfilled by the offered services. On the corresponding peer, it invokes functionality to discover an appropriate service and to determine and contact one or more provider peers offering that particular service*" [GHMS 2003]. Accordingly an elementary p2p service may be defined as "*services that support basic functionality in a P2P system, such as discovery of peers or content, peer membership management, query formulation and routing, etc*".

Throughout this report we will accommodate both of these definitions. Thus, we are regarding p2p services both as coarse services and as elementary services with respect to the introduced classification scheme.Figure 14 presents the use model of p2p services, which adheres to the first definition of a p2p service [GHMS 2003]:

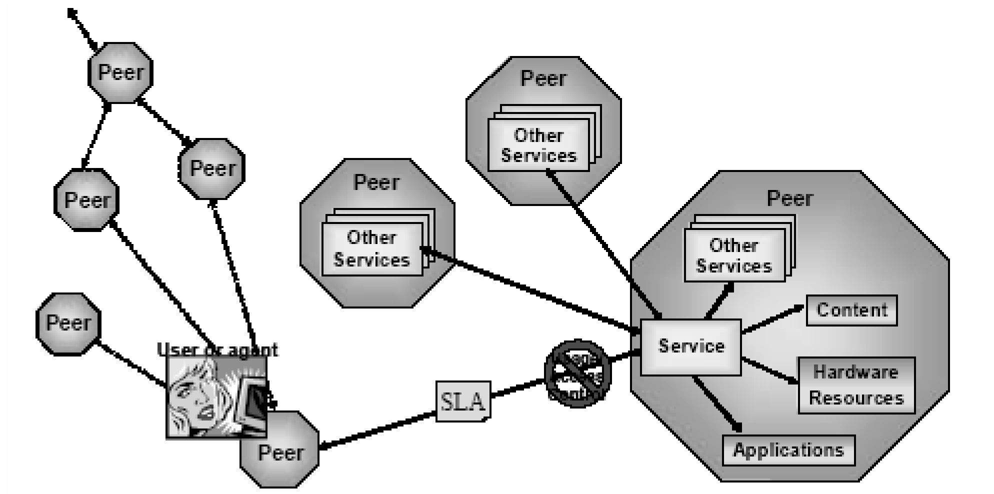

Figure 14: Use Model of P2P Services

The diversity among the types of services i.e. coarse and elementary p2p services, along with the proliferation of p2p systems has made it difficult to establish a widely accepted set of standards and protocols that would facilitate the description of p2p services. Each one of the contemporary p2p networks accommodates its own proprietary set of protocols, which are arbitrarily designed so that they enable their specific features and properties.

Since there is no common infrastructure, protocols and standards used for the specification of p2p services, we will have to look at each of the existing proprietary p2p platforms and systems. For each of the addressed p2p system/platform we will contrive a conceptual model describing its constructs and their interrelationships.

However, the list of available p2p systems and platforms is enormous. Hence, we need to specify a set of criteria that are going to help us in identifying the systems and platforms that are going to be tackled by the SODIUM project. The list of selection criteria that we have identified consists of:

- *Acceptance*: The selected platforms and systems should be widely accepted and used by developers, p2p community, end users, etc. This will help us in having a gross effect on the existing technology and systems.
- *Service notion support*: It is desirable to select platforms and systems that accommodate the notion of service. Such systems will enable the specification of the “p2p service” term and the identification of properties and features that such services should have.
- *Openness*: The selected platforms and systems should follow an open source license approach and be based on open and extendable protocols and standards. This will facilitate the identification of their architecture and the provision of possible extensions that might be needed for accommodating the p2p service concept.
- *Application domain*: The selected platforms and systems should be able to support the development of applications for as many as possible application domains. This will enhance the effect of the systems as far as market penetration is concerned.

Based on the aforementioned criteria list the set of systems and/or platforms that are going to be addressed by this report are:

- JXTA [JXTA]
- Gnutella [Gnutella]
- Edutella [Edutella]

In the following we illustrate the outcomes of our investigation on the selected p2p systems. More specifically section 2.4.1 presents the concepts and properties provided by the JXTA platform with emphasis on those that are supporting the JXTA service notion. Section 2.4.2 presents the Gnutella protocol along with its conceptual model and finally section 2.4.3 presents Edutella and its supported concepts.

2.4.1 JXTA

JXTA [JXTA] is a set of open, generalized peer-to-peer protocols that allow network connected devices to communicate and collaborate as peers. JXTA protocols are independent of any programming language and up to now there have been several implementations for different environments. In general we may regard JXTA as a framework that can be used for the development of p2p networks.

The provided JXTA protocols standardize the services that are used for:

- Discovering peers
- Organizing peers into peer groups
- Advertising and discovering network services
- Handling peer communication
- Handling peer monitoring

A JXTA p2p network is composed of a set of interconnected peers which can be organized in peer groups. Peer groups provide a common set of services such as document sharing, instant messaging, etc. Peers and Peer Groups advertise themselves along with their provided services so that other peers may know how to connect and interact with their respective services. Peers can communicate with each other through pipes, which bind to specific endpoints (e.g. specific IP address and port) of the interacting peers and provide an asynchronous, one-way communication channel that is used for exchanging messages. Messages are simple XML documents whose envelop contains information such as routing, message digest and credential information and their body consists of application related data.

The essential aspects of the JXTA architecture that differentiate it from other network models are:

- The use of XML documents for the description of the provided resources and the exchanged messages
- Independence of pipes and peers and of peers and endpoints that doesn’t rely on the use of a central naming mechanism such as DNS
- A unified naming scheme and mechanism for addressing resources

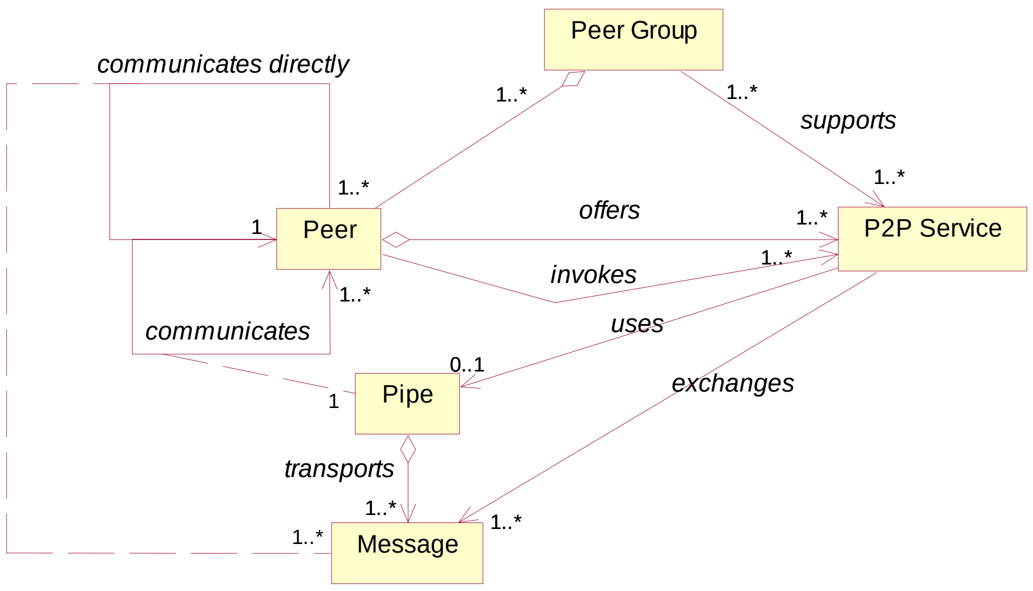

Figure 15: JXTA Main concepts

One of the major advantages of the JXTA platform is the establishment of the service notion. A Service (or network service) according to [JXTA] is a realization of a behaviour that may be provided by a peer or a set of peers, published to the p2p network, discovered and invoked by other peers. The JXTA platform and set of protocols support all these operations.

According to the JXTA specifications, two types of services may exist within a p2p network, namely the Peer Services and the Peer Group Services. A peer service is accessible only at the peer that is publishing it and if that peer fails, the service also fails. Multiple instances of the service may run on different peers, but each instance publishes its own advertisement.

A peer group service is a collection of service instances (potentially cooperating with each other) that are running on multiple peers of the peer group. If any peer fails, the collective peer group service is not affected (assuming the service is still available from another peer of that group). Peer group services are published as a part of the peer group advertisement.

In JXTA services are regarded as Modules that can be instantiated on a peer (see Figure 15). Peers are able to discover available services in a p2p network through their respective ModuleAdvertisements, which advertise behaviors that are provided as services. A ModuleSpecAdvertisement is a service description which provides textual information related to the service and information on how a service can be invoked (e.g. through a reference to a PipeAdvertisement). A PipeAdvertisement, provides information that can be used for establishing a pipe (communication channel) among communicating peers.

In addition, peers are able to download a service’s code and provide instances of a service through a ModuleImplementationAdvertisement. The ModuleImplementationAdvertisement provides references to a ModuleSpecAdvertisement which describes a service, and the location where a service implementation may be retrieved along with parameters describing the necessary properties for the service instantiation. The aforementioned concepts are illustrated in Figure 16.

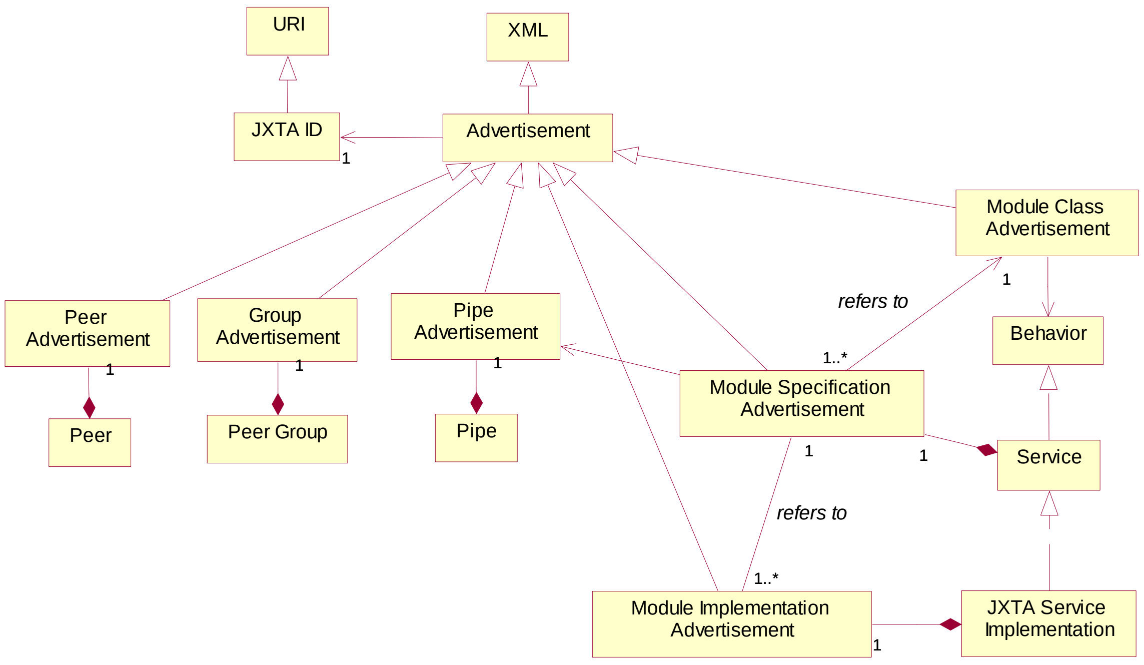

Figure 16: JXTA Advertisements structure

All the basic elements of the JXTA platform apart from messages have a unique JXTA ID, which facilitates their naming and identification. Through these IDs the JXTA platform and set of protocols remain independent of the underlying network and naming schemes and are able to pinpoint specific network addresses (see Figure 17).

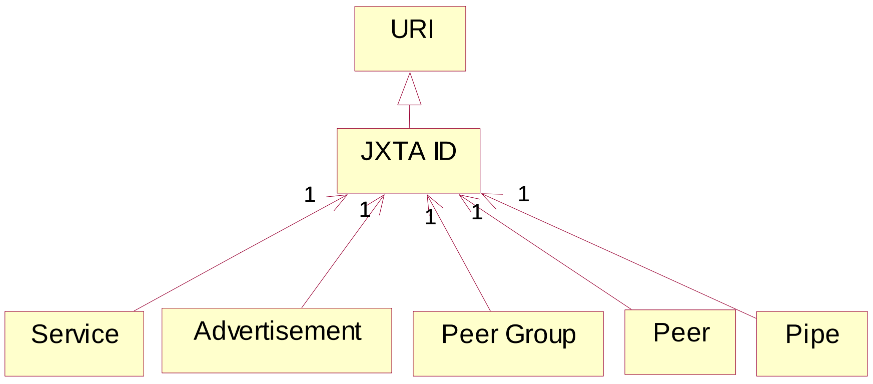

Figure 17: JXTA Elements

2.4.2 Gnutella

Gnutella [Gnutella] is a simple file sharing protocol. We consider the Gnutella file sharing service as adhering to the first definition of p2p service. More specifically it represents the coarse functionality of the Gnutella system, which is file searching and downloading.

The following figure presents the abstract model of the Gnutella file sharing service, which resides on the peers of the Gnutella network. Peers of the Gnutella network contain files and exchange messages, specified by the Gnutella protocol.

The Gnutella protocol [Gnutella] specifies the processes that are used for bootstrapping a peer, querying for available resources and retrieving requested files. Therefore, upon instantiation a new peer this peer should know the IP address of another Gnutella peer in order to connect to the p2p network. After connecting to the network the peer sends a message (ping) to advertise its presence to all the peers it knows. Every peer, which receives a ping message, replies with a similar pong message that contains the number and the total size of files the peer has.

When a peer wishes to find a file it submits a keyword-based query to the other peers it knows. Pending on if they have such a file, these peers may respond with results and will forward the query to other peers within their knowledge. In order to avoid flooding Gnutella has equipped its queries with a Time-To-Live (TTL) field which specifies the max number of hops a query may have within the p2p network.

If a resource is found and it is selected for downloading, a direct point to point connection is opened between the client and the host of the resource, and the file is downloaded directly over HTTP.

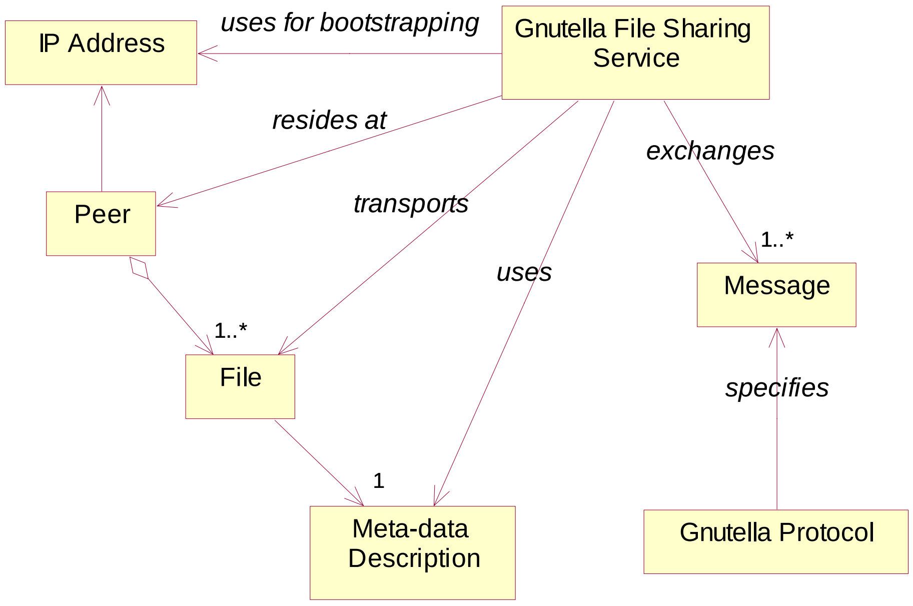

Figure 18: Abstract model for Gnutella file sharing service

When a machine hosting a resource cannot accept HTTP connections because it is behind a firewall, it is possible for the client to send a "Push-Request" packet to the host, instructing it to make an outbound connection to the client on a firewall-friendly port, and "upload" the requested resource, as opposed to the more usual client "download" method.

2.4.3 Edutella

Edutella [Edutella] is project that aims to provide a multi-staged effort to scope, specify, architect and implement an RDF-based metadata infrastructure for JXTA p2p applications. An initial implementation of this infrastructure has been incorporated in the development of a p2p system for the exchange of educational resources. The exchanged resources are described using schemas like IEEE LOM, IMS, and ADL SCORM.

The provided system was founded on the JXTA platform and is based on the exchange of RDF meta-data. It provides specific services which complement the JXTA layers that are illustrated in Figure 19.

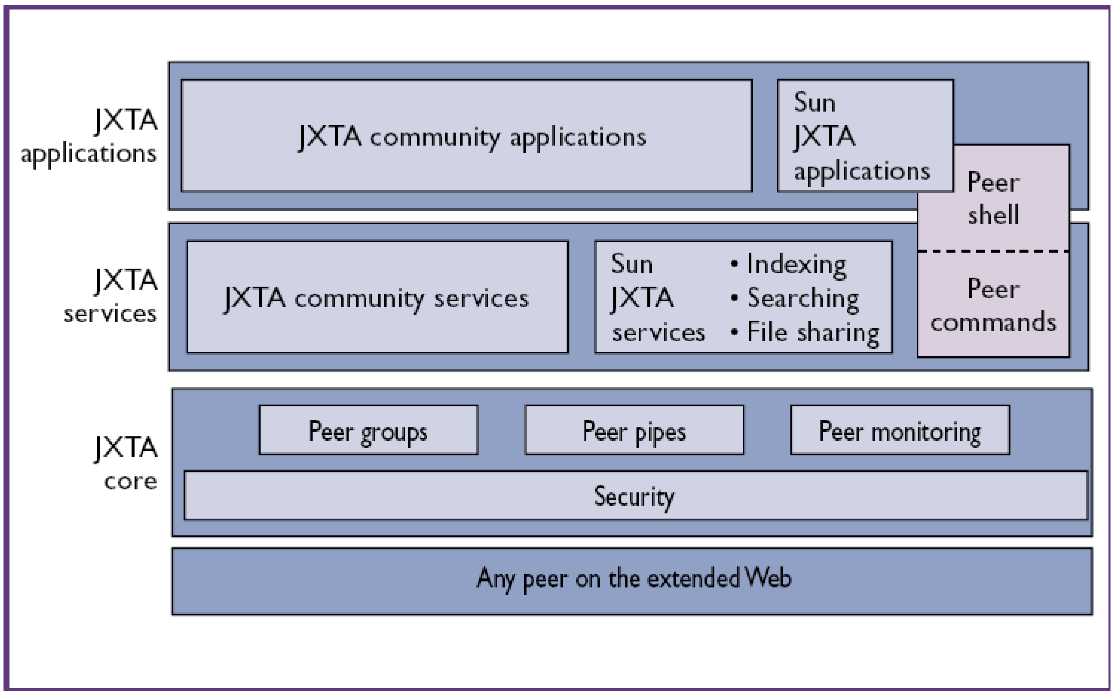

Figure 19: JXTA layer

Edutella Services (described in web service languages like DAML-S or WSDL, etc.) complement the JXTA Service Layer, building upon the JXTA Core Layer. Edutella Peers lie in the Application Layer and utilize the functionality provided by the Edutella services as well as possibly other JXTA services.

The initially specified Edutella services are the following [Edutella]:

- *Query Service:* standardized query and retrieval of RDF metadata (the core Edutella sevice)
- *Replication Service:* provides data persistence / availability and workload balancing while maintaining data integrity and consistency

- *Mapping Service:* translates between different metadata vocabularies to enable interoperability between different peers
- *Mediation Service:* defines views that join data from different meta-data sources and reconcile conflicting and overlapping information) and Annotation Service (annotate materials stored anywhere within the Edutella Network

Edutella peers are highly heterogeneous in terms of the functionality (i.e. services) they offer. Thus every peer provides a kind of local repository for RDF triples (e.g., a relational database) as well as a kind of local query language (e.g. SQL). Additionally peer might offer more complex services such as annotation, mediation or mapping services [Edutella].

The Edutella query service is the most basic service within the Edutella network. Peers register the queries they may be asked through the query service by specifying supported metadata schemas (e.g., “this peer provides metadata according to the LOM 6.1 or DCMI standards”) or by specifying individual properties or even values for these properties (e.g., “this peer provides metadata of the form dc title(X,Y)” or “this peer provides metadata of the form dc title(X,’Artificial Intelligence’)”). Queries are propagated through the Edutella network to the subset of peers which have registered their interest in this kind of queries. The resulting RDF statements / models are sent back to the requesting peer.

Edutella provides a set of wrappers, which are used for translating queries and results from the common Edutella format to the local format of a peer and vice versa, as well as a set of JXTA-based libraries that are facilitating the connection of peers to the Edutella p2p network.

To handle queries the wrapper uses the common Edutella query exchange format and data model for query and result representation. For communication with the Edutella network the wrapper translates the local data model into the Edutella Common Data Model (ECDM) and vice versa, and connects to the Edutella Network using the JXTA p2p primitives, transmitting the queries based on the ECDM in RDF/XML form (Figure 20). The ECDM is based on Datalog which is a non-procedural query language that shares with relational databases and with RDF the central feature, that data are conceptually grouped around properties.

In order to handle different query capabilities, several query exchange language levels (RDF-QEL-i) are defined describing what kind of queries a peer can handle (conjunctive queries, relational algebra, transitive closure, etc.)

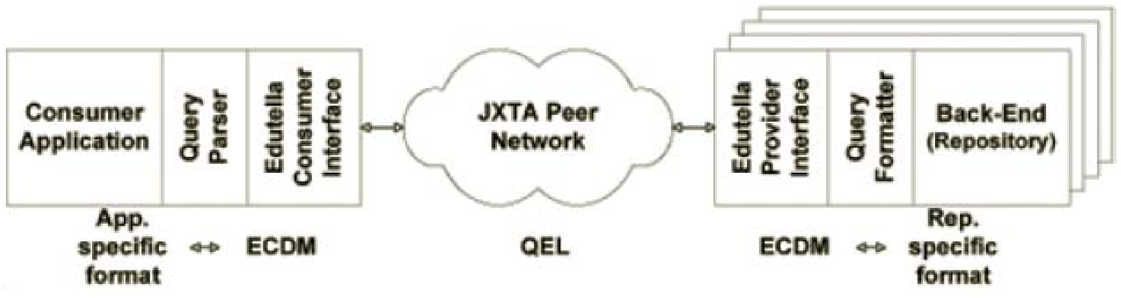

Figure 20: Query processing in Edutella

In Figure 21 we present the abstract model for Edutella service, where the term “Edutella service” represents the process of RDF-based query registration and exchange that we have already described (it does not refer to any specific Edutella service).

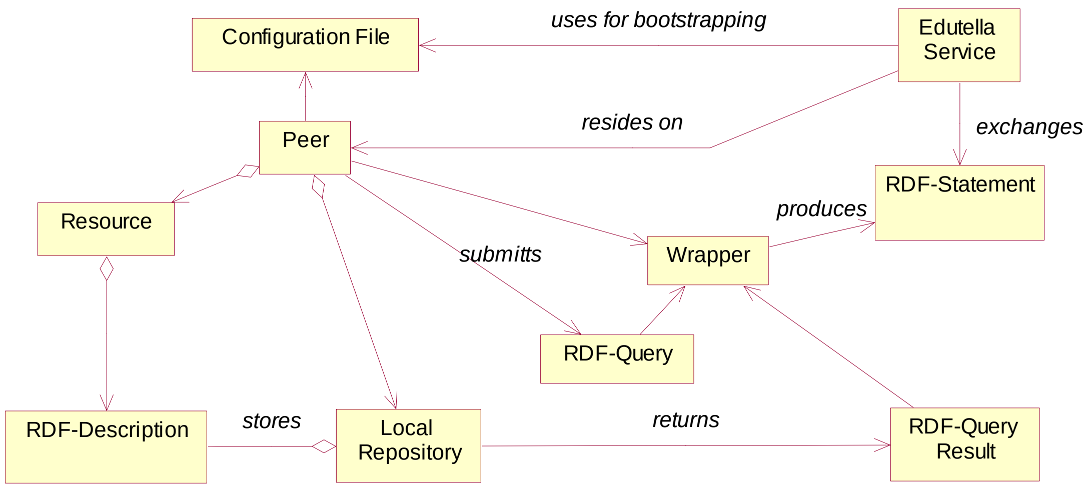

Figure 21: Abstract model for Edutella

2.5 Sodium Platform Supported Models

The state of the art analysis brings up a set of differences among and within each type of service respectively. It is crucial at this stage, to pinpoint the service model of each type of service that will be our basis hereafter.

Therefore, for each of the addressed types of services we have:

- *Web Services*: The current state of the art reveals that there are no drastic changes within the basic standards and protocols that are used. However, most of these standards are under update. Hence, we need to specify which version of the WSDL, SOAP and UDDI we will use.

  Therefore, for the WSDL we will use version 1.1, for SOAP we will use version 1.1 and for UDDI we will use version 2.0.

- *Grid Services*: As far as grid services are concerned, the current status and trends in the grid service community show that the OGSI model is becoming obsolete, whereas the WSRF model is prevailing. Therefore, it is appropriate to use as a basis for our work the WSRF model.

- *P2P Services*: Among the investigated p2p platforms and/or systems that satisfy (partly or fully) the criteria that have been defined in section 2.4 the most appropriate platforms are Edutella and JXTA. These two platforms provide an inherent support for the notion of service and this nominates them as candidates for the provision of p2p services.

  Yet, Edutella is a platform that has been built on JXTA and it incorporates web service standards and protocols for the provision of services. Thus, although Edutella is a suitable platform for the provision of p2p services, the provided services are similar to web services and to JXTA services. Considering all the above JXTA will be used hereafter as the selected platform for the provision of p2p services.

3. COMPARISON OF SERVICE MODELS

An assessment of the service models of each of the addressed service oriented technologies brings up a set of common features and properties that may be regarded as the common denominator of the web, grid and p2p services. Nevertheless, apart from this set of common features there is a plethora of differences.

This section will elaborate on, classify and document the common and distinct concepts introduced by the addressed service oriented technologies.

3.1 Similarities

According to the definitions that were given a service is regarded as a software system that exchanges messages, which are usually XML formatted, resides at a specific network address and has a description that may be an XML formatted document.

Descriptions are all published to specific registry/publishing services, which are used by service requestors for the discovery of available services. A service description contains information that can be used for the identification and invocation of a service. A description conveys information, such as the specific endpoint where a service resides, the protocol that can be used for the message exchange and text descriptions providing human readable information about the service.

In some cases, the specification of the message exchange mechanism may not be explicitly described in a description document, e.g. in P2P services an implied scheme is used. In such cases information related to the service endpoint or the protocol that is used is inferred by the underlying platform.

Exchanged messages are composed of parts, which are namely the header and payload information parts. The header part conveys information that is manipulated by the intermediate nodes/middleware, which transport the messages. Such information could be routing information, security or transaction context related information. The payload part of a message conveys information that is used by the service or by its client. This information is application specific and it normally abides by data types that are specified by the platform (simple types, e.g. Strings, Integers, etc) or the service provider (complex types, e.g. Addresses, Contacts, etc).

3.2 Differences

Despite the similarities that have been described before there is a surfeit of differences, which stem from the extensions and modifications incorporated by each type of service for the provision of their respective characteristics and properties. This set of discrepancies spans across all aspects and levels of abstraction of a service.

In order to facilitate our study we will introduce a list of viewpoints that will guide us through the identification and classification of the differences that exist among the addressed types of services (i.e. web services, grid services, p2p services). The set of viewpoints consists of:

1. *Goals and constraints*: This viewpoint focuses on the business and technical goals pursued by each type of service and their respective constraints in doing so.

2. *Descriptions*: This viewpoint focuses on the information that is used for the description of a service. It may be further broken down to:
    a. *Syntactic features and properties*: That focuses on the syntactic features which are incorporated by each type of service.
    b. *Semantic features and properties*: That focuses on the semantic features which are incorporated by the addressed types of services.
    c. *QoS features and properties*: That focuses on the illustration of the QoS features of the addressed types of services.

In the following we will identify the set of differences among the investigated types of services for each the aforementioned viewpoints.

3.2.1 Goals and Constraints

Focusing on the goals and constraints of each type of service we can point out some basic differences:

- **Service Model**: The two categories that have been used for the classification of service models are namely the "*Stateless*" and "*Stateful*" models. Web services are an ardent supporter of the stateless service model, whereas grid services of the stateful service model. As far as p2p services are concerned they are leaning towards the stateful service model, though there are implementations that pursue the stateless model
- **Intended Clients**: As far as web services are concerned, an invoker could be any machine connected to the net. A web service client needs only to have the necessary infrastructure for exchanging messages (e.g. SOAP messages) with a web service. On the other hand, when it comes to p2p and grid services there is another constraint inflicted. P2P service clients should be members of a p2p network, before invoking a service, whereas a grid service client should be provided with the necessary credentials for invoking the service and using the resources provided by a grid. This implies that apart from invoking a service a client should also be a member of a respective p2p network or have the appropriate credentials to use the resources of a grid.

3.2.2 Descriptions

Based on the state of the art analysis presented in section 2 we may easily spot the differences among the investigated types of services related to their descriptions. These differences scatter across syntactic, semantic and quality of service domains.

In the following we will present the identified differences within each of the specified domains.

3.2.2.1 Syntactic Features

The domain of syntactic features that are used in service description yields an ample of differences. These differences stem from the discrepancies of the incorporated service models along with that of the infrastructure that is leveraged by each service type. Therefore, a thorough look into the investigated types of services brings up the following:

- **Resource element**: Grid services adhere to the stateful service model, whereas web services to the stateless one. In order to support the stateful service model the grid service community has introduced the concept of resource. This is a ‘first order’ element of the grid service technology with ‘value’ equal to that of the service. The management of a resource element is an important aspect of the grid service technology that causes a set of ramifications which need to be tackled. Such ramifications are resource lifetime management, description and discovery.

  In order to properly address the features of the resource element and of its related aspects the grid service model introduced a set of syntactic elements such as the ones presented in section 2.3.

- **Peer to Peer elements**: P2P services on the other hand have incorporated syntactic concepts which reflect their network topology. For a p2p service client it’s important to be knowledgeable of either the peer that is offering the service or the peer group that is supporting the service. Therefore p2p service descriptions have incorporated elements such as the ones presented in 2.4 or the ones depicted in Figure 15, Figure 16 and Figure 21.

- **Service structure**: Another source of discrepancies is the structural elements that a service consists of. According to the grid and web services models a service is decomposed into an interface, which consists of a set of operations. An operation groups the set of messages that are exchanged among the service and its clients for its invocation.

  However, the service structure that is incorporated by most of the p2p service models[3] is a suppressed version of the one used by web and grid services. A p2p service is composed of a single interface and a single operation, which are not usually described in any service description document[4]. Furthermore, the messages that are exchanged among a p2p service and its clients are not usually described also, since the client side middleware that is responsible for the service invocation has this knowledge preconfigured within it.

- **Message structure:** SOAP is the prevailing message structure that has been mostly incorporated by web and grid services and by some p2p service technologies. According to SOAP, a message consists of an envelop element which is composed of a header and a body part. The header and body parts may contain a set of header and body entries respectively. Yet, the messages that are exchanged among a p2p service and its clients may not have a common container such as the envelop or the header and body parts of SOAP [JXTAv2 2003], though they be composed of multiple entries.

3.2.2.2 Semantic Features

The semantic web paradigm [BerHenLa01] has introduced a new approach in describing internet resources that facilitates their automated discovery and the reasoning on their

[3] E.g. JXTA or Gnutella follow this approach whereas Edutella uses the web and grid services paradigm

[4] With the exception of the Edutella service model which doesn’t pursue this approach.

provided features and properties. Each of the addressed service types has tackled the issue of semantics in various ways.

Judging on the approaches that have been pursued by the investigated service types we end up with:

- **Web service semantics**: With the past years there have been proposed many approaches for applying semantics in web services. OWL-S [OWLS 2004], WSMO [WSMO] and WSDL-S [WSDL-S] are some of the most well known paradigms that have been used. These approaches address the semantic annotation of a service by introducing additional elements.
- **Grid Service semantics**: As far as grid services are concerned semantics have to be applied in the description of services along with the resources that are shared in a grid. The existing approaches in providing semantic annotation for grid services have been based on the use of technologies that are already introduced for web services. Thus, contemporary grid service descriptions are based on the use of OWL [OWL 2004] and RDF [RDF 2004] or OWL-S[OWLS 2004] and WSMO[WSMO].
- **P2P service semantics**: P2P service platforms have used many approaches in applying semantics. Most of the existing platforms have incorporated meta-data that convey semantic meaning for their elements which are usually attached to the resources of a p2p network, such as files. Furthermore, the applied concepts do not usually come from a formally described ontology such as an OWL[OWL 2004] ontology, but rather from an arbitrary, application specific set of concepts. Nevertheless, there have been platforms such as Edutella that incorporate formally described ontologies which are also applied on service descriptions.

3.2.2.3 QoS Features

Another field of differences among the investigated types of services is the description of quality properties. Each type of service has incorporated various approaches for addressing the description of quality features. The investigation of the web, grid and p2p service types has end up with the following:

- **Web Service QoS**: Web services have so far incorporated many approaches in specifying quality aspects. These approaches are associating quality attributes on a service as a whole and on its constituent operations.
- **Grid Service QoS**: QoS specification of grid services on the other hand has been based on the use of meta-data elements that are attached to the resources which are provided in a grid. These meta-elements provide a “proprietary” approach for the specification of quality features of services and resources. Apart from these primitive approaches for the specification of quality in grids there have been other approaches that provide advanced quality specification and support management of services and resources along with the underlying network infrastructure. Such approaches are G-QoS [G-QoSM 02] that has been proposed by Rashid J. Al-Ali and provides concepts similar to those in [BhSnCh] and the Globus Architecture for Reservation and Allocation (GARA) that has been described in [FoKesNahRoy].

- **P2P Service QoS**: When considering quality features for p2p services there are not so many approaches addressing their specification. This issue is usually addressed by specific p2p applications which introduce quality aspects that are of importance with respect to these applications (e.g. network bandwidth, latency, throughput, etc). Furthermore, some contemporary p2p platforms provide meta-data that are tackling the specification of quality aspects of the shared resources (e.g. sampling rate for mp3 files, etc.).

  Yet, there have been proposals for algorithms and systems that take into account either the underlying network's quality features or application specific quality attributes for the calculation of an aggregated quality that is used either for the selection of resources or for the establishment of communication paths among the peers of a p2p network [GWBBCBCG01][XuNa 02][GuNah 02].

4. GENERIC SERVICE MODEL DEFINITION

Following on the observations related to the similarities and discrepancies among the investigated types of services, an appropriate structure for the service model is a layered one. A layered structure facilitates the specification of a basic layer which consists of all common concepts and the provision of additional layers on top of this core layer for the specification of the distinct features incorporated by each type of service. As far as Semantics, QoS, Trust and Security along with Management aspects are concerned these issues are orthogonal with respect to the service model and may be applied to the concepts of each layer respectively.

Thus, a representation of the Generic Service Model's (GeSMO) structure is presented in Figure 22. This model consists of two layers namely the core and extension layers.

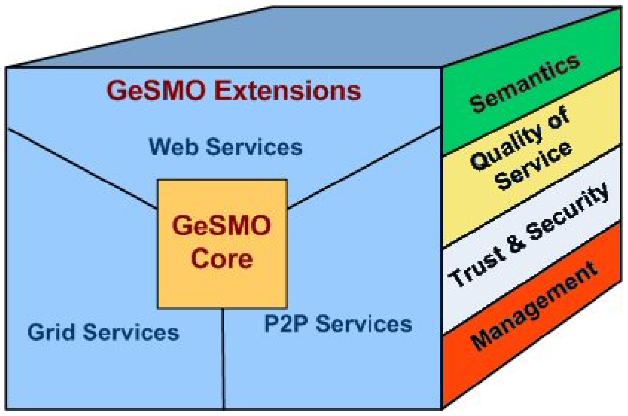

Figure 22: GeSMO structure

The core layer consists of all common features identified in previous sections (see section 3.1). The extension layer on the other hand is divided into three sectors, which provide the distinct characteristics supported by each of the investigated types of services (see section 3.2). As far as orthogonal aspects such as Semantics, QoS, Trust and Security and Management are concerned these may be attached to any element of the GeSMO so as to provide needed extensions.

In the following we describe each of the identified layers and the relationships among their elements. For each element we provide a definition and a description of its relationships with other elements along with an explanation. Consequently, we present a set of conformance constraints that need to be satisfied by the languages that will be based upon this model so as to achieve compliance with the model.

4.1 Core Service Model Layer

The fundamental concept of this layer which binds together all the other elements is the service. As it has been argued above a service may be conceived as a software system that i) has some functional and non-functional properties, ii) has a certain behavior, iii) resides at a network address, iv) is described by a description document and v) is accessible by its clients via messages. An investigation of the service concept and of its integral components (i.e. description, message) from multiple points of view provides a set of additional elements related to the service notion.

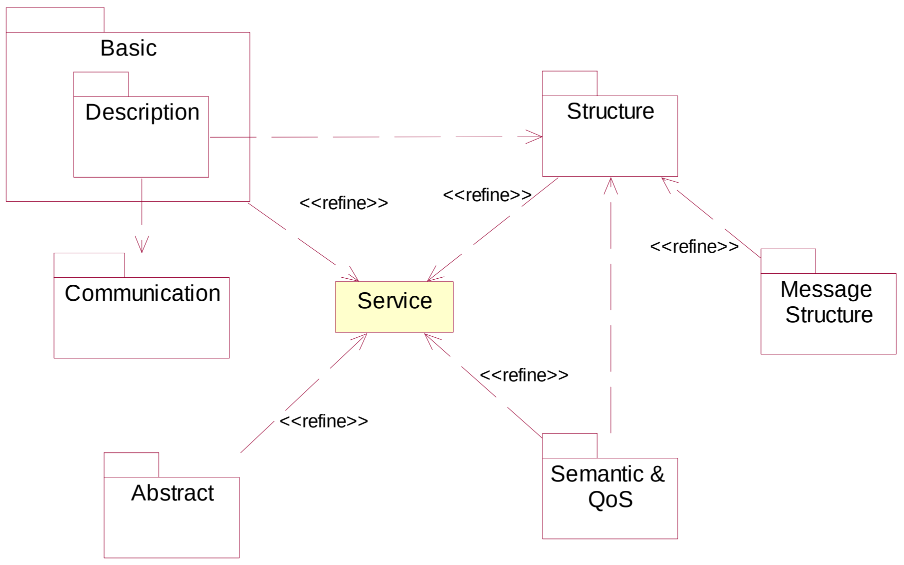

Figure 23: Modules of core model

The set of additional elements that stem from the multi-viewpoint evaluation of the service concept can be grouped into sets of related elements which address a service's aspect. The set of viewpoints that have been used within this investigation are:

- Abstract: This viewpoint looks into the service notion from an abstract point of view and tries to identify its relationships with elements of the software engineering field
- Basic: This viewpoint pinpoints the minimal set of elements that need to be provided. The elements that are identified within this viewpoint may be further analyzed in other sub-viewpoints.
    - Description: This viewpoint focuses on the elements that are related to a service's description
- Structure: The structure viewpoint identifies the structural elements that a service may comprise
- Semantics & QoS: This viewpoint identifies the elements of the service model that may have semantics and qos annotations
- Message Structure: This viewpoint provides a look into the structure and the elements of messages that are exchanged among a service and its clients
- Communication: The communication viewpoint identifies the elements that are related to the underlying network communication details i.e. communication protocols that are used, network address, message wire format, etc.

Figure 23 illustrates the set of modules that group together the concepts of each of the identified viewpoints and their respective associations. In the following we provide a specification of the core concepts as they have been perceived from the identified viewpoints.

4.1.1 Abstract Service Model

The abstract service model depicts a service and its related elements as they have been identified from an abstract point of view. Specifically it tries to relate the service notion with a set of fundamental concepts that are pertinent in the description of a software system.

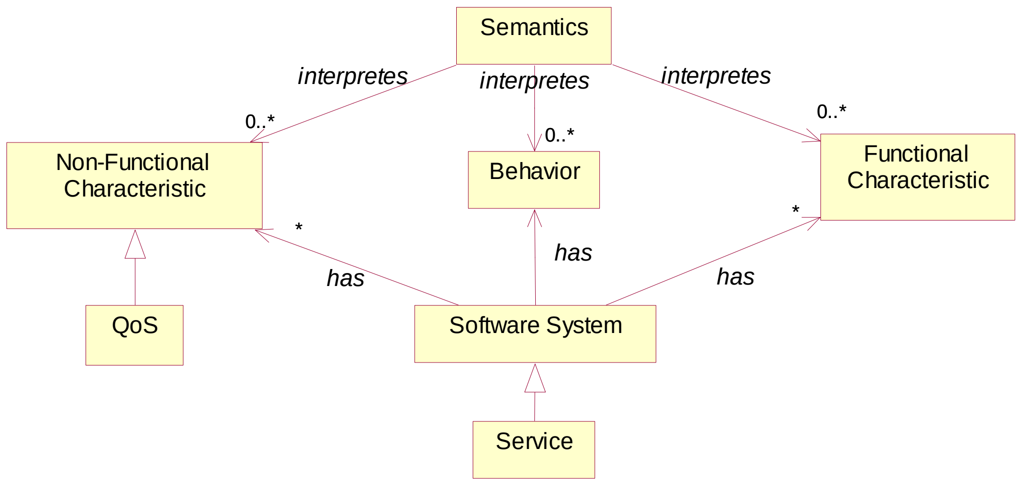

Figure 24: Service features

Functional Characteristics

Definition

A functional characteristic for the purposes of this model represents the attributes of the system that contribute in the fulfillment of the system's functionalities.

Relationships

- A software system has functional characteristics

Explanation

Functional characteristics are elements of the system which contribute in the system's operation. For example, a functional characteristic could be the Operating System that is used by a system, the network address where a web server resides or the operations that a component provides to its users.

Non-Functional Characteristics

Definition

Non-Functional characteristics represent non-functional properties of a system.

Relationships

- A system has non-functional characteristics

Explanation

Non-Functional characteristics represent non-functional properties of the system that either provide additional information about a system unrelated to its operation or contribute in the user's satisfaction. Such properties could be contact details about a system's provider or the performance and reliability properties as perceived by a system's user.

Behavior

Definition

Behavior in this model represents the set of system's reaction (or actions) to external or internal events.

Relationships

- Every system has a behavior

Explanation

A software system's behavior consists of the system's reactions to external or internal events. Such events could stem from the system's normal or abnormal operations.

Software System

Definition

A software system is an implementation of a system through a programming language (see Figure 24). Such a system provides useful functionalities to its users.

Relationships

- A software system has a behavior
- A software system has functional characteristics
- A software system has non-functional characteristics

Explanation

A software system is an application or computer program that realizes specific functionalities and has certain non-functional characteristics.

This concept is similar to the Agent element identified in [w3c 2004].

Semantics

Definition

Semantics represent information which has a formally defined meaning. Semantics may be attached to any element and annotate it with semantic interpretation.

Relationships

- Semantics may interpret a systems functional characteristics
- Semantics may interpret a system's non-functional characteristics

- Semantics may interpret a system's behavior

Explanation

Semantics annotate the elements that they are attached to with formally specified meaning that can be machine and human understandable [BerHenLa01].

QoS

Definition

QoS within this model represent quantifiable non-functional aspects of a system. Quality of Service may be associated with a system as a whole or to a system's components.

Relationships

- QoS is a quantifiable specification of non-functional characteristics

Explanation

QoS are quantifiable non-functional aspects of a system. An example of such an element could be a performance indicator which describes the number of requests that are executed by a system within a given time interval.

Service

Definition

Within this viewpoint a service is a software system holding has some functional and non-functional characteristics and exposing a specific behavior.

Relationships

- A service is a software system

Explanation

Within the context of this viewpoint a service is a software system. As such it has some functional and non-functional characteristics and exhibits a certain behavior. All of its constituent components i.e. behavior, functional and non-functional characteristics may be annotated with semantics, whereas some of its non-functional characteristics may be represented as quantifiable quality properties.

4.1.2 Basic Service Model

The basic service model depicts a minimal set of concepts and their respective relationships that define the concept of service (see Figure 25). This set of constructs according to [Vogels 2003] suffices for the invocation of a service, but they need to be further annotated so as to facilitate the whole set of operations that are supported by the service model i.e. publication, discovery, invocation, composition, etc.

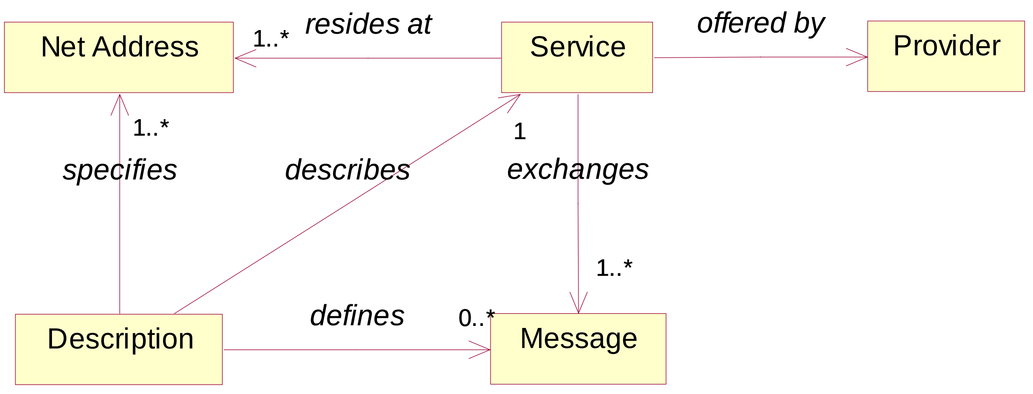

Figure 25: Minimal service model

Provider

Definition

A provider is an entity that provides a service (see Figure 25).

Relationships

- A provider offers services

Explanation

The provider element represents an entity that provides services. Such an entity may be a person or an organization.

This concept is similar to the Provider Entity element of the [w3c 2004].

Net Address

Definition

A Net Address is an element which represents a specific network address (see Figure 24).

Relationships

- A service resides at a specific net address
- A service description specifies the net address of a service

Explanation

A net address element represents the network endpoint where a service resides. This element may be represented as a URI or in another format that is conceivable by the underlying communication infrastructure. A service client uses this network address so as to exchange messages with the service.

This element is similar to the Address element of [w3c 2004].

Message

Definition

A message is the smallest data element that is exchanged among a service and its clients (see Figure 25, Figure 26 and Figure 30).

Relationships

- A service exchanges messages
- Descriptions specify messages

Explanation

A message is the smallest unit of information that is exchanged between a service and its consumer. Messages are used for conveying information that is needed by the service for performing its provided functionality.

Service

Definition

A service within this viewpoint is a self-described system that is offered by a provider at a specific network address where its respective clients can invoke it through messages.

Relationships

- A service is offered by a provider
- A service resides at a network address
- A service exchanges messages
- A service is described by a description document

Explanation

According to this model a service represents a system that is offered by a service provider. This service is accessible at a network address and exchanges messages with clients. A client is able to identify what the service does and how to invoke it through the description document which provides specific details about the service.

Description

Definition

According to Figure 25 a description is a document that provides information about a service such as the network address where it resides and the type of messages that are exchanged during its call.

Relationships

- A description specifies the network address where a service resides
- A description describes the messages exchanged by a service
- A description describes a service

Explanation

A description is a document or a set of documents which provide information about a service that can be used by clients so as to identify what the service offers and how somebody can invoke it.

4.1.2.1 Service Description Model

The service description model provides a detailed specification of the information that a description document may convey. A description document may provide descriptions or links to other documents for the whole or for parts of the information that is depicted in Figure 26.

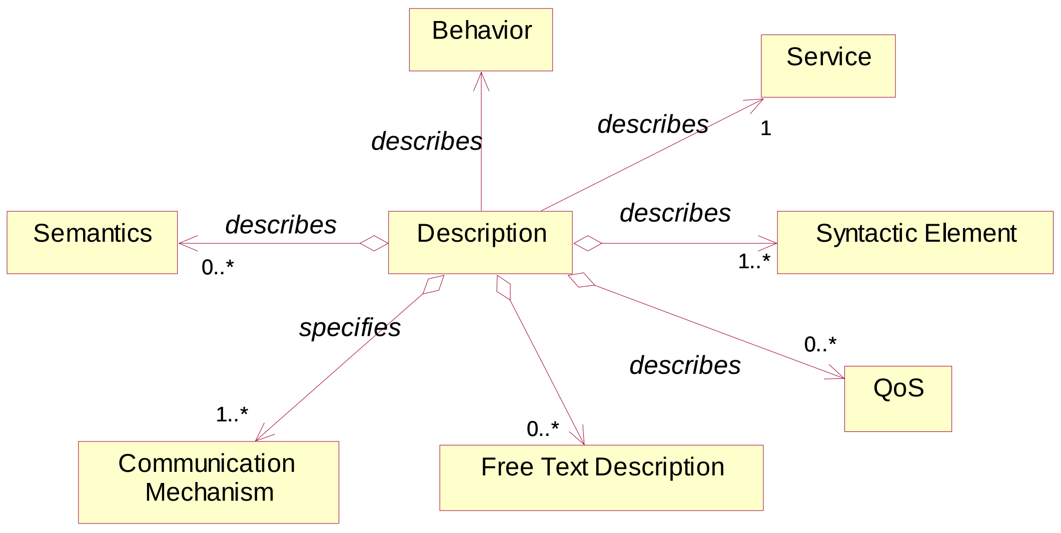

Figure 26: Service Description

Communication Mechanism

Definition

A communication mechanism within this viewpoint represents the underlying network details that are used by a service and its clients for their communication.

Relationships

- A description provides information about the communication mechanisms that are used

Explanation

A communication mechanism within the context of this viewpoint represents the network details that need to be known to a client so as it can invoke a service.

Service

Definition

A service according to Figure 26 is a self-described software system.

Relationships

- A service is described by a description

Explanation

A description document provides information about the characteristics of a specific service. The information that is described within a description document is accessible to the service clients so as they are aware of what a service does and how to invoke it.

Behavior

Definition

A description document provides information for a service's behavior which represents a service's actions and reactions to external or internal events.

Relationships

- A description document may describe a service's behavior

Explanation

The behavior of a service or a system in general may be described through a description document. A description document conveys all necessary information about a system's actions and reactions to external or internal events.

Semantics

Definition

Semantics represent information that provides formal interpretation to the elements that they are attached to.

Relationships

- A description document may contain semantics

Explanation

Semantics represent formally defined pieces of information that may be attached to the elements of a description document and annotate them with semantic interpretation.

QoS

Definition

QoS within this viewpoint represents pieces of information that provide measurable values to the non-functional characteristics of a service or of its components

Relationships

- A description document may convey QoS

Explanation

QoS within the context of this viewpoint represent information with measurable values for a service's non-functional characteristics. QoS attributes may be attached to a service or to its constituent elements. Thus, a description document which provides information about a service's components may describe their QoS attributes as well.

Syntactic Element

Definition

A syntactic element represents a service's structural component that a service client needs to be aware of.

Relationships

- A description document describes a service's syntactic elements

Explanation

A syntactic element is an abstraction of a service's structural components i.e. interfaces, operations, messages. A description document should provide the details of these elements so that a service client can invoke a service.

Free Text Description

Definition

A free text description represents pieces or text that provide human readable information

Relationships

- A description document may contain free text descriptions

Explanation

A free text description provides human readable pieces of information

Description

Definition

A description is a document or a set of documents that provide structured information for a service. As it is illustrated in Figure 25 and Figure 26 a description provides information on what a service does and how it can be invoked.

Relationships

- A description describes a service
- A description specifies the net address where a service resides

- A description may describe a service’s behavior
- A description specifies a service’s syntactic elements
- A description specifies the communication mechanisms that are used by a service
- A description may contain additional free text descriptions
- A description may describe a service’s semantics
- A description may describe a service’s quality of service

Explanation

A description provides valuable information about what a service does and how can somebody invoke it. It contains information about the messages that are exchanged, the communication mechanisms that may be used and the endpoints where messages should be transmitted to. A description document is usually structured as an XML document.

Descriptions may also provide additional information, such as human readable or machine understandable data that describe other features such as the semantics or quality of service.

4.1.3 Structure Model

The structure model defines the set of structural elements that a service may be broken down into. The structural elements that have been identified are depicted in Figure 27.

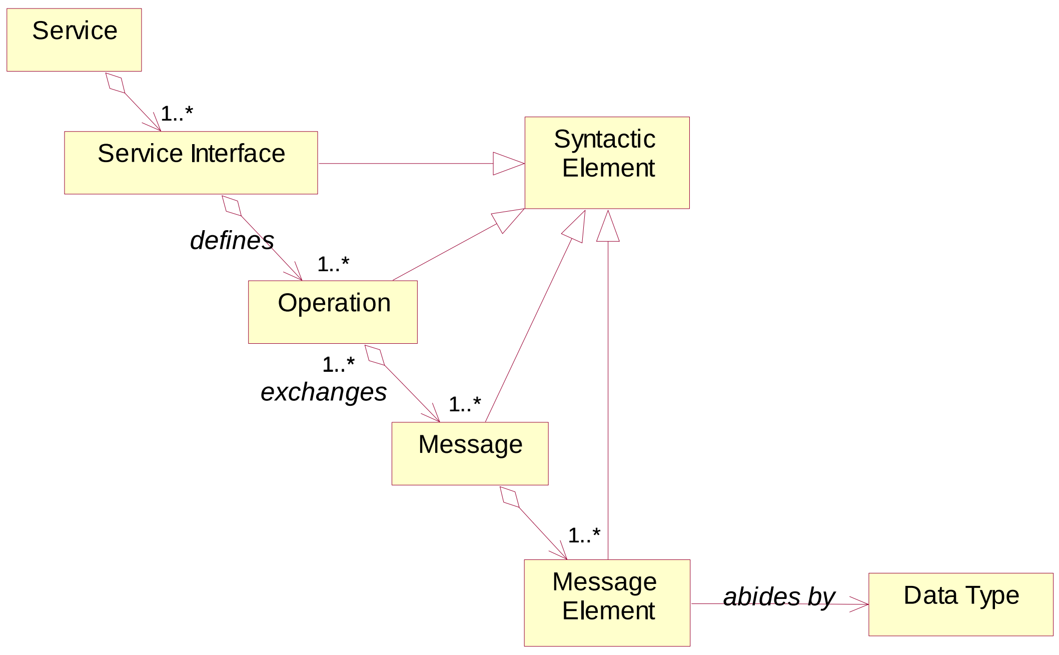

Figure 27: Service Structure

Service

Definition

A service is a software system which is offered to the net by a service provider at a specific network address. Clients are based on the service description in order to access it through message exchanges. A service has an interface and a behavior, which may be described by a service description document.

Relationships

- A service has an interface
- A service may have semantics
- A service may have QoS

Explanation

A service is a software system that provides specific functionalities to its clients. Using a service's interface clients may invoke the operations that the service provides through messages.

Service Interface

Definition

A service interface specifies the set of operations that a service provides to its clients along with the messages that may be exchanged.

Relationships

- A service may have one or more interfaces
- An interface has a set of operations

Explanation

An interface specifies a service's set of operations and the messages that are available to its clients. Clients using the specified interface are able to interact with the service.

Operation

Definition

An operation represents a simple action that may be performed by a service. In order to invoke this action a set of messages need to be exchanged among a service and its clients according to a specific communication pattern.

Relationships

- An interface specifies a set of operations
- An operation exchanges a set of messages

Explanation

An operation specifies a simple form of action that may be performed by a service. In order to invoke an operation a set of messages need to be exchanged among the service and its clients. The order and the type of the messages that are exchanged are specified by the operation element.

This element is similar to the operation element declared in [CGMSW].

Message

Definition

A message element within this viewpoint represents the pieces of information that are exchanged during the invocation of a service's operation.

Relationship

- An operation exchanges a set of messages
- A message is composed of message elements

Explanation

A message represents the pieces of information that are exchanged among a service and its clients during the invocation of a specific operation. The information that is conveyed by a message is related to the service's application logic.

Message Element

Definition

A message element is the smallest part of information that is exchanged among a service and its clients during the execution of a service's operation.

Relationships

- A message is composed of message elements
- A message element abides by a data type

Explanation

A message element within this viewpoint represents the smallest part of information that is exchanged among a service and its client during the invocation of a service's operation. Specifically a message element represents a data-type conforming piece of information that a message is composed of.

Data Type

Definition

A data type represents a formally defined type that is used for the specification of a message element's format and domain of values

Relationships

- A message element conforms to a specific data type

Explanation

A data type represents a user-defined (composite types) or platform-defined (simple types) type that specifies the format and values of conforming parameters.

4.1.4 Semantic and QoS Model

The Semantic and QoS model specifies which of the service's structural elements may have semantic or QoS annotations. The semantic annotations that are associated with a service or its constituent elements may be described in one or more description documents. Within our model, the set of components that may have semantic or QoS attributes attached are depicted in Figure 28.

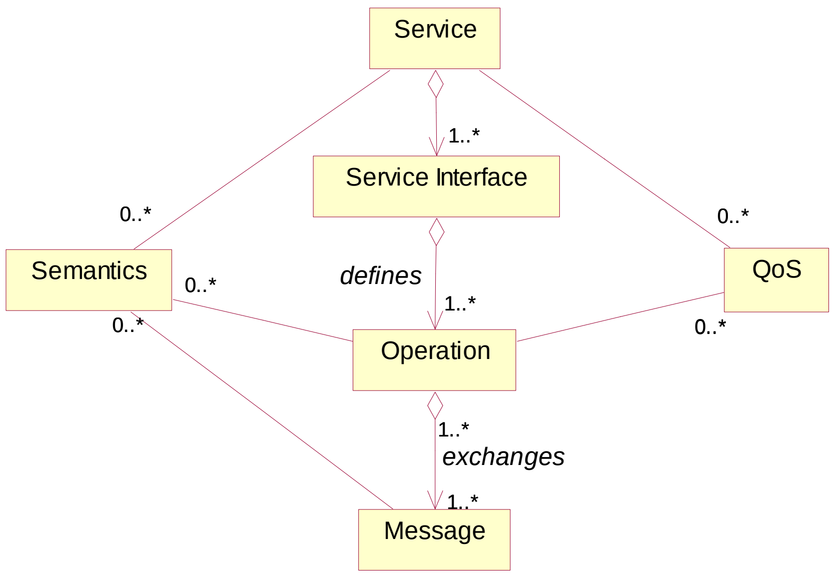

Figure 28: Service Semantic and QoS annotation

Semantics

Definition

Semantics within this viewpoint represent information that provides formally specified meaning to its attached elements.

Relationships

- A service may have semantics
- An operation may have semantics
- A message may have semantics

Explanation

Semantics within this viewpoint represent attributes with formally defined meaning that provide semantic interpretation to the attached elements.

QoS

Definition

QoS within this viewpoint represent information attributes that provide measurable values to non-functional characteristics.

Relationships

- A service may have QoS
- An operation may have QoS

Explanation

QoS within this viewpoint represent information attributes that provide quantifiable values to non-functional characteristics of their attached elements.

Service

Definition

A service element within this viewpoint represents a software system that may have attached semantic and QoS attributes

Relationships

- A service may have semantics
- A service may have QoS
- A service has a service interface

Explanation

A service element within this viewpoint represents a software system that is annotated with semantic and QoS attributes.

Service Interface

Definition

A service interface within this viewpoint represents the set of operations that a service makes available to its clients.

Relationships

- A service provides at least one interface to its clients
- An interface is composed of operations

Explanation

An interface, as it has been also described in section 4.1.3, specifies the set of operation that a service provides to its clients.

Operation

Definition

An operation within this viewpoint represents the simplest action performed by a service, which may be semantically or QoS annotated

Relationships

- An interface is composed of a set of operations
- An operation may have semantics
- An operation may have QoS
- An operation exchanges messages

Explanation

An operation represents the simplest action performed by a service and may have semantic and QoS attributes attached.

Message

Definition

A message represents the set of information that is exchanged during the execution of a service's operation

Relationships

- An operation exchanges a set of messages
- A message may have semantics

Explanation

A message within this viewpoint represents a semantically annotated set of information which is exchanged upon invocation of a service's operation.

4.1.5 Communication Mechanism Model

The communication mechanism model describes the elements and their respective associations that are used in the description of the communication channels among a service and its clients. Figure 29 illustrates the elements that are going to be used within our model.

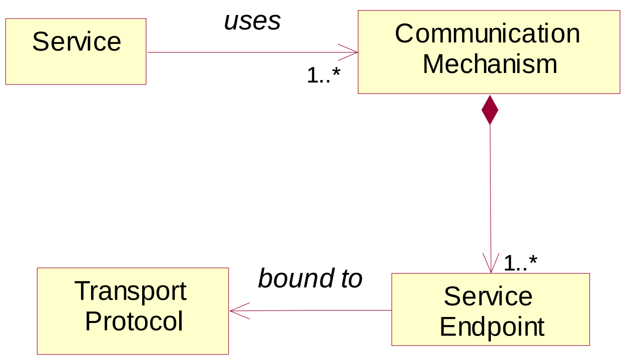

Figure 29: Communication Mechanism

Service

Definition

A service within this viewpoint represents a software system that is accessible to its clients through the net.

Relationships

- A service uses at least one communication mechanism

Explanation

A service within this viewpoint represents a software system that its clients may access it through the net, by using a specific communication mechanism. This mechanism specifies the communication details that the service's clients should use in order to invoke it.

Communication Mechanism

Definition

A communication mechanism represents the underlying network infrastructure that is used for transporting a message along with the actual endpoint where the message should be transmitted to.

Relationships

- A communication mechanism describes the endpoint where a service resides
- A communication mechanism specifies the transport protocols that may be used for a message exchange

Explanation

The communication mechanism element represents the infrastructure that is used for transferring a message among a service and its clients. Such a mechanism specifies the endpoint where a message should be transmitted to along with the transportation protocol that should is used during this exchange. E.g. a communication mechanism could specify that a message is transmitted over HTTP at a specific network address.

A communication mechanism is responsible for transmitting a message to a specific endpoint. Furthermore, the mechanism is also responsible for transforming the message to the appropriate format that will enable its exchange over the underlying network infrastructure.

Service Endpoint

Definition

A service endpoint specifies the network address where a service resides along with the specific protocol that is used when accessing a service. Furthermore, the service endpoint specifies the encodings or formats that are used by the exchanged messages.

Relationships

- A communication mechanism specifies the service endpoints
- A service endpoint is bound to a specific transportation protocol

Explanation

A service endpoint specifies a message's format and the network address where it should be send to, along with the network protocol that will be used for this exchange.

Transport Protocol

Definition

The transportation protocol element specifies the network protocol that will be used for the message transmission.

Relationships

- A communication mechanism specifies the transportation protocols that are used for the message exchanges
- A service endpoint is bound to a specific transportation protocol

Explanation

A transportation protocol represents the underlying communication protocol that will be used for the message exchange. Examples of such a protocol are HTTP, TCP/IP or SMTP.

This element is similar to the Message Transport element specified in [w3c 2004]

4.1.6 Message Structure Model

The message structure model specifies the components of message along with their associations. Figure 30 illustrates the integral components of a message that is exchanged among a service and its respective clients.

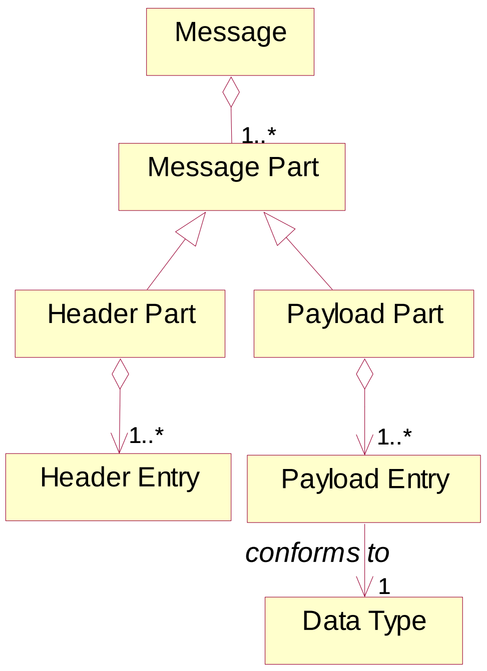

Figure 30: Message Structure

Message

Definition

A message within this viewpoint represents the bundle of information that is exchanged among a service and its client over the net.

Relationships

- A message is composed of message parts

Explanation

A message within this scope represents the bundle of information that is exchanged among a service and a client over the net.

A message consists of various parts, which could be either header parts or payload parts. The header part conveys information that is consumed by the intermediate nodes/middleware that convey the message, whereas the payload part is used by the service and its respective client.

Message Part

Definition

A message part is an abstract container of information that is exchange within a message.

Relationships

- A message consists of at least one message part
- A payload part is a specialization of a message part
- A header part is a specialization of a message part

Explanation

A message part is an abstract container of information that is conveyed by a message. A message part can be either a header part or a payload part. A message should contain at least one message part which should be a payload part.

Header Part

Definition

A header part conveys information about a message that is manipulated by intermediate nodes or middleware that take part in the message exchange.

Relationships

- A header part is a specialization of the message part
- A header part consists of multiple header entries

Explanation

Header parts are used for exchanging information that is not relevant to the service‘s application logic. The header part contains information that is processed by intermediate nodes or middleware that take part in the message exchange. E.g. a header part may convey information such as security context or transaction context related information.

Payload Part

Definition

The payload part of a message contains application related information that is exchanged among a service client and the service.

Relationships

- A message may contain at least one payload part
- A payload part is a specialization of the message part
- A payload part consists of multiple payload entries

Explanation

The payload part of a message contains all the application related information that is exchanged among a service and its clients. The payload part is a container of more than one payload entries.

The payload part resembles to the Message Body element that is specified in [w3c 2004].

Header Entry

Definition

A header entry is the smallest part of header information that may be conveyed by a message.

Relationships

- A header par is composed of one or more header entries

Explanation

A header entry is the smallest part of header information that is conveyed by a message. The information of a header part is unrelated to a service's application logic, has separate semantics and may be independently standardized.

The information of a header entry is tended to be processed by intermediate nodes or middleware of the communication path and can be processed independently of the payload part.

Payload Entry

Definition

A payload entry is the smallest part of formatted information that is conveyed in the payload part of a message.

Relationships

- A payload part contains one or more payload entries
- A payload entry abides to a data type

Explanation

A payload entry represents the smallest part of application related, formatted information that is contained in a message. Payload entries conform to data types that are either pre-defined or custom-made.

Data Type

Definition

A data type represents a predefined or custom made type of data.

Relationships

- A payload entry abides by a data type

Explanation

A data type represents a simple or complex type of data that a payload entry conforms to. Such types could be pre-defined ones like the XML Schema defined types [XMLSchDT] or application related types.

4.2 Extensions

This section will provide a description of the extensions layer that will be built on the core layer. This layer will use the concepts of the core layer as a basis and it will provide appropriate extensions for the concepts that are missing.

The concepts of the extensions layer have been grouped into three modules that are addressing the web, grid and p2p service types respectively. In the following we are presenting the elements that have been introduced for the aforementioned service types.

4.2.1 Web Service Type

Introduction

As it has been specified in section 2.5 the set of protocols that will be used as a basis for the specification of web services consists of WSDL [WSDL 2001], SOAP [SOAP] and UDDI [UDDI v2 DS]. Thus the elements that will be provided will abide by these standards.

The description of the missing concepts for the web service type is a straightforward process. This is because most of the needed elements have been already described in the core layer. Therefore, through some simple extensions that are depicted in Figure 31 we are able to provide the missing concepts.

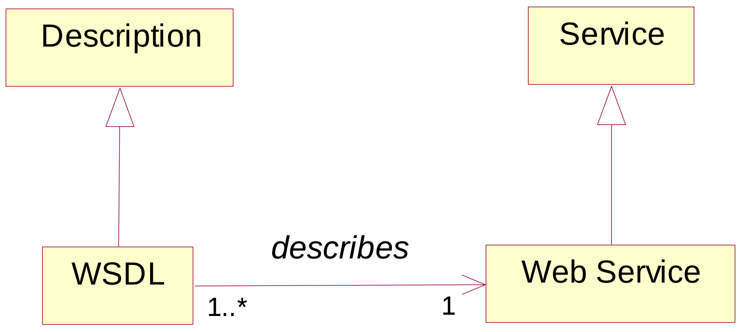

Figure 31: Web Service conceptual model

Web Service

Definition

A web service is a service that is offered to its clients through the use of the de-facto Web service standards such as HTTP, SOAP and WSDL.

Relationships

- A web service is a specialization of the service

Explanation

A web service is a specialization of the service element defined in the core layer. It comprises a collection of URL-located services that can be called over the Internet. Each Web service is available to its clients through the use of de-facto standards such as HTTP, SOAP and WSDL.

WSDL

Definition

A WSDL file is a specialization of the Description element that is based on the WSDL standard [WSDL 2001].

Relationships

- A web service is described by a WSDL file

Explanation

A WSDL file provides the definition of a web service which is basically a grouping of interfaces and operations where different Web service operations can be called. All of these can have one or more bindings to a specific transport protocol where there are three main choices: HTTP Get, HTTP Post or SOAP.

4.2.2 Grid Service Type

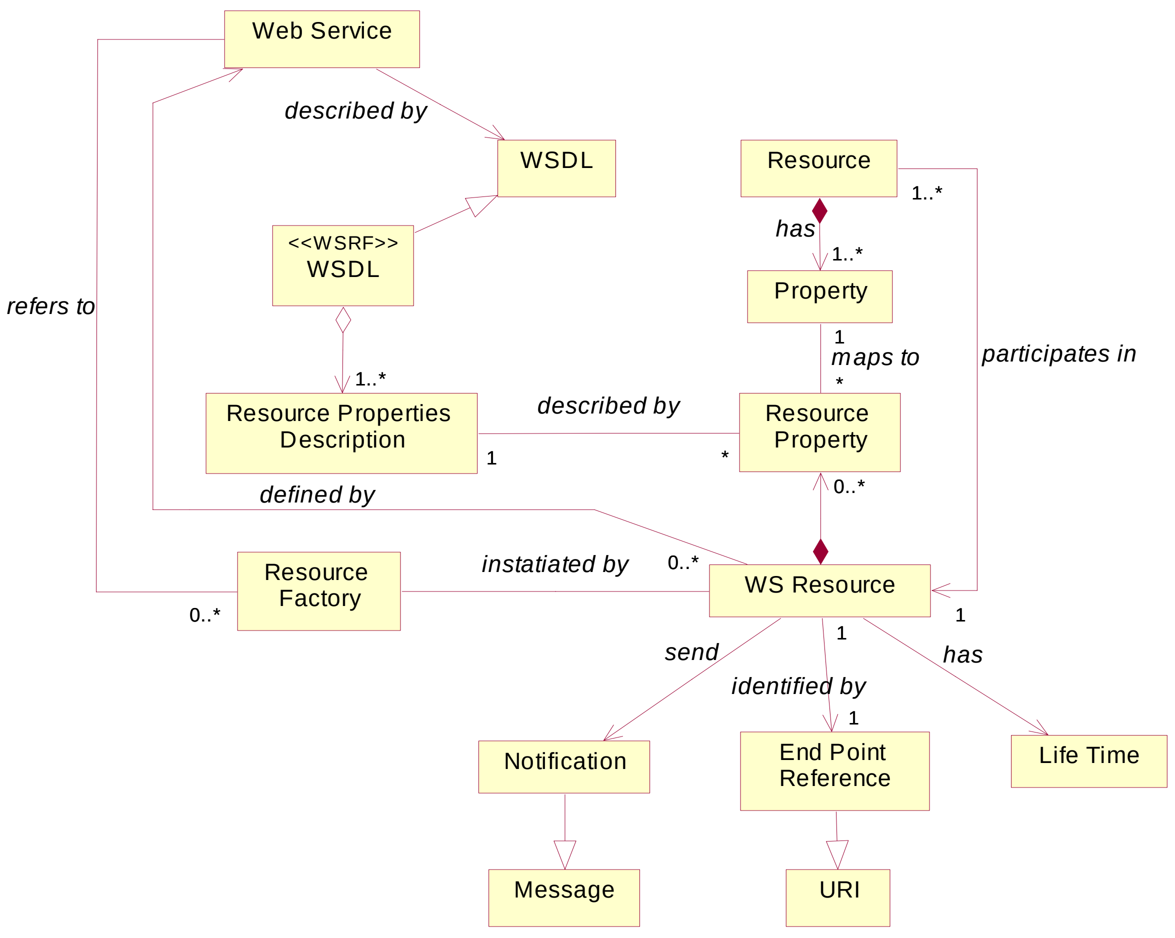

Figure 65: Conceptual Grid service model

Introduction

Figure 65 depicts the relationship between WS-Resource, Resource, Web Service and WSDL. A WS-Resource is a stateful resource loosely coupled with a web service which defines standardized ways to access the resource. The scope of what a resource can be is broad. It can be any hardware or software entity which has a state and whose state is also accessible by means of software. An instance of WS-Resource is defined as a web service as well as an identifier for a specific view on the resource. A client, which instantiates a WS-Resource gets back an identifier defining its view on the resource. It is however a misconception that the resource gets instantiated as this may not be always true.

A Resource has properties which are described/defined within the WSDL document through Resource Properties Definitions. In case the resource is a hard disk, a property may for instance be the number of sectors whereas in case the resource is an implementation of a counter, the current value may be a property. These Resource Properties are mapped to the actual Properties of the Resource. The web service provides the necessary operations to access/modify the properties. It is solely up to the web service to implement how the resource properties are accessed. At no time the properties of the resource are directly modified by a client.

The web service must also provide other standardized/defined operations for resource management, lifecycle management and notifications. These operations are defined in the WS-Notification, WS-ResourceProperties, WS-ResourceLifetime, WS-Addressing standards.

WS-Resource

Definition

A WS-Resource is a web service loosely coupled with a stateful resource.

Relationships

- A WS-Resource is defined by a web service
- A resource participates in a WS-Resource
- Is identified by an endpoint reference
- A WS-Resource sends notifications
- A WS-Resource has a lifetime
- A WS-Resource is identified by an EndpointReference
- Is instantiated by the Resource Factory
- A WS-Resource has Resource Properties

Explanation

A WS-Resource loosely couples a web service with a stateful resource. The web service which is part of the WS-Resource provides standardized operations to manage the resource.

Resource

Definition

A resource in the context of the WS-Resource standard is a resource which has a defined state.

Relationships

- A resource has properties
- A resource participates in a WS-Resource

Explanation

A resource in the context of WS-Resource can be any resource which is accessible by means of software. Examples are a database, Hard-disk, CPU and many more.

Property

Definition

A Property in the context of this model is a property of a resource.

Relationships

- A property is mapped to a Resource Property

Explanation

A property of a resource can be any property of the resource that can be accessed. If the resource for instance is a hard disk, a possible property is the number of sectors.

Resource Property

Definition

A resource property is a data element of the WS-Resource which can be read and potentially set by the web service.

Relationships

- A resource property maps to a Property of the resource

Explanation

A resource property is a data element that represents or is mapped to the property of a resource.

Properties are read and written by a client using operations of the web service. Only the web service accesses the properties.

Resource Properties Definitions

Definition

A Resource Properties Definition describes the resource properties, this means the data type of the resource properties.

Relationships

- A resource property description describes a resource property

Explanation

The resource property definition defines the data types used for the representation of the resource properties. This information is available in the WSDL document.

Web Service

Definition

In addition to what has been described in section 4.2.1, a web service being part of a WS-Resource has to provide a set of operations defined by the WS-Notification, WS-ResourceProperties, WS-ResourceLifetime, WS-Addressing standards.

Relationships

- A web service defines a WS-Resource
- Is described by a WSDL
- Is a Service
- Refers to a Resource Factory

Explanation

The web service part of the WS-Resource has to provide a set of operations as defined in the WS-Notification, WS-ResourceProperties, WS-ResourceLifetime, WS-Renewable References, WS-ServiceGroup, WS-BaseFaults. These operations can be used to manage the lifetime of a WS-Resource, the properties and notifications.

The standards defining these operations have been presented in section 2.3.1.

WSDL

Definition

In addition to what has been described about the WSDL in the section 4.2.1, a WSDL in this case also contains the description of the Resource Properties.

Relationships

- WSDL contains the descriptions of the resource properties

Explanation

Within this context a WSDL document additionally contains descriptions of the resource properties which can be accessed using the operations defined in WS-ResourceProperties.

Resource Factory

Definition

A ResourceFactory is used to instantiate a WS-Resource much like the factory pattern is used in Object Oriented Programming (OOP).

Relationships

- A resource factory refers to a web service
- A resource factory instantiates a resource

Explanation

The Factory design pattern is commonly used in Objected-Oriented software systems to enable the creation of multiple, similar artifacts. A WSRF Resource Factory is a Web service operation that is used by a client to create other WS-Resource instances. When a client needs to create a new instance of a particular WS-Resource it locates the corresponding web service and invokes the corresponding Resource Factory. The client gets as an outcome an endpoint reference that can be used to access the newly created WS-Resource instance.

Notification

Definition

A Notification is a message sent from the web service to the client. Notifications are triggered by events happening at the WS-Resource.

Relationships

- Is a Message
- A resource triggers notifications

Explanation

Notifications are used to notify the client of events happening at the WS-Resource. A client can subscribe to specific events such as a property change. The notion of notification has been significantly broadened in WSRF compared to OGSI. While in OGSI it was merely used to inform a client about property changes, with WSRF it is possible to use this pattern to implement additional functionality. Notifications are mostly used to implement asynchronous service calls where a client invokes a service and receives a notification upon the completed execution asynchronously. The client can access possible intermediate results by reading the resource properties.

The client uses an unsubscribe operation as soon as it does not want to receive any further notifications.

LifeTime

Definition

The LifeTime defines the time left until destruction of the resource. It does not need to be defined.

Relationships

- Every resource has a lifetime

Explanation

If the lifetime of the WS-Resource instance is set, then destruction will happen implicitly after expiration of the time set in LifeTime. This means that the instance does not need to be explicitly destroyed. The LifeTime can be read by the client and may also be set by it.

The client also has the option of extending the LifeTime if this operation is supported by the WS-Resource.

EndpointReference

Definition

The EndpointReference uniquely identifies a resource instance. It is created upon resource instantiation.

Relationships

- Is a URI
- Every resource has a unique endpoint reference

Explanation

The EndpointReference uniquely identifies a resource instances by the means of an URI. This URI is returned to the client upon creation of the resource instance. In case the WS-Resource is only to be instantiated once (equivalent to the singleton pattern in OO) then the operation in charge of creating the instance will always return the same EndpointReference.

The EndpointReference is used by the client in all subsequent communication with the WS-Resouce. To do so, the EndpointReference is included in the header of each SOAP message in order to uniquely identify the instance the client wants to use.

4.2.3 P2P Service Type

Introduction

As it has been appointed in section 2.5 the platform that will be used by this report as well as the SODIUM project for the provision of p2p services is JXTA. Yet, the established elements and their respective associations may easily satisfy the needs of other types of p2p platforms and/or architectures as well.

The incorporation of the JXTA model though, isn't a straightforward process. Due to its lenient approach and the freedom that a developer is provided with many of the specified concepts and components can be easily bypassed. Such an example is the communication mechanisms that may be used for the interaction among two endpoints. Although JXTA describes a communication mechanism based on the use of the pipe element, it doesn't strictly enforce its use. Rather than that, developers are able to use any kind of communication mechanism and infrastructure they like.

This open approach and the freedom that JXTA provides to its users render the specification of a service model and the provision of supporting tools a complicated task. Therefore, as far as GeSMO is concerned we use the invocation approach which is based on pipes for the exchange of messages among services and their consumers.

This decision nonetheless, doesn't hinder the applicability or the extensibility of the p2p service type. Extensions may be provided so as to accommodate additional communication mechanisms or schemes.

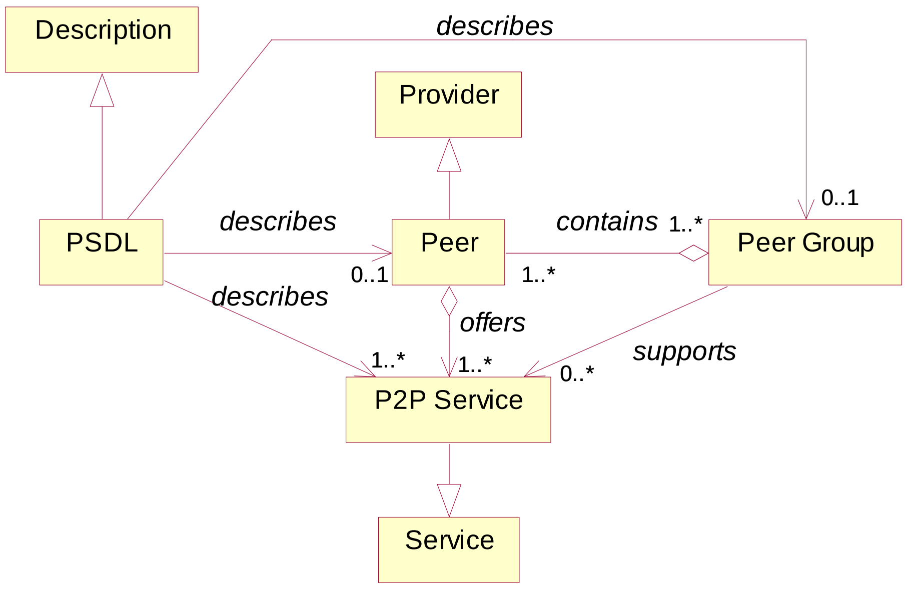

Figure 32: P2P Service description model

Peer

Definition

A peer represents a node of a p2p network, which is able to communicate with other peers and provide them with services (see Figure 32).

Relationships

- A peer is a service provider
- A peer may participate in more than one peer groups
- A psdl description describes a peer

Explanation

A peer represents a node of a p2p network that can communicate and provide services to the other peers of a p2p network. Every peer in a p2p network may invoke services that are provided by the peers. Furthermore peers may participate in more than one peer groups, which specify the set of services that all of its members should provide.

Peer Group

Definition

A peer group represents the logical group of peers that may be formulated in a p2p network.

Relationships

- A peer group consists of peers

- A peer group may support a set of services
- A psdl may describe a peer group

Explanation

A peer group represents the logical group of peers that may be formulated in a p2p network. A peer may participate in more than one peer groups, which specify a set of services that all member peers should provide.

A peer group may serve as a boundary specification for many aspects, e.g. logical, security, trust, etc. Thus it may act as a boundary for the provision or use of a service. E.g. all peers within a group should provide or are able to use a specific service.

P2P Service

Definition

A P2P Service represents a service that is provided by a peer or a set of peers in a p2p network.

Relationships

- A p2p service is a service
- A peer offers p2p services
- A peer group may support a set of services
- A psdl document describes a p2p service

Explanation

A p2p service represents a service that is provided by a peer or a set of peers in a p2p network. The level of granularity of a p2p service may vary from a coarse-grain to a fine-grain level.

A p2p service may be described in a psdl document, which facilitates the specification of its operations and details on how to invoke it. A client, which should be another peer of the p2p network, may use the psdl description so as to find out what a service does and how to invoke it.

PSDL

Definition

A psdl document describes the necessary features of a p2p service that enable a p2p service client to find out what the service does and how to invoke it.

Relationships

- A psdl is a description document
- A psdl describes a p2p service

- A psdl describes the peer which provides a service
- A psdl describes the pipes that may be used for message exchange
- A psdl specifies the pipe advertisements of the pipes that are used for message exchange

Explanation

A psdl provides a description on what a p2p service does and how it can be invoked. A psdl document is a specialized description that extends the existing p2p service advertisements.

The psdl descriptions do not aim to replace the existing service descriptions, but rather to annotate them with concepts of the core GeSMO model that they lack. The service model that is supported by the JXTA framework suppresses concepts such as those of the service interface or operation. These concepts however are crucial concepts that are supported by all other types of services.

Furthermore, psdl descriptions aim to support the development of enhanced and more automated service binding and invocation tools. Existing binding and invocation mechanisms that are supported by the JXTA framework are heavily dependent on human intervention, since a user is required to identify the proper mechanism (proxy) for binding to and invoking a p2p service [JXTA].

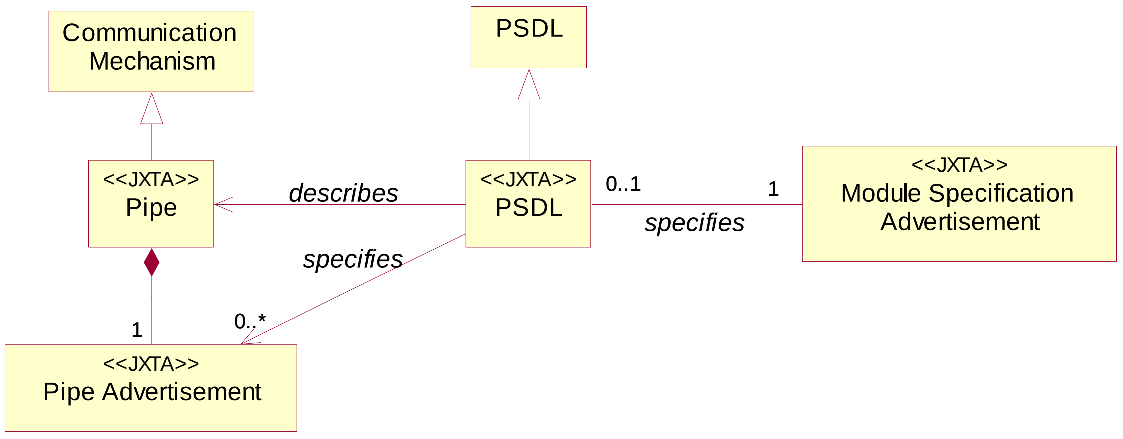

Figure 33: P2P Service and JXTA associations

Pipe

Definition

A pipe is a communication mechanism which is specified by the JXTA platform and it's used for handling the communication among two or more endpoint in a p2p network (see Figure 33, Figure 15).

Relationships

- A pipe is a communication mechanism
- Every pipe has a pipe advertisement

- A pipe is described in psdl description

Explanation

A pipe represents a communication mechanism of the JXTA platform that is used for exchanging messages among a set of endpoints within a JXTA network. According to JXTA [JXTA] a pipe enables the un- or encrypted, uni- or bi-directional communication among a set of endpoints within a p2p network. Every pipe within a JXTA p2p network must have a respective pipe advertisement which provides all necessary information for the identification and use of the pipe mechanism.

Within the SODIUM project a p2p service must use the pipe mechanism for exchanging messages with its respective clients. Furthermore, the psdl description document that has been introduced for the extended description of a p2p service will provide references to the pipe advertisements of the service's pipes.

Pipe Advertisement

Definition

A pipe advertisement is a JXTA construct which provides a description for a pipe and it's used for the identification and instantiation of a pipe among a set of endpoints (see Figure 16).

Relationships

- Every pipe has a corresponding pipe advertisement
- A psdl document specifies a set of pipe advertisements

Explanation

A pipe advertisement is a description document that provides for publishing, identification and instantiation of a pipe within a JXTA network, among a set of communicating endpoints. According to the JXTA specification, every pipe within a p2p network has a pipe advertisement, which is used by the underlying infrastructure for the establishment of a communication channel among a set of endpoints.

Module Specification Advertisement

Definition

A module specification advertisement is a JXTA established document that provides for the description of p2p service (see Figure 16).

Relationships

- A module specification advertisement is referred by a psdl description

Explanation

A module specification advertisement is a document that is specified by the JXTA platform and it's used for describing and publishing a service or a module. A module specification advertisement describes the service’s behavior and may also specify how the service can be invoked. Every service must have a module specification advertisement.

The psdl description of a p2p service must point to the module specification advertisement of that service.

4.3 Conformance Requirements

All languages within the SODIUM project as it has been specified beforehand will be based on GeSMO. Thus, it is required that all languages should conform to the specified concepts and models. Considering that the aforementioned specifications are high-level models which describe the concepts that are supported by the addressed service types, this is not difficult to achieve.

Therefore, the conformance criterion that all SODIUM languages should comply with is to at least support the aforementioned concepts and their defined relationships. Moreover, all languages are free to provide appropriate extensions as needed and additional associations that are missing, as long as these extensions don’t suppress or hinder the specified concepts and their respective features.

5. CONCLUSIONS

The objective of the work presented in this report was to provide a generic service model (GeSMO) that would introduce a common set of features that are shared among the web, grid and p2p service types along with their distinct properties that are incorporated by each service type. This generic service model would serve as a point of reference for the SODIUM project's languages i.e. VSCL, USQL and USCL by specifying the concepts that are shared among these languages.

In order to construct this generic service model we have followed a process which consists of five steps. These steps are:

- Establish a set of requirements for GeSMO
- Investigate the current state of the art in service oriented technologies
- Identify similarities and discrepancies among the investigated service models
- Specify a structure for GeSMO and describe its elements
- Assess the model's correctness through a paradigm

The first step according to this process was to established a set of high level requirements that would serve as a placeholder for the provided model. These requirements embellish GeSMO with added value and leverage its acceptance.

The following step, according to the process, was to investigate contemporary conceptual models, standards, protocols and frameworks that have been used by each of the addressed service types. The investigated standards and proposals address all aspects of a service's lifecycle ranging from description, discovery, invocation, composition and management to some common orthogonal aspects such as specification of semantics and quality of service. For each of the examined protocols and standards we have constructed a model with its basic concepts and we have provided a brief description of its properties.

Consequently, we have selected the protocols, standards, platforms and/or architectures which would serve as the basis for the specification of GeSMO and that would be tackled by the SODIUM project. Specifically, regarding grid services it has been appointed that we will defer to the WSRF specification whereas for p2p services it has been appointed that the platform which will be used is JXTA.

Based on the provided state of the art analysis and our decisions related to the selected protocols and standards, we have pinpointed the similarities and differences among the investigated types of services. The outcome of this comparison was that despite their intrinsic similarities, the addressed service types have a set of differences that are scattered across all levels of abstractions and all aspects of a service model.

The identified similarities and differences helped us in defining the structure of GeSMO. The layered structure has been appointed as the most appropriate since it enables the specification of a common set of concepts that are incorporated by each of the addressed service types and the construction of additional layers, on top of the common layer, for the provision of necessary extensions that are needed by each service type.

For the specification of each layer a modular approach has been incorporated. This approach facilitates the grouping of concepts and associations that are addressing specific aspects of a service or different levels of abstractions. E.g. abstract model, basic model, semantic model, etc.

Our investigation has led us to the definition of a service according to which a service may be conceived as i) a software system that i) has some functional and non-functional properties, ii) it exposes a certain behavior, iii) it resides at a network address, iv) it is described by a description document and v) it is accessible to its clients via messages.

As far as our model is concerned we have also identified some other crucial elements that need to be supported by each language or framework that will be based upon GeSMO. According to our model a service should have at least one interface which consists of a set of operations. These operations describe the messages that are exchanged among a service and its client when a specific functionality is invoked.

The set of concepts that have been established in the core layer have been extended with additional elements and properties by each of the specific service types i.e. web, grid and p2p services. Considering web services, the provision of the missing elements was a straightforward process, since most of the necessary constructs have been already described in the core layer. As it has also been illustrated the concepts that have been provided for grid services – which are based on the WSRF specification- have extended the concepts that were introduced for web services.

With respect to p2p services, the platform that we used for their specification is JXTA. Nonetheless, the elements that are provided for this service type may support the specification of p2p services of other platforms as well. This is with the provision that additional elements related to the communication mechanism of the selected p2p platform will be provided as extensions.

The validity of our model has been assessed with the help of a scenario that was based on the requirements of the Medisystem's pilot application, which are described in [VREDSZ05]. The presented scenario was used as input for the identification of web, grid and p2p services that are needed for its realization. Examples with the xml representation of each service type as well as the USQL queries and responses were provided. All these paradigms exemplify how the GeSMO has been applied in the specification of the USQL language and in the specification of the service types that are leveraged by USCL for the invocation of services.

Concluding we should state again that the aim of this report was not to provide a service description language, but rather to model a set of concepts that are needed and shared by each of the addressed service types i.e. web, grid and p2p services. This generic service model will serve as a basis for the provision of the SODIUM project's languages, which are free to provide additional extensions as long as these don't obscure or tamper the existing elements or their specified properties.

6. REFERENCES

[w3c 2004] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, *Web Services Architecture*, W3C Working Group Note, Feb 2004, http://www.w3c.org/ws-arch/

[CKMTW 2003] Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S., *The Next Step In Web Services*, Communications of the ACM, vol. 46 No. 10, October 2003

[GKS 2002] A. Gokhale, B. Kumar, A. Sahuguet, *Reinventing the Wheel? CORBA vs. Web Services*, Proc. of the WWW2002, May 2002

[CORBA] OMG, Common Object Request Broker Architecture (CORBA), http://www.corba.org

[Szyp 2003] Szyperski, C., *Component Technology-What, Where and How?*, Proc. of the 25th International conference on Software engineering, May 2003

[AlCaKuMa 2003] Alonso, G., Casati, F., Kuno, H., Machiraju V., *Web Services: Concepts, Architectures and Applications*, Springer Verlag

[XLANG] Satish Thatte. *Xlang.* Technical report, 2001. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.html.

[WSFL] Frank Leymann. *Web services flow language (wsfl 1.0)*. Technical report, 2001. http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[XMLSchema] W3C, *XML Schema Part 1: Structures Second Edition*, W3C Recommendation, Oct 2004, http://www.w3.org/TR/xmlschema-1/

[XPATH] W3C, *XML Path Language (XPath)Version 1.0*, W3C Recommendation, Nov 1999, http://www.w3.org/TR/xpath

[BPMI 2002] BPMI, BPML: *Business Process Modeling Language 1.0 (2002)*, http://bpmi.org/bpml-spec.esp

[ebXML 2003] OASIS and UN/CEFACT, *Electronic Business XML (ebXML)*, http://www.ebxml.org

[ebMS] OASIS, "*Message Service Specification Version 2.0*", OASIS ebXML Messaging Services Technical Committee, OASIS Standard, 1 April 2002, http://www.ebxml.org/specs/ebMS2.pdf

[Edutella] Edutella project, http://edutella.jxta.org

[WSR] OASIS, *Web Services Reliability (WS-Reliability) version 1.1*, OASIS Web Services Reliable Messaging TC, Committee Draft 1.086, 24 August 2004, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm

[CFFFGMST 2004] Czajkowski K., Ferguson D., Foster I., Frey J., Graham S., Maguire T., Snelling D., Tuecke S. *From Open Grid Services Infrastructure to WSResource Framework: Refactoring & Evolution, Version 1.0*, Whitepaper, http://www.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf February 2004.

[OGSI] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, P. Vanderbilt, *Open Grid Services Infrastructure (OGSI). Version 1.0*, Global Grid Forum, June 2003

[Czaj 2004] Czajkowski K., et al *The WS-Resource Framework version 1.0*, Whitepaper, http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf, March 2004

[PWWR 2003] Parastatidis S., Webber J., Watson P., Rischbeck T. *A Grid Application Framework based on Web Services Specifications and Practices, version 1.0,* Whitepaper, http://www.neresc.ac.uk/projects/gaf, September 2003

[FKNT 2002] Foster, I., Kesselman, C., Nick, J., Tuecke, S., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Globus Project, 2002, Available at http://www.globus.org/research/papers/ogsa.pdf

[FKT 2001] Foster, I., Kesselman, C., Tuecke, S., *The Anatomy of the Grid, Enabling Scalable Virtual Organizations*, Supercomputer Applications, 2001

[GWD-I] Open Grid Services Architecture, http://forge.gridforum.org/projects/ogsa-wg

[OGSA] I. Foster, H. Kishimoto, *Open Grid Services Architecture*, version 1.0, May 2004, http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-ogsa-spec.pdf

[OGSI] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, P. Vanderbilt, *Open Grid Services Infrastructure*, Feb 2003, http://www.gridforum.org/Meetings/ggf7/drafts/draft-ggf-ogsi-gridservice-23_2003-02-17.pdf

[GHMS 2003] Gerke, J., Hausheer, D., Mischke, J., Stiller, B., *An Architecture for a Service Oriented Peer-to-Peer System (SOPPS)*, in Praxis der Informationsverarbeitung und Kommunikation (PIK) 2/03, p.90-95, April 2003, http://www.mmapps.org/papers/sopps.pdf

[SGNP] Joshua Apgar, Andrew Grimshaw, Steven Harris, Marty Humphrey, Anh Nguyen-Tuong, *Secure Grid Naming Protocol (SGNP)* Draft Specification, February, 2002

[SNMP] IETF, A Simple Network Management Protocol (SNMP), RFC1157, http://www.ietf.org/rfc/rfc1157.txt

[SNIA] Storage Networking Industry Association, http://www.snia.org/tech_activities/SMI/cim/

[JMX] SUN, Java Management Extensions, http://java.sun.com/products/JavaManagement/

[Kerberos] B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks, IEEE Communications, 32(9):33-38. Sep. 1994

[PKI] IETF, Public-Key Infrastructure (X.509), http://www.ietf.org/html.charters/pkix-charter.html

[GKS 2002] A. Gokhale, B. Kumar, A. Sahuguet, *Reinventing the Wheel? CORBA vs. Web Services*, Proc. of the WWW2002, May 2002

[Gnutella], Gnutella, www.gnutella.com

[GPA WG] GGF Grid Protocol Architecture Working Group, http://www-itg.lbl.gov/GPA/

[JXTA] Project JXTA, http://www.jxta.org/

[JXTAv2 2003] Sun Microsystems, *Project JXTA v2.0: Java Programmer's Guide*, May 2003, http://www.jxta.org/docs/JxtaProgGuide_v2.pdf

[Kaler 2002] Kaler, C., *Web Services Security (WS-Security)*, Version 1.0, http://www-106.ibm.com/developerworks/library/ws-secure

[XSign] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, E. Simon, "*XML Signature Syntax and Processing*,", W3C Proposed Recommendation, August 2001, http://www.w3.org/TR/2001/PR-xmldsig-core-20010820

[XEnc] T. Imamura B. Dillaway, E. Simon, "*XML Encryption Syntax and Processing*," W3C Working Draft , Mar 2002, http://www.w3.org/TR/xmlenc-core/

[OASIS] OASIS, http://www.oasis-open.org

[OWL 2004] W3C, *OWL Web Ontology Language Overview*, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-features/

[RDF 2004] W3C, RDF Primer, W3C Recommendation, Feb, 2004, http://www.w3.org/TR/rdf-primer/

[SOAP] SOAP, http://www.w3.org/TR/SOAP

[SOAP 1.1] D Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, d. Winer, *Simple Object Access Protocol (SOAP) 1.1*, W3C Note, May 2000

[SOAP 1.2] Martin Gudgin, Marc Hadley, Jean-Jacques Moreau, Henrik Frystyk Nielsen, *SOAP 1.2 Part 1: Messaging Framework*, W3C Candidate Recommendation (work in progress), 20 December 2002, http://www.w3.org/TR/2002/CR-soap12-part1-20021219/

[Szyp 2003] Szyperski, C., *Component Technology-What, Where and How?*, Proc. of the 25th International conference on Software engineering, May 2003

[TCFFGK 2002] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., *Grid Service Specification*, Available at: http://www.gridforum.org/ogsi-wg/drafts/GS_Spec_draft02_2002-06-13.pdf, Last Access: August 16th, 2004

[BPEL4WS 2003] Thatte, S., *Business Process Execution Language for Web Services version 1.1*, http://dev2dev.bea.com/techtrack/BPEL4WS.jsp

[UDDI] OASIS, UDDI -Technical Committee, http://www.uddi.org/

[UDDI v2 DS] OASIS, *UDDI Version 2.03 Data Structure Reference*, UDDI Committee Specification, July 2002, http://uddi.org/pubs/DataStructure_v2.htm

[W3C 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D., *Web Services Architecture*, W3C Working Group Note, Feb 2004, http://www.w3c.org/ws-arch/

[W3C] World Wide Web Consortium, http://www.w3.org/

[WSA 2004] Web Services Architecture, W3C Working Group Note 11 February 2004

[WS-Addressing] IBM, *WS-Addressing, an XML serialization and standard SOAP binding for representing network wide “pointers” to services*, http://www.ibm.com/developerworks/webservices/library/ws-add/

[WSAO] IBM, *Web Services architecture overview: The next stage of evolution for e-business*, published on 01.09.2000, http://www-106.ibm.com/developerworks/library/w-ovr/?dwzone=ws

[WSCI 2002] BEA Systems, Intalio, SAP, Sun Microsystems, *Web Service Choreography Interface (WSCI) 1.0*, http://www.w3.org/TR/wsci

[WSDL 2001] W3C, *Web Services Description Language (WSDL) 1.1*. Note, W3C, http://www.w3.org/TR/wsdl

[WS-I] Web Services Interoperability Organization, http://www.ws-i.org/

[WS-I BP 1.1] Keith Ballinger, David Ehnebuske, Christofer Ferris, Martin Gudgin, Mark Nottingham, Prasad Yendluri, eds. *Basic Profile Version 1.1*, WS-I specification, 8 August 2004, http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-07-21.html

[WSRF] Globus Org, The WS-Resource Framework, http://www.globus.org/wsrf/

[WS-ServiceGroup] Tom Maguire, Jeffrey Frey, Nataraj Nagaratnam, Igor Sedukhin, David Snelling, Karl Czajkowski, Steve Tuecke, William Vambenepe, *Web Services Service Group – Specification (WS-ServiceGroup),* version 1.0, Mar 2004, http://www.ibm.com/developerworks/library/ws-resource/ws-servicegroup.pdf

[WS-ResourceLifetime] Jeffrey Frey, Steve Graham, Karl Czajkowski, Donald F Ferguson, Ian Foster, Frank Leymann, Tom Maguire, Nataraj Nagaratnam, Martin Nally, Tony Storey, Igor Sedukhin, David Snelling, Steve Tuecke, William Vambenepe, Sanjiva Weerawarana, *Web Services Resource Lifetime (WS-ResourceLifetime),* version 1.1, May 2004, http://www-106.ibm.com/developerworks/library/wsresource/ws-resourcelifetime.pdf

[WS-ResourceProperties] Steve Graham, Karl Czajkowski, Donald F Ferguson, Ian Foster, Jeffrey Frey, Frank Leymann, Tom Maguire, Nataraj Nagaratnam, Martin Nally, Tony Storey, Igor Sedukhin, David Snelling, Steve Tuecke, William Vambenepe, Sanjiva Weerawarana, *Web Services Resource Properties (WS-ResourceProperties),* version 1.1, Mar 2003, http://www-106.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf

[WS-Base Notification] Steve Graham, Peter Niblett, Dave Chappell, Amy Lewis, Nataraj Nagaratnam, Jay Parikh, Sanjay Patil, Shivajee Samdarshi, Igor Sedukhin, David Snelling, Steve Tuecke, William Vambenepe, Bill Weihl, *Web Services Base Notification (WS-Base Notification),* version 1.0, Mar 2004, ftp://www6.software.ibm.com/software/developer/library/ws-notification/WSBaseN.pdf

[WS-BrokeredNotification] Steve Graham, Peter Niblett, Dave Chappell, Amy Lewis, Nataraj Nagaratnam, Jay Parikh, Sanjay Patil, Shivajee Samdarshi, Igor Sedukhin, David Snelling, Steve Tuecke, William Vambenepe, Bill Weihl, *Web Services Brokered Notification (WS-BrokeredNotification),* version 1.0, Mar 2004, http://www-106.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf

[WS-BaseFaults] Steve Tuecke, Karl Czajkowski, Jeffrey Frey, Ian Foster, Steve Graham, Tom Maguire, Igor Sedukhin, David Snelling, William Vambenepe, *Web Services Base Faults (WS-BaseFaults),* version 1.0, Mar 2004, http://www-106.ibm.com/developerworks/library/ws-resource/ws-basefaults.pdf

[Vogels 2003] Werner Vogels, *Web Services Are Not Distributed Objects*, IEEE Internet Computing, Nov.-Dec. 2003

[RFC2616] IETF, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, R. Fielding et al, IESG and IETF, June 1999, http://www/ietf.org/rfc/rfc2616.txt

[WSSe]OASIS Service Security Technical Committee, *OASIS Web Services Security TC*, http://www.oasis-open.org/committees/wss/

[WSDL-S] *WSDL-S: Web Service Semantics - WSDL-S* Technical Note, http://lsdis.cd.uga.edu/Projects/METEOR-S/WSDL-S Version 1.0, April, 2005

[OWLS 2004] W3C, *OWL-S: Semantic Markup for Web Services*, W3C submission, Nov. 2004 http://www.w3.org/Submission/OWL-S/

[OWL-S/DAML-S] OWL-S Home Page http://www.daml.org/services/owl-s/, 2003.

[HPBTG03] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *Swrl: A semantic web rule language combining owl and ruleml*, 2003. Available at http://www.daml.org/2003/11/swrl/.

[KlCa04] G. Klyne and J. J. Carroll. *Resource Description Framework (RDF): concepts and abstract syntax*, 2004. W3C Recommendation. Available at http://www.w3.org/TR/2004/REC-rdf-concepts-20040210.

[KIF98] *KIF. Knowledge Interchange Format: Draft proposed American National Standard* (dpans). Technical Report 2/98-004, ANS, 1998. Also at http://logic.stanford.edu/kif/dpans.html.

[PDDL98] M. Ghallab et al. *PDDL-The Planning Domain Definition Language V. 2. Technical Report*, report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998

[WSMO] Web Service Modeling Ontology: http://www.wsmo.org/TR/d2/v1.1/

[FenBus02] D. Fensel and C. Bussler: *The Web Service Modeling Framework WSMF*, Electronic Commerce Research and Applications, 1(2), 2002

[WSMOD14] WSMO, D14 v0.2. Ontology-based Choreography and Orchestration of WSMO Services, http://www.wsmo.org/TR/d14/v0.2/

[WSMOD15] WSMO, *D15v0.1. Orchestration in WSMO*, http://www.wsmo.org/TR/d16/d16.1/v0.2/

[WeKuLa98] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf: RFC 2413 - Dublin Core Metadata for Resource Discovery, September 1998.

[WSMOD24] WSMO, *D24.2v0.1. WSMO Grounding*, http://wsmo.org/TR/d24/d24.2/v0.1/

[CGMSW] R. Chinnici, M. Gudgin, J-J. Moreau, J. Schlimmer and S. Weerawarana (editors), *Web Services Description Language Part 1: Core Language*, W3C Last Call Working Draft available at http://www.w3.org/TR/wsdl20/

[WSML] J. de Bruijn (Ed.), *The WSML Family of Representation Languages*, WSMO Deliverable D16, DERI Working Draft, 2004, latest version available at http://www.wsmo.org/2004/d16/.

[BerHenLa01] T. Berners-Lee, J. Hendler, O. Lassila, *The Semantic Web*, Scientific America 284(5), pp 34-43, May 2001

[XMLSchDT] W3C, *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation, October 2004

[G-QoSM 02] R. J. Al-Ali, O. F. Rana, D. W. Walker, S. Jha, and S. Sohail, G-QoSM: Grid Service Discovery Using QoS Properties, Computing and Informatics, vol.21, pages 363-382, 2002. ISSN 1335-9150

[WS-QoS] Web Service QoS, http://www.wsqos.net/

[WSLA] Web Service Level Agreements (WSLA) Project, http://www.research.ibm.com/wsla/

[WSLA v1.0] H. Ludwig, A. Keller, A. Dan, R. P. King, R. Franck, *Web Service Level Agreement (WSLA) Language Specification*, IBM, version 1.0 , wsla-2003/01/28

[WSOL] Vladimir Tosic, Bernard Pagurek, Kruti Patel, Babak Esfandiari, Wei Ma, *"Management Applications of the Web Service Offerings Language (WSOL)"*, Proc. of CAiSE'03, Velden, Austria, June 16-20, 2003, Springer-Verlag, Lecture Notes in Computer Science (LNCS), No.2681, pp. 468-484, 2003

[BHATBVKL04] Arne J. Berre, Alex Hahn, David Akehurst, Aphrodite Tsalgatidou, Jean Bezivin, Francois Vermaut, Lea Kutvonen, Peter F. Linington, *D9.1: State-of-the art for Interoperability architecture approaches*, Interop project IST-508011 deliverable, Nov 2004

[BhSnCh] P. Bhoj, S. Snighal and Chutani, *SLA Management in Federated Environments*, Technical Report in HPL-98-203, HP Labs, Palo-Alto,CA 94034, USA

[FoKesNahRoy] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, A. Roy, *A Distributed Resource Management Architecture that Supports Advance, Reservation and Co-Allocation*, Proceedings of the Instational Workshop on QoS, pp27-39

[VREDSZ05] George Valais, Radu Dop, Mugur Enache, Cristina Dima, Dan Stanescu, Zaharia Ana-Maria, *D5 – Specification of requirements for User Applications Part II: Requirements specification for the MEDISYSTEM Pilot*, SODIUM project IST-FP6-004559 deliverable, June 2005

[USQL] Aphrodite Tsalgatidou, Michael Pantazoglou, George Athanasopoulos, *D8-Specification of the Unified Service Query Language (USQL)*, SODIUM project IST-FP6-004559 deliverable, June 2005

[USCL] Cesare Pautasso, Thomas Heinis, Gustavo Alonso, *D6-SODIUM Unified Service Composition Language (USCL)*, SODIUM project IST-FP6-004559 deliverable, June 2005

[VSCL] Hjørdis Hoff, Roy Grønmo, David Skogan, Audun Strand, *D7-SODIUM Visual Service Composition Language(VSCL)*, SODIUM project IST-FP6-004559 deliverable, June 2005

[SEKT] Semantically-Enabled Knowledge Technologies –SEKT: http://www.sekt-project.com/

[DIP] Data, Information and Process Integration with Semantic Web Services – DIP: http://dip.semanticweb.org/

[KW] Knowledge Web – KW: http://knowledgeweb.semanticweb.org/

[GWBBCBCG01] S. Gribble, M. Welsh, R. von Behren, E. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R. Katz, Z. Mao, S. Ross, and B. Zhao. The Ninja Architecture for Robust Internet-Scale Systems and Services. Computer Networks, Special Issue on Pervasive Computing, 2001

[XuNa 02] D. Xu and K. Nahrstedt. Finding Service Paths in a Media Service Proxy Network. Proc. of SPIE/ACM Multimedia Computing and Networking Conference (MMCN'02), San Jose, CA, January 2002

[GuNah 02] X. Gu, K. Nahrstedt, *A Scalable QoS-Aware Service Aggregation Model for Peer-to-Peer Computing Grids*, In Proc. of IEEE International Symposium on High Performance Distributed Computing (HPDC 2002), Edinburgh, Scotland.