Interoperability among Heterogeneous Services

George Athanasopoulos, Aphrodite Tsalgatidou, Michael Pantazoglou
Dept. of Informatics and Telecommunications,
National and Kapodistrian University of Athens
{gathanas, atsalga,michaelp}@di.uoa.gr

Abstract

Service-Oriented Computing (SOC) has been marked as the technology trend which caters for the interoperability among the components of a distributed system. However, the emergence of various incompatible instantiations of the SOC paradigm e.g. web, grid and p2p services, as well as the interoperability problems encountered within each of these instantiations (e.g. web service interoperability problems addressed by the WS-I Basic Profile) state clearly that interoperability is still elusive. In order to address this problem we first need to identify all problem dimensions and consequently to provide appropriate solutions. Within this paper we describe a set of interoperability dimensions that need to be considered and we present a generic service model which we view as a first step in addressing some of the identified problem dimensions.

1. Introduction

The development of open, large-scale, distributed systems has been always confronted with the problem of interoperability. Existing component-based technologies have tried to address this issue, but they haven't managed to provide a widespread solution that would enable the interoperation of diverse components developed by different providers, in multi-vendor platforms [1].

Service-Oriented Computing (SOC) has been branded as the technology trend whose primal objective is to facilitate interoperability between the components of a distributed system. The most well-known instantiations of the service-oriented computing paradigm are web [7] and grid services [15], but there are also other types such as p2p (Peer-to-Peer) services [16], which are currently gaining momentum. All these types of services are built on top of XML [20] and other proven communication protocols such as HTTP [19] and TCP/IP [21]. Researchers have been also articulating for quite a long time on a set of common concepts that are shared among all types of services. This set of common characteristics includes properties such as self-description, internet accessibility and message-oriented communication [14]. These features along with the use of XML [20] provide an infrastructure that leverages interoperability among the components of a service-oriented system.

Nevertheless, despite the original hype surrounding the service-oriented technology, interoperability still remains an open issue. Although existing service-oriented computing instantiations provide a basic infrastructure for tackling interoperability, they still do not fully address it. This is mainly due to the multidimensional nature of interoperability, as it has been noted by other researchers, too [2], [13]. Thus, although each of the service-oriented computing paradigm instantiations - e.g. web, grid and p2p services - provides a basic infrastructure for supporting interoperability, they fail to address all dimensions of the problem. Efforts such as those undertaken by WS-I [17], which has provided a basic interoperability profile for addressing some of the interoperability problems among web services, foreground the need for addressing the problem in various dimensions. Furthermore, this multiplicity of existing service types has further aggravated the problem, as, albeit these service types share some common characteristics, they adhere to incompatible models and standards and employ distinct platforms and middleware to perform their basic activities [8]. Therefore, the continuous proliferation of such heterogeneous services renders the support for their interoperation an important task.

The goal of this paper is twofold: on one hand it aims to elaborate on the problem of interoperability among heterogeneous types of services such as web, grid and p2p services and to view it from different dimensions. In this way, the issues that need to be addressed in order to facilitate the integration of heterogeneous services can be exemplified more clearly. On the other hand, the paper aims to take into account these issues and develop a generic service model that can leverage interoperability among web, grid and p2p services.

The rest of the paper is structured as follows. Section 2 elaborates on the problem of service interoperability and establishes a set of interoperability dimensions. Section 3 presents the interoperability dimensions that are involved when trying to integrate heterogeneous services. Section 4 briefly presents a generic service model which was established so as to address some of the interoperability concerns identified in section 3.

IEEE International Conference on Services Computing (SCC'06)
0-7695-2670-5/06 $20.00 © 2006 IEEE

Finally section 5 concludes with a short discussion on the open issues and our future work plans.

2. Interoperability Dimensions

Research on *service interoperability* has received considerable momentum within the last decade. According to the majority of the proposed interoperability models [2][5][6], service interoperability has been sub-divided into the *signature*, *protocol* and *semantic levels*[1]. An appropriate classification scheme that considers the evolution of distributed computing from the emergence of RPC or component models such as DCOM [28] and EJBs [27] to CORBA [26], has been proposed in [5] and is presented in Figure 1.

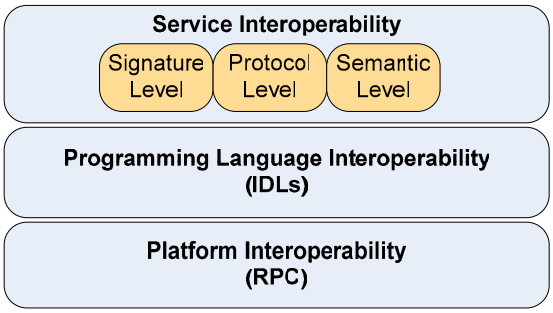

Figure 1: Classic Interoperability dimensions

In [3], Strang proposed to extend the service interoperability levels with the addition of the *context interoperability level*, which is of high importance to context-aware applications, whereas Ruiz in [4] proposed to extend it with the addition of the *quality interoperability level* (Figure 2).

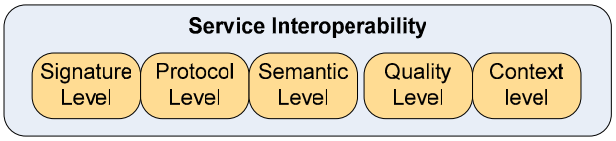

Figure 2: Extended set of Service Interoperability dimensions

Each of these levels describes specific interoperability concerns which need to be tackled when integrating two service-oriented systems. These concerns are briefly presented below:

- *Signature level*: This level addresses the interface definition conformance. This includes the operations, types and order of parameters of a service interface as well as standards such as the interface definition languages (IDL).
- *Protocol level*: Interoperability on the protocol level addresses the order in which the methods of a service are invoked. The Web Service Choreography Description Language (WS-CDL) [18] can be seen as an effort to resolve this interoperability problem.
- *Semantic level*: Semantic interoperability addresses the problem of common understanding between service providers and service requestors. This problem can be tackled through ontologies and semantic service description frameworks such as OWL-S [24], WSMO [25] or METEOR-S [23].
- *Quality level*: Quality of Service interoperability (QoS) addresses the conformance of the quality requirements of a service requestor and the quality properties offered by the service provider.
- *Context level*: Context level interoperability refers to the conformance between the context representations used by service providers and the context representations requested by the clients. Context level interoperability is important when dealing with service interoperability in ubiquitous computing environments.

Although the aforementioned service interoperability levels (or dimensions) are the most commonly used, they are not the only ones. One can consider other dimensions as the ones that we propose below (see Figure 3):

- *Business Domain*: Business Domain interoperability represents the ability of two business systems to interoperate. This includes the sharing of common domain concepts and processes.
- *Application*: Applications are instantiations of specific business domains. Thus, application interoperability represents the ability of two specific business system instantiations as perceived by their developers to interoperate. This includes the use of compatible data structures, functionality and orchestrations.
- *Platform*: Platform interoperability represents the ability of the underlying middleware leveraged by two applications to interoperate. This includes features such as the use of compatible data type representations (e.g. real numbers having the same accuracy and same format), interface specification mechanisms (e.g. Interface Definition Languages) or architectural styles (e.g. use of Message-Oriented communication styles). As it can be seen, this layer differs from the one presented in Figure 1 in that it addresses both programming language interoperability and the platform interoperability dimensions.

[1] In this paper, the terms *interoperability dimension* and *interoperability level* have the same meaning and are used interchangeably.

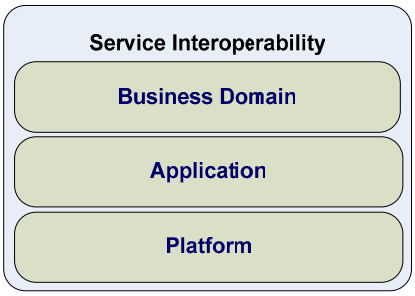

Figure 3: Additional Service Interoperability dimensions

Both classification frameworks (depicted in Figures 1 and 2) provide appropriate concepts for modeling service interoperability dimensions. Actually, these frameworks are orthogonal to each other since they have been derived from different points of view. Specifically, the first classification framework takes an internal look on the aspects that need to be considered when dealing with the integration of two systems, whereas the second one uses a system architect 'coarse' point of view for the identification of the various levels which are affected by the integration. Thus, these frameworks may be integrated into a single one which facilitates the classification of interoperability concerns from two distinct points of view. Figure 4 illustrates this integrated view of service interoperability dimensions along with the association between the concepts of the two individual frameworks.

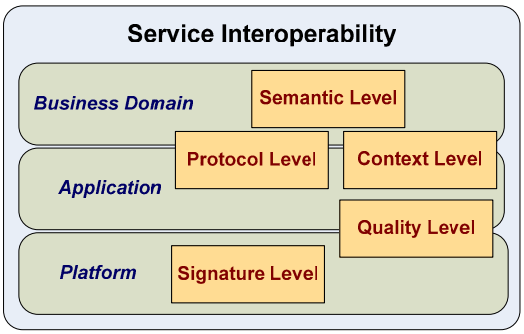

Figure 4: An integrated view of Service Interoperability Dimensions

As it can be seen in Figure 4, except from the semantic and signature interoperability levels (which are mapped to business domain and platform levels respectively), all the others don't have a one-to-one correspondence. Thus, interoperability issues referring to the protocol and context levels are associated with both the application and business levels; interoperability issues referring to quality level on the other hand are associated the application and platform levels. Let us consider for example the interoperability problem that may arise when integrating two systems that implement two incompatible processes (i.e. processes with incompatible choreographies), which is a protocol level issue. This problem may exist either due to the fact that the two systems support different business processes (in which case we have an interoperability problem at a business domain level) or because their developers have selected different algorithms for their implementation (in which case we have an interoperability problem at the application level).

In the following we present how the integrated framework depicted in Figure 4 can be used for the identification of the interoperability problems which arise when integrating heterogeneous services.

3. Interoperability Concerns for Heterogeneous Services

As it has been stated in [8], contemporary instantiations of the service-oriented computing paradigm, i.e. web, grid and p2p services, are ruled by different models, protocols and standards. The range of discrepancies and diversities among web, grid and p2p services spans across all service aspects such as description and discovery mechanisms, quality characteristics or service provision platforms. Based on the results of a thorough investigation on web, grid and p2p services that was undertaken for the purposes of the SODIUM project [10] we came up with a set of incompatibilities [9]. These incompatibilities are briefly presented below:

- *Supported Models*: There are two different models, namely the *stateful* service model and the *stateless* service model supported by these types of services. More specifically, web services are a proponent of the stateless service model, whereas grid services adhere to the stateful service model. P2P services on the other hand are leaning towards the stateful service model, though some stateless implementations also exist.
- *Intended Clients*: Although there might be specific security constraints dictating a different case, web services in general may be invoked by any client with internet access, provided that a client has the necessary infrastructure (e.g. a SOAP engine) to exchange messages (e.g. SOAP messages) with the service provider. When it comes to p2p services, the respective client should be either a member of the p2p network or it should have access to the network through another peer (proxy peer). As far as grid services are concerned, the client of a service needs to be provided with the necessary credentials so as to utilize resources of a specific virtual organization.
- *Syntactic Features*: The different models supported by the investigated types of services result in a set of syntactic diversities as well. Briefly, while grid and

web services adhere to the WSDL [7] defined service model (i.e. a service comprises a set of distinct operations) and utilize the SOAP message format, p2p services adhere to a suppressed service model where each service comprises a single operation, and employ proprietary message formats (see [16]). Furthermore, grid services have introduced the notion of *resource* along with other related concepts such as *resource lifetime* so as to support the stateful service model, whereas for the description of p2p services it is necessary to use concepts denoting the p2p network topology (e.g. peer, peer group, etc).

- *QoS Properties*: The origins of grid services are in high-performance scientific computing, whereas of web services in business-oriented systems. P2P services on the other hand are derived from collaborative information systems such as file sharing and instant messaging. Albeit there are certain implementations that do not adhere to the following claim, we may argue that the origins of each of these types of services have dictated specific quality properties for each of them respectively. Thus, grid services in general have to be reliable, secure and of high performance, whereas p2p services need not necessarily provide for all these properties.

All these differences among the investigated types of services end up to interoperability concerns that can be classified according to the previously identified interoperability dimensions. These concerns are summarized below:

- The difference on the supported models has an impact on the *signature, protocol* and *semantic* levels as well as on the *platform, application* and *business domain* dimensions of service interoperability. Specifically, as we have stated above, the use of the stateful service model by grid services resulted in the provision of additional concepts such as the *resource* or the *resource lifetime*. In addition, processes need to be provided so as to support the stateful service model, e.g. lifetime management or discovering and binding to specific resources. These modifications have an impact on the platforms that are utilized for the provision and invocation of services, on the steps that need to be followed for the integration of such services as well as on the semantics that are needed for the description of the additional elements and process steps.
- The difference on the intended clients has an impact on the *signature* and *protocol* levels as well as on the *platform* and *application* dimensions of service interoperability. This because in order to support the interoperability among services that are targeting different clients, the details of each platform need to be reconsidered. In addition, the different interaction patterns used by the clients and providers of such services have an impact on the process steps within an application that are followed for their interoperation.
- The difference on the syntactic features has an effect on the *signature* and *platform* dimensions of service interoperability. The use of incompatible structures and elements for the description of service syntactic characteristics has an effect on the languages used for their descriptions as well as on the middleware used for the provision and usage of such services. Therefore, in order to facilitate the integration of services with incompatible syntactic features, specific middleware needs to be provided so as to accommodate the mapping of the features of one service type to the features of another service type.
- The difference on QoS properties has an effect on the *quality* level as well as *platform* and *application* dimensions of service interoperability. Considering for example the issues of security, billing or availability, appropriate middleware and processes need to be used so as to cater for the provision, monitoring or management of such quality properties. Thus, the integration of services which are either demanding or providing incompatible quality characteristics has to be supported by appropriate middleware as well as by the use of specific process steps within an application.

As it can be easily seen, none of the identified discrepancies among the investigated types of services resulted in interoperability concerns for the context dimension. This is due to the fact that our investigation was focused on the interoperability problems that emerge when integrating heterogeneous services in general and not on services supporting context-aware applications.

Based on the results of our analysis, the interoperability dimensions that seem to be highly affected by the integration of heterogeneous services are the *signature, protocol, platform* and *application* ones. Thus, in order to facilitate the integration of heterogeneous services, special care should be given on these levels.

Several approaches could be used to accommodate the aforementioned interoperability problems. One is through the establishment of standards catering for the representation of aspects such as syntactic features, semantics or QoS. Another one is through the provision of appropriate middleware, languages and tools which will facilitate the interoperation of heterogeneous services. However, for such standards, middleware and languages to be developed we first need to establish a model with the concepts that each of the addressed types of services uses; this is actually what we do in the following section, where we present a generic model

that was developed by the authors in order to tackle this need.

4. Generic Service Model

In this section, we briefly present the structure and the concepts of a Generic Service MOdel, namely GeSMO. This model can be used as a basis for the development of appropriate languages and middleware that can address service interoperability. As we have mentioned before, GeSMO was based on a thorough investigation of the current state of the art on web, grid and p2p services. Specifically with respect to p2p services, GeSMO has been primarily influenced by the work in JXTA [16] as the latter is one of the very few p2p networks supporting the notion of service.

GeSMO was constructed in such a way that it efficiently models all common characteristics of web, p2p and grid services, while at the same time it provides for the modeling of the distinct characteristics per service type. In addition, the model has the following properties:

- *Generality*: The model is generic enough so as to support the modeling of all types of services and not just web, p2p and grid services.
- *Abstraction*: The model incorporates abstractions of all common concepts of the addressed types of services, which can be instantiated to the concepts supported by each specific type of service.
- *Extensibility*: The model can be easily extended with new features and properties, as well as with new service types.
- *Modularity*: The model is constructed in a modular manner, thus allowing the easy modification or extension of specific information parts.
- *Expressiveness*: The model is expressive enough to accommodate several service activities such as discovery and composition.
- *Simplicity*: The generic service model is simple enough to be easily used by a variety of users and tools which can be built on top of it.

The architecture that was selected for the development of the Generic Service Model (GeSMO) is a layered one consisting of the following layers (Figure 5):

- *a core layer* which models the concepts which are common to all investigated types of services;
- *an extension layer*, on top of the core layer, which provides for the distinct features of each service type; in the current version, this extension layer is divided into three modules, which model the distinct characteristics of each of the investigated types of services;
- *a number of layers* orthogonal to the core layer and its extensions, which model features related to semantics, quality, trust, security and management, as these features may be applied to all the concepts of the core layer and of its extensions.

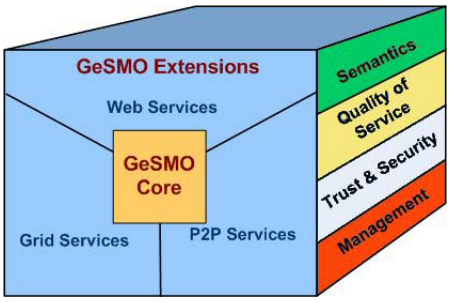

Figure 5: GeSMO's layered architecture

As expected, the fundamental element in GeSMO is the notion of service. The combination of concepts met in each of the aforementioned layers allows for modeling a service from multiple points of view. Figure 6 depicts the service concept along with some of the viewpoints that were used for its refinement.

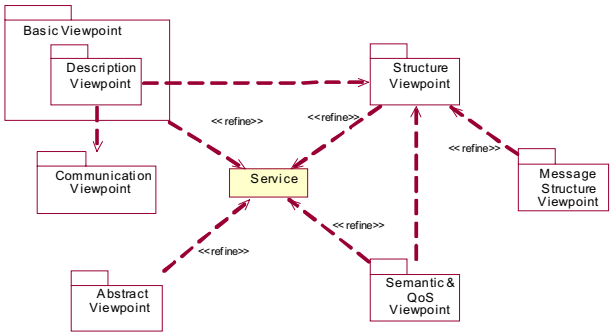

Figure 6: Generic Service Model Viewpoints

As we can see in Figure 6, among the viewpoints that were used for refining the concept of service are:

i) an abstract point of view
ii) a basic point of view
iii) a description point of view
iv) a structural point of view and
v) a semantic and a QoS view point

These views are exemplified in the following figures. Thus, from an abstract point of view (see Figure 7) a service is a software system which has a set of functional and non-functional characteristics and exhibits a certain behavior. Its behavioral, functional and non-functional properties can have a semantic interpretation, whilst some of its non-functional properties can be quantified as QoS properties.

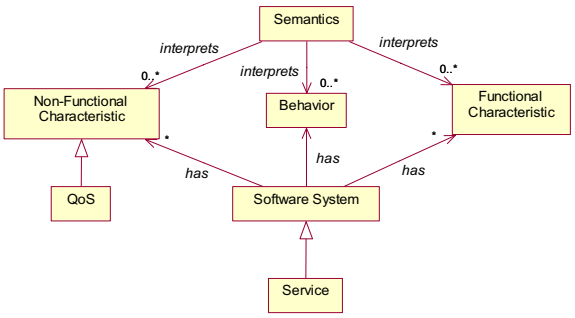

Figure 7: A Service from an abstract point of view

Moreover, services are regarded as self-described software systems, which interact with their clients over the Internet through messages. As we can see in Figure 8, a service description facilitates service clients in identifying the messages that can be exchanged as well as where and how these messages should be send.

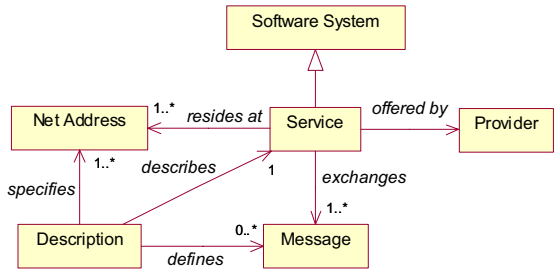

Figure 8: Basic Service Model

Specifically, the description of a service may convey information such as the one presented in Figure 9. As we see in this figure, a service description can convey information regarding its behavioral, semantic and quality features. In addition, a service description normally provides information about the communication mechanisms that may be used for accessing the service as well as its comprising syntactic elements.

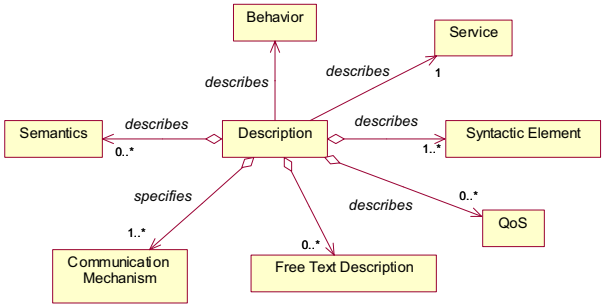

Figure 9: A Service's description point of view

The syntactic elements of a service along with their structure can be identified in a structural point of view (Figure 10). In this view, a service is associated with several elements organized in a certain way. Specifically, a service provides one or more interfaces which consist of the operations that this service offers to its clients. An operation groups a set of messages that are exchanged among a service and its respective clients. Each message consists of a set of elements which adhere to specific data types.

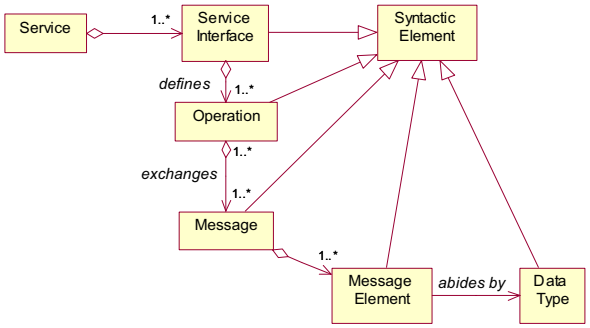

Figure 10: Structural point of view of a Service

A semantic and Quality-of-Service (QoS) point of view (Figure 11) illustrates which parts of a service can be semantically annotated, as well as which of them can be quantified and thus have specific QoS properties. As it can be seen in Figure 11, a service, its operations and the exchanged messages may have semantic interpretation. In addition, a service and its operations may also be associated with specific QoS properties.

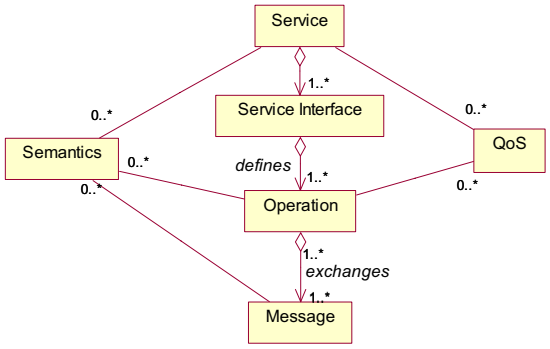

Figure 11: Semantic and QoS point of view for a service

The aforementioned concepts along with the additional set of concepts that are defined in [9] establish a set of elements that are shared by the investigated types of services, i.e. web, grid and p2p services. As it can be easily seen, most of these elements are primarily concerned with the syntactic features and concepts of a service. In other words, at its current state, the generic service model primarily addresses interoperability

concerns related to the signature and platform levels. However, work related to this model is still ongoing so as to address the rest of the identified interoperability dimensions.

We would like to note that the development priorities of GeSMO were in alignment with our need to promptly provide results that could be used in the SODIUM project [10]. More specifically, GeSMO served as a multi-purpose tool within SODIUM as follows. Firstly, it was used as a basis for the provision of necessary tools and languages that facilitate the unified discovery and composition of heterogeneous services [12]. Specifically, it was used as the basis for the development of:

- a visual service composition language that facilitates the modeling of workflow graphs comprising heterogeneous services [30],
- an XML-based service query language that facilitates the discovery of services over heterogeneous registries or networks [8] and
- an XML-based, service composition language that facilitates the orchestration of services [29].

The openness and extensibility of GeSMO guided the design and implementation of the above languages and their respective tools. Thus, each of the provided languages and tools leverages an extensible (pluggable) architecture which provides for the accommodation of additional properties that were not originally addressed.

Secondly, GeSMO facilitates the communication within the SODIUM project, in the sense that it supports the exchange of information not only among the tools of the SODIUM platform but also among the various project stakeholders.

Finally, GeSMO provided the basis for the development of a middleware along with a description language that facilitates the discovery and invocation of JXTA p2p services [11].

5. Conclusions and Future Work

The Service-Oriented Computing (SOC) paradigm has been advocated as the technology trend whose primary objective is to leverage the interoperability among the components of a service-oriented application. Despite the hype surrounding SOC as well as its contemporary instantiations – i.e. web, grid and p2p services - the problem of interoperability is still open. Web services, which are the most well known instantiation of the SOC paradigm, are facing serious interoperability problems, which have been partially addressed by the WS-I Basic Interoperability profile [17]. The proliferation of other instantiations of the SOC paradigm such as grid services and p2p services has further aggravated the problem. The high degree of heterogeneity across all aspects of service-oriented computing, e.g. description, discovery, composition, invocation, etc, render the integration of such services an arduous task. In order to provide for the interoperability among such heterogeneous services we need to establish the problem dimensions and provide appropriate solutions to each of these dimensions.

Within this paper we presented an integrated classification scheme that provides for the identification and categorization of interoperability concerns. This scheme was used for the identification of the service aspects that are affected by the discrepancies among the web, grid and p2p services. As it was expected, the discrepancies among the investigated types of services have an impact on several of the identified interoperability dimensions. Among the highly affected dimensions that have been identified by our analysis are the signature, protocol and quality levels as well as the levels of platform, application and business domain which are orthogonal to the first ones.

In order to address the interoperability concerns at each of the identified interoperability dimensions several actions need to be undertaken. Such actions include the development of standards tackling aspects such as syntactic, semantics and quality of service descriptions as well as the development of appropriate middleware that facilitates the integration of heterogeneous services. For such actions to be effective, conceptual models describing the concepts of each type of service need to be established.

Such a conceptual model was presented in this paper. This model called GeSMO was the basis for the provision of appropriate languages, tools and middleware (developed within the SODIUM project) which facilitate service interoperability. We would like to note that, although the provided generic service model served as a basis for handling interoperability at the signature or platform level, it needs to be further extended in order to fully address the interoperability problem. Thus, issues regarding other interoperability dimensions such as protocol, quality or application as well as semantic or business domain are going to be addressed in our future work. Further future plans for the generic service model include the provision of extensions to address other types of services such as sensor services and the incorporation of additional p2p networks and platforms that provide for other p2p services besides the JXTA platform that is currently addressed.

Acknowledgement: This work has been partially supported by the Special Account of Research Funds of the National and Kapodistrian University of Athens under contract 70/4/5829 and by the European Commission under contract IST-FP6-004559 for the SODIUM project [10].

6. References

[1] N. Medvidovic, D. Rosenblum, and R. Gamble, "Bridging Heterogeneous Software Interoperability Platforms", Technical Report, USC-CSE-99-529, Center for Software Engineering, USC, November 1999

[2] J. Fang, S. Hu, Y. Han "A Service Interoperability Assessment Model for Service Composition", In Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04), Sept. 2004, Shanghai, China

[3] T. Strang, C. Linnhoff-Popien, "Service Interoperability in Ubiquitous Computing Environments", Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w), L'Aquila, Italy, January,2003

[4] A.Ruiz, et al, "Addressing Interoperability in Multi-Organisational Web-Based Systems", Proceedings of the 2nd ECOOP Workshop on Object Interoperability (WOI'2000), Sophia Antipolis, France, June 12, 2000

[5] A. Vallecillo, J. Hernandez, and J. M. Troya, "Woi'00: New issues in object interoperability," in LNCS 1964: ECOOP'2000 Workshop Reader, pp. 256–269, Springer

[6] T. Murer, D. Scherer, and A. Wuertz, "Improving component interoperability information," in Proceedings of Workshop on Component-Oriented Programming (WCOP'96) at 10th European Conference on Object-Oriented Programming (ECOOP'96), pp. 150–158, dpunkt, July 1996

[7] A. Tsalgatidou, T. Pilioura, "An Overview of Standards and Related Technology in Web Services", International Journal of Distributed and Parallel Databases, Special Issue on E-Services, 12(2): 135-162, Sep. 2002

[8] A. Tsalgatidou, G. Athanasopoulos, M. Pantazoglou, "Semantically Enhanced Discovery of Heterogeneous Services", 1st International IFIP/WG12.5 Working Conference on Industrial Applications of Semantic Web (IASW2005), 25-27 Aug. 2005, Jyväskylä, Finland

[9] A. Tsalgatidou, G. Athanasopoulos, M.Pantazoglou, et al. Generic Service Model Specification, Technical Report, available at: http://www.di.uoa.gr/~gathanas/TR/gesmo-1.0-report.pdf

[10] SODIUM project (IST – FP6-004559), www.atc.gr/sodium

[11] A. Tsalgatidou, G. Athanasopoulos, M.Pantazoglou, et. al "D4: Generic Service Model Specification", SODIUM (IST-FP6-004559) Project's Deliverable, Jun 2005

[12] A. Tsalgatidou, G. Athanasopoulos, M. Pantazoglou, et al. "Developing Scientific Workflows from Heterogeneous Services", To appear in ACM SIGMOD-RECORD

[13] M. Burstein, et al. "A semantic web service architecture", In IEEE Internet Computing, Sept.-Oct. 05, pp 72-81

[14] W. Vogels, "Web Services Are Not Distributed Objects", IEEE Internet Computing, Nov.-Dec. 2003

[15] K. Czajkowski, et al. From Open Grid Services Infrastructure to WSResource Framework: Refactoring & Evolution, Version 1.0, Whitepaper, February 2004.

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

[17] WS-I, www.ws-i.org

[18] W3C, Web Service Choreography Description Language (WS-CDL) ver 1.0, Nov 2005, http://www.w3.org/TR/ws-cdl-10/

[19] R. Fielding, et al. Hypertext Transfer Protocol -- HTTP/1.1, IETF, June 1999

[20] T. Bray, et al. Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004, http://www.w3.org/TR/2004/REC-xml-20040204/

[21] A. S. Tanenbaum, Computer Networks, Prentice Hall, , fourth edition, 2003, ISBN: 0-13-066102-3

[22] Keith Ballinger, et al. *Basic Profile Version 1.1*, WS-I specification, 8 August 2004, http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-07-21.html

[23] Web Service Semantics, WSDL-S, W3C Member Submission, http://www.w3.org/Submission/WSDL-S/

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

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

[26] J. Siegel. CORBA Fundamentals and Programming. Wiley, 1996

[27] Sun Microsystems, Enterprise Java Beans 3.0 Early Draft Review 2, 2005. http://java.sun.com/

[28] Microsoft Corporation. Distributed Component Object Model Protocol-DCOM/1.0, draft, November 1996 http://www.microsoft.com/Com/resources/comdocs.asp

[29] C. Pautasso, G. Alonso, "From Web Service Composition to Megaprogramming" In Proceedings of the 5th VLDB Workshop on Technologies for E-Services (TES-04), Toronto, Canada, August 29-30, 2004

[30] H. Hoff, et al., "D7 Specification of the Visual Composition Language (VSCL)", SODIUM (IST-FP6-004559) Project's Deliverable, Jun 2005