CHAPTER 4
GRID-INTEROPERABILITY

4.1 Introduction

The number of web-based information systems has been increasing since Internet became the global open network accessible for all. The recent Semantic Web that provides supplementary meaningful information (meta-data) about Web resources facilitates automatic processing of machines and interoperability between different systems. The integration of heterogeneous data sources in the Semantic Web context using a semantic edition approach based on ontology. The ontology description language OWL to formalize ontologies of different resources and to describe their relations and correspondences allowing the semantic interoperability between these resources[1].

Interoperability is found to be the major aspect of Web services in the WWW. The main thing is of its implementations effectively and proved to be the best for data mining and relationships. The Web services Interoperability and the collaborative work will be the future scenario of WWW. Its implementation typically holds both the perspective of the Internet. The SOAP builders identify interoperability problems and solutions for it.

Due to the increasing development of web data management and collaborative work, the traditional single database system cannot satisfy the requirements of data sharing and interoperation in web environment, and the existing databases cannot be discarded totally, so developing some collaborative software that can access and process the data of multiple pre-existing databases is a necessary trend. A multidatabase system (MDDS) is a layer of software that can integrate a collection of pre-existing,
heterogeneous, distributed database systems called local database systems (LDBSs) [2]. It mainly solves the problem that how to achieve the schema integration and data interoperability among multiple LDBSs[3].

In Grid computing it is an emerging problem in the world of context-centric software-based decision-support is the need for potentially disparate systems to interoperate in meaningful and useful ways. The interoperability used the elementary communication of data and to support a more powerful context-oriented inter-system relationship. It requires such functionality to support and promote meaningful interoperability to share the useful data in the Grid environment.

Interoperability will only be possible where the agent can retrieve ontological statements. Figure 4.1 shows how XML, RDF and OWL relate in this approach. A local ontology could be made available to the application for mapping between the vocabularies that occur in the document stores to be integrated. As new vocabularies are to be supported, mappings are added to the ontology. It would be the task of the application to make the necessary inferences and resolve inferred concepts back to their instances in the document stores.

Figure 4.1: Semantic interoperability through RDF/OWL

(Source : borrowed from[4])
A problem is that OWL assertions can only be layered onto RDF, not XML, and hence ontological mappings can not be applied directly to elements in the NewsML TopicSet, MPEG-7 Semantics or any other XML vocabulary. A RDF layer would need to be added to make interoperability possible.

In other words, the meaningful integration of potentially disparate systems in a manner that allows each collaborating system to retain its potentially unique means of representing, or perceiving, the domain over which it operates[4]. The issue of Grid interoperability is becoming increasingly important as a focus in software engineering and in research areas.

4.2 Interoperability Related to Grid

Service Oriented Computing has revolutionized the way Information Technology is structured and has stimulated great expectation among the developers community. Service Oriented Architectures (SOA) emerged with the promise to support the loose coupling of system parts in a far better way than existing technologies and provide agility, flexibility and cost savings via reusability, efficiency and interoperability both between components of Service Oriented systems and between different systems. However, the lack of agreement on what constitutes a SOA and the vast amount of emerging standards makes it difficult to understand and utilize the potentials of e-Services technologies. In this context, interoperability, which is one of the basic characteristics and benefits of SOA, needs to be further explored in order to find out the open issues and best practices.

The interoperability requirements and related issues in the e-Service area, considering interoperability both in terms of intra- and inter paradigm integration. The goal is to find the challenges and provide a roadmap of best practices for interoperability[5].
Grid computing has changed the way distributed systems are constructed, programmed and used. With a focus on Web services to provide interoperability, service oriented grid computing in particular have introduced a number of new problems which cannot be solved by traditional distributed systems research. One area which is of particular importance is the deployment of services[6].

In Grid The broker which ensures the requirements of the service and the grid environment are taken into consideration during deployment and provides transparent and interoperable service deployment.

There are a number of grid systems which provides support to dynamic deployment. These systems however fail to address the requirements of the service, QoS and trust.

Gridbus is a grid system focused on the grid economy, allowing providers of services offered by the grid to charge for their use. The user of the grid system is able to specify some QoS parameters and query parameters such as price. The Gridbus system then finds a service which matches these requirements. Gridbus has been developed to cooperate with other grids such as Globus and exposes some of the services through a Web service interface [7]. Gridbus, although addressing QoS, does not provide any self configuration mechanisms, including dynamically deploying services in which their requirements are considered.

ProActive is a grid system which has been developed for object oriented parallel processing, mobile and distributed computing [8], and attempts to solve the problem of code reuse by introducing a grid programming and deployment framework. The aim of ProActive is to enable a simple hierarchical deployment model through the use of Java objects and focuses on scalability [9]. Although ProActive provides some self configuration mechanisms (deployment and migration) these mechanisms can only be used with ProActive Java objects. Therefore,
deployment is not dynamic. The deployment mechanism is not interoperable or flexible and do not take into consideration a service or object’s requirements during deployment. 

The Australian BioGrid Portal [10] provides access to molecular docking applications and chemical databases. These services are used to provide data access, analysis and visualisation of molecular screening results using web based tools. The portal to the BioGrid does not offer dynamic deployment with consideration of a service’s requirements. The grid requires that the client deploying the service knows the specifics of each APAC node, their network address and the software installed on each node. This is a very primitive approach which requires the users of the system to be directly involved in constructing and deploying the required services[11].

Autonomic characteristics have the potential to significantly improve service oriented grid environments; self discovery and negotiation, self configuration and self healing mechanisms reduce the complexity of grid systems while improving reliability, interoperability and usability. To ensure autonomic characteristics are provided for all types of grid and Web service systems, and it identify how autonomic computing characteristics can be integrated into service oriented grids[12].

4.3 e-Services and Interoperability

In the e-Services domain we view interoperability as the ability of systems, applications and services, to communicate, exchange data and files, work together or operate on behalf of one another. Standardization is one of the basic ways to achieve e-Services interoperability, therefore, in the following we briefly describe the three e-Services categories with a focus on standardization efforts in each area[13].
"A Web Service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described, and discovered by XML artifacts, and supports direct interactions with other software applications using XML-based messages via Internet-based protocols" (W3C, 2002). What makes the Web Service vision attractive is the ability to discover the Web Services that fulfill users' needs, negotiate service contracts and have the services delivered where and when the users need them. Currently witnessing the rapid development and maturation of a stack of interrelated standards that are defining the Web Service infrastructure along with a great number of development tools that support the Web Service development. The key standards for describing, advertising, discovering and binding WSs are WSDL (WSDL, 2001) UDDI (UDDI, 2005) and SOAP (SOAP, 2001)].

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. Significant effort led by the Global Grid Forum has been channelled toward the standardization of protocols and interfaces. Such effort is the Open Grid Services Architecture Grid Architecture (OGSA) which integrates Grid and Web Services. OGSA was initially materialized by the Open Grid Services Infrastructure (OGSI), and more recently by the Web Services Resource Framework (WSRF) proposal which together with Web Services Notification specification have been submitted to OASIS for standardization. Many Grid communities use the open source Globus tool kit. The term "Peer-to-Peer" (P2P) refers to a class of systems and applications that takes advantage of resources - storage, cycles, content, human presence – available at the edges of the Internet. P2P
systems cover a large spectrum of application domains such as distributed computing, file sharing, collaboration, and platforms[15].

Standards for P2P technologies have not yet been established. Efforts for defining specifications are made by the P2P Working Group whereas two standardization initiatives relevant to P2P networking are Jabber and JXTA. Jabber is an XML protocol for instant messaging applications that is currently being standardized under IETF (IETF, 2005). JXTA is an opensource framework for P2P systems, initiated by SUN that supports only a minimal underlying architecture as a base[16].

4.3.1 Intra-Paradigm Interoperability Roadmap

In each e-Service category interoperability has specific context. In the following we identify the specific requirements for interoperability in the areas of Web, Grid and P2P services and present the ways these interoperability concerns are being addressed by each category of e-Services. Also propose roadmap actions for intra-paradigm interoperability. Furthermore, present synergies between different kinds of e-Services which also promote Service Oriented systems interoperability and present roadmap actions for inter-paradigm interoperability[17].

4.3.2 Inter-Paradigm Interoperability Roadmap

Integration of heterogeneous e-Services allows the exploitation of the specific characteristics and benefits of each e-Services type by the other types of e-Services and the alleviation of their complexities. In the following we present synergies between Web, Grid and P2P services and propose roadmap actions for interparadigm interoperability that could lead to more flexible, efficient and interoperable Service-Oriented systems[18].
4.4 Web Services Interoperability

Web Services promise universal interoperability and integration by establishing commonly agreed protocols for mutually understanding what a service offers and for delivering this functionality in an implementation independent way. Interoperability of legacy applications is also enabled by allowing legacy applications to be exposed as WSs and, thus, facilitating a seamless integration between heterogeneous systems. Furthermore, new services can be created and dynamically published and discovered without disrupting the existing environment. Thus, WS technology provides a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. The issue of WS interoperability is addressed at a conceptual level by the W3C's Web Services Architecture (WSA) which identifies those global elements of the global Web Services network that are required in order to ensure interoperability between Web Services. The various WS standards and enabling technologies address technical level interoperability.

The common standards for the basic WS activities description, publication and invocation (WSDL, UDDI, SOAP) have effectively become de facto standards, and, thus, support basic technical interoperability. However, there is a need for enhanced interoperability both for basic WS activities, like description, as well as value added ones, like WS management and composition and also for non functional aspects like security.

The following, we describe the interoperability requirements concerning the aforementioned aspects. Web Services description languages should support, beyond the functional characteristics of WSs, the description of semantic information that will enable the proper and meaningful interoperation between different WSs or applications.
Semantic description is not supported by WSDL and is partially addressed by ebXML. Web Service management which refers to monitoring, controlling, and reporting of WS qualities and WS usage, also requires semantics, management policies and management capabilities, which should be understood by the requester and provider[19].

In order to improve the security and reliability of WSs and to address more complex business scenarios that require WSs composition, a wide range of protocols have been proposed, the most prominent of which is BPEL4WS[20].

In order to address the aforementioned interoperability issues, the proposed roadmap [21] are:

i. Monitor progress of protocols through key standards bodies,

ii. Establish policy for compliance with standards,

iii. Use WS-I profiles wherever relevant. Create local profiles only where necessary, and plan to upgrade to WS-I as they are published,

iv. Coordinate use of protocols to ensure consistent implementation of versions and profiles. Publish best practices,

v. Use ontologies and ontology languages, e.g. OWL, for provision of semantics concerning WSs activities, e.g. WS description and WS management,

vi. Use implementations of standards that have proven interoperability,

vii. Wherever possible wait for implementation of protocols in products, and

viii. Keep up with the advancement of standards.
4.5 Grid Services interoperability

Grid services interoperability can be viewed along two different dimensions:

- interoperability between distributed resources in a Grid application, and
- interoperability between different Grid applications.

Interoperability between different distributed resources in a Grid application is a main goal of the various Grid projects despite the different infrastructures they use. E-Services interoperability analysis and roadmap actions and the different aspects on which they focus. The OGSA/OGSI and WSRF models provide a framework for Service Oriented Grids aiming at supporting interoperability of distributed services and resources. Grid middleware implementations based on these models.

IGENE provide services such as identification, authentication and authorization and promote interoperability by allowing interoperation of Grid components independently of the operating system and network topology. However, there are interoperability issues when different entities use the Grid for sharing purposes. Efforts are being undertaken to deal with the authorization of a community in a Grid environment like the Community Authorization Service (CAS) of Globus.

Significant effort has also been channelled toward enabling interoperability between different Grid applications. There is a number of different approaches that include the definition of a minimal set of Grid services which enable the interoperation of different Grid applications. The integration of different Grid infrastructures and the common Grid resources description and the Semantic Grid which aims at providing interoperability across time as well as space both for anticipated and unanticipated reuse of services, information, and knowledge. Furthermore, the Enterprise Grid Alliance has recently released a reference model for
enterprise grids, laying the foundation for standardized solutions that will enable cross-departmental and cross-organizational interactions. The above analysis of Grid services interoperability issues leads to the identification of the following roadmap [28] are:

i. Monitor progress of protocols through key standards bodies,

ii. Adapt standards that have gained broad acceptance (e.g. WS+I standards),

iii. Conform to a reference model (e.g. EGA’s reference model for enterprise grids),

iv. Adapt semantic grid services vision, and

v. Use tools that have gained industry acceptance (e.g. Globus).

4.6 P2P Services Interoperability

P2P services interoperability, similarly to Grid Services, can be viewed either as:

- interoperability between different peers in a P2P network and
- interoperability between different P2P applications.

Interoperability between different peers of P2P systems needs advanced interoperability techniques, since the various member nodes of a P2P network that have variety of operating systems, networking technologies and other platforms in business applications need to communicate, exchange content and aggregate their diverse resources, such as computing power or storage space. Most P2P systems use proprietary implementations and protocols for the peers interoperability and functionality. XML is expected to support peers interoperability in P2P systems, since it can be used for messaging, data storage, application deployment, resource description and distributed search[29]. A uniform resource description is commonly agreed to be a basic requirement that can be addressed by ontologies and standardized ontology languages, e.g. OWL, or local schema
mappings[30] and semantic routing which are the basis of several recent proposals for P2P data management.

Sun Microsystems' JXTA and Microsoft's .Net Framework[31] are the most significant efforts to define the core elements of a P2P platform. Based on the above, the proposed roadmap actions for P2P services interoperability are as follows[32]:

i. Agreement on the use of uniform P2P resources description, e.g. using ontologies, OWL and local schema mappings,

ii. Agreement on an underlying platform with core P2P elements for building P2P systems,

iii. Establishment of a collection of P2P standards that could launch P2P as a global way of doing business and

iv. Use of XML in P2P systems

The technologies enabling this convergence are some of the leading P2P protocols such as Jabber and JXTA, whereas Microsoft .NET Framework provides a rich platform for building P2P applications and Intel is building P2P components and services on top of Microsoft's .NET platform[33]. The techniques that the P2P and Grid models use to handle some of the main issues of distributing computing are discussed in[34] in order to find a common foundation that could alleviate the complexities of each other and fulfill the need for secure, scalable and decentralized collaboration. Another approach to combining aspects of Grid computing with P2P architectures is found in the proposal for a new architecture stack for Grids presented in the Next Generation Grids second Group's report[35]. Based on the above, the proposed roadmap actions for interparadigm interoperability are as follows[36]:

i. Monitor progress of protocols in the e-Service area,

ii. Monitor integration efforts,
iii. Adapt interoperable protocols for synergies between different kinds of e-Services, e.g. JXTA and WS standards and

iv. Adapt integration paradigms between different e-Services to alleviate the complexities of each other.

Table 4.1: Interoperability approaches and roadmap actions for e-Services

<table>
<thead>
<tr>
<th>Standards Efforts</th>
<th>Web Services</th>
<th>P2P services</th>
<th>Grid services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Architecture</td>
<td>Web Services Architecture (W3C)</td>
<td>Proposed architecture: JXTA</td>
<td>OGSA, EGA’s Reference Model for enterprise grids</td>
</tr>
<tr>
<td>Standards for basic activities (description, publishing, invocation)</td>
<td>WSDL/UBDI/SOAP initiative, ebXML</td>
<td>Standardization efforts: JXTA, Jabber for instant messaging systems, P2Pwg</td>
<td>OGSA/OGSI and WSRP models, no standardized Grid middleware implementations</td>
</tr>
<tr>
<td>Standards for value added activities</td>
<td>Many standardization efforts (BPEL4WS, WS-security) but a few mature standards</td>
<td>No standards</td>
<td>OGSA security architectural components, no completely standardized implementations</td>
</tr>
<tr>
<td>Semantics support for basic and value-added activities</td>
<td>Partially addressed (e.g. ebXML)</td>
<td>New approaches: Semantic routing (e.g. RDF-based routing algorithms, etc.)</td>
<td>Partially addressed-Semantic Grid vision</td>
</tr>
<tr>
<td>Approaches to intraparadigm Inter/ty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Most different P2P systems do not interoperate, exceptions: Jabber with other M systems. Magi with JXTA, different JXTA–built systems.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Need for interoperability between different infrastructures, Grids middleware enable grid resources interoperability.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intraparadigm Inter/ty Roadmap Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Monitor progress of protocols.</td>
</tr>
<tr>
<td>2. Establish policy for compliance with standards.</td>
</tr>
<tr>
<td>3. Use WS-I profiles.</td>
</tr>
<tr>
<td>4. Use interoperable standards and keep up with the advancement of standards.</td>
</tr>
<tr>
<td>5. Provide semantics.</td>
</tr>
<tr>
<td>6. Wherever possible wait for implementation of protocols in products.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Integration efforts for heterogeneous e-Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2P based WS discovery</td>
</tr>
<tr>
<td>Peer discovery in P2P systems using WS as registries</td>
</tr>
<tr>
<td>Search engines built using WS and P2P technology</td>
</tr>
<tr>
<td>WS Grids based on specific WS standards (WS-I+ profile)</td>
</tr>
<tr>
<td>Grid architecture based on P2P principles</td>
</tr>
<tr>
<td>Convergence of e-Services for unified service discovery</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other approaches &amp; Roadmap Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Agree on a uniform P2P resources description, e.g. using ontologies, OWL, and local schema mappings.</td>
</tr>
<tr>
<td>2. Agree on a common platform for building P2P systems.</td>
</tr>
<tr>
<td>4. Use XML in P2P systems development.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other approaches &amp; Roadmap Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Monitor progress of protocols.</td>
</tr>
<tr>
<td>2. Adapt standards that have gained; broad acceptance (e.g. WS+ I standards).</td>
</tr>
<tr>
<td>3. Adapt semantic Grid Services vision.</td>
</tr>
<tr>
<td>4. Use tools that have gained industry acceptance (e.g. Globus).</td>
</tr>
</tbody>
</table>
As more and more special-purpose, non-interconnected software-intensive systems for application areas varying from embedded weapon system software to logistic management systems are being used, the issue of interoperability is becoming increasingly important as a focus in software engineering research. Advances in computer communications technology, the recognition of common areas of functionality in related systems, and an increased awareness of how enhanced information access can lead to improved capability are driving an interest toward integration of current stand-alone systems to meet future system requirements. A prime difficulty in achieving interoperability among heterogeneous components of a system federation is that the component systems were developed independently, without any requirement for interoperation. Thus systems have different architectures, different hardware platforms, different operating systems, different host languages, and different data models. Short of redeveloping a new system using the consolidated requirements from the various component systems and a common architecture, hardware platform, operating system, host language, etc. (a cost prohibitive approach), a means must be devised to achieve the goal of component interoperability in the face of expected limited acquisition budgets. In order to overcome the
limitations identified above, we have explored technologies and methods for achieving system interoperation[37].

4.7 Methods Of Interoperability

The interoperability supports web services-oriented interoperability among systems operating over potentially disparate representations. The intelligence and semantic web-based design incorporates technologies such as inference engines, rule-based systems, XML, XSLT, web services and service-oriented architectures to provide the needed infrastructure to support meaningful interoperability among distributed systems in Grid environment.

Grid computing aims at integrating geographically distributed resources into a sharing and cooperative environment. Grid monitoring needs to cover diverse resources within a heterogeneous environment.

System interoperability involves not only the ability of systems to exchange information but also includes the capability for interaction and execution between the systems providing Interoperability where the systems may have different architectures, different hardware platforms, different operating systems, different host languages and different data models. A prime difficulty in achieving interoperability among heterogeneous components of a system federation is that the component systems were developed independently, without any requirement for interoperation. The heterogeneity of systems and databases is also an one of the difficulties faced in creating interoperability among the systems of Grid. The complete realization of the potential synergistic benefits can only be achieved if system interoperability is attained. This criteria was used to evaluate seven of the leading approaches for achieving interoperability among independently developed systems.
A number of architectures, technologies, applications, and languages claim to provide support for component or system interoperation. The selected seven of the leading approaches claiming interoperability support for comparison, these Selected approaches include the Common Object Request Broker Architecture (CORBA), Microsoft’s Component Object Model (COM) line of technologies, Microsoft’s .NET platform, Sun’s Java 2 Enterprise Edition (J2EE), SeeBeyondTM Technology Corporation’s integration suite, the High Level Architecture (HLA) for modeling and simulation, and the World Wide Web Consortium’s (W3C) eXtensible Markup Language (XML) [38].

These seven approaches for improving the interoperability among distributed systems are discussed as follows.[All approaches are taken from [39].

4.7.1 Common Object Request Broker Architecture(CORBA)

The Object Management Group (OMG) has as one of its goals "to provide a common architectural framework for object-oriented applications based on widely available interface specifications" [40]. The OMG achieves this with the establishment of the Object Management Architecture (OMA) of which the Common Object Request Broker Architecture (CORBA) is a part. The OMA is a set of standards that provides a common architectural framework on which applications are built. The OMA consists of an Object Request Broker (ORB) function, Object services, Common facilities, Domain interfaces and Application objects [41].

- it provides a standard mechanism for defining the interfaces between components;
- it specifies a number of standard services such as directory and naming services, persistent object services, and transaction services that are available to all CORBA compliant
applications; and it provides the mechanisms to allow application components or separate CORBA provides capabilities in three areas to support interoperability:

- applications to communicate with each other.

These capabilities are provided in a platform and language independent fashion [42]. CORBA provides the capability for addressing heterogeneities of hardware and operating systems, organizational models, and for partially resolving heterogeneity of presentation. CORBA also provides the capability for heterogeneity resolution in both an information exchange and joint task execution scenario. However, CORBA’s shortcomings include

i. failure to address the complete spectrum of heterogeneity;

ii. lack of assistance in correlating different models of the same realworld entity on component systems;

iii. lack of assistance in defining the translations required to resolve other than lowlevel hardware and operating system differences;

iv. required prior knowledge by a client system of a server’s method name and the type and model of the method’s parameters in order to utilize their functionality; and

v. modification to existing systems not developed in compliance with the CORBA standard is required in order to enable system interoperation.

4.7.2 COM, DCOM, COM+

Introduced by Microsoft in 1993, the Component Object Model (COM) is a software architecture that enables applications and systems to be built from binary components supplied by different software vendors. COM and its successor architectures Distributed COM (DCOM) and COM+ are competing technologies to the OMG’s CORBA.
COM’s original function was to provide a general purpose mechanism for component integration on Windows platforms. DCOM added support for distributed components when introduced on Windows NT and on Windows 95. COM+ provided a unification of COM, DCOM and Microsoft Transaction Server when introduced with Windows 2000 [43].

In evaluating COM, DCOM, and COM+ against the criteria used to compare interoperability approaches, the three technologies are assessed as one as they represent the evolution of the same concept. The COM+ architecture family provides a capability similar to that provided by CORBA. In addition, it enables interoperation among binary software components whereas CORBA addresses interoperability at the source code level. The COM+ family shares CORBA’s failure to address the complete spectrum of heterogeneity and lack of assistance in correlating and resolving differences in real-world entity models. The COM+ family requires prior knowledge of remote system methods in order to utilize their functionality and requires modification to existing systems not developed in compliance with COM+ standards in order to enable their interoperation.

4.7.3 Microsoft .NET

Microsoft .NET is a platform built on top of the Windows operating system that provides the infrastructure needed for solving common problems in Internet applications. .NET provides a number of capabilities that support interoperation of systems and components, including a Common Language Runtime (CLR), an intermediate language which removes differences in implementation of system components, and an interoperability mechanism that enables .NET programs to access legacy code. The heart of .NET is the CLR which is the next generation successor to the component technology provided in Microsoft’s COM. While CLR replaces COM, the support for distributed component use provided in
COM+ has been retained in .NET. The COM+ services used for building distributed systems have been incorporated into .NET Enterprise Services.

The CLR relies on the Microsoft Intermediate Language (MSIL) to provide a CPU-independent set of instructions that can be efficiently converted to native code, similar to the approach used by Sun’s Java Virtual Machine. MSIL implements a Common Type System (CTS) which defines the supported types in the CLR and specifies how they interact with each other. All development tools produce the same MSIL, regardless of the language in which their source code is written. Therefore, differences in implementation are gone by the time they reach the CLR. The MSIL code produced by the development tool is then compiled by a Just-in-time compiler (JITter) to produce the actual machine code that runs on a specific platform [44].

Microsoft .NET provides interoperability mechanisms that enable the integration of .NET programs and legacy code. Specifically, .NET supports three scenarios for interfacing with legacy code:

I. it enables applications that are part of the managed code environment provided by the CLR to access unmanaged Dynamic Link Library (DLL) functions;

II. it provides for managed code to instantiate and call interface methods of COM servers; and

III. it allows unmanaged code to instantiate and call methods on .NET servers [19].

The CLR provides .NET with a programming environment that supports cross-language interoperability where classes and objects in one language can be used in another language without the use of a specialized Interface Definition Language (IDL). This contrasts with the approach used by COM and CORBA which permit methods of a remote class to be called as an alien only if the call adheres to a standard calling model. In .NET you
may define a class in one language and then, using a different language, derive a class from the original one or call one of the original class’s methods. This passes an instance of a class to a method on another class that is written in a different language. The cross-language support provided by .NET is currently limited to four languages—Visual Basic, C++ with Managed Extension, C# (C Sharp), and JScript. With its heritage as a Windows platform and with its cross-language capability limited to four languages, .NET should not yet be considered as a broad-based interoperability solution.

4.7.4 Java 2 Enterprise Edition (J2EE)

The Java 2 Enterprise Edition (J2EE) specification “defines a Java platform with features aimed at enterprise level computing environments” [45]. It extends the Standard Edition specification primarily in the areas of security, deployment and interoperability. In support of interoperability it provides a number of distributed computing protocols and APIs that can be used in creating a system federation. Berg identifies four cornerstones for creating a distributed application:

i. Data resources,
ii. Naming and lookup of resources and services,
iii. Remote invocation or messaging, and
iv. Transaction control

Of these, remote invocation or messaging is considered key to achieving system interoperability. J2EE provides the capability for remote method invocation or message exchange using one or more of the following technologies. The capability for remote method invocation is provided by either Java Remote Method Invocation (RMI) or CORBA. The capability for message exchange among applications is provided by Java Message Service (JMS). Remote Method Invocation (RMI). Java’s built-in distributed object protocol, RMI, enables the definition of objects that can
be called remotely from other applications in a network. This capability provides the foundation for joint task execution, and consequently information exchange, among systems in a federation. RMI handles the details of packaging method parameters, sending them across a network to a remote object, unpackaging them at the destination, invoking the correct method using the passed parameters, and returning any method result back to the caller. RMI provides an analogous capability to that presented by CORBA, but does not include all of CORBA’s power. RMI uses a simpler, standardized model for establishing and managing connections between a client and server.

RMI uses Java’s serialization API for packaging and unpackaging data during transmission across the network. Serialization is used to convert an object’s state into a machine-independent encoded form that is transmitted between applications or systems. This encoded form is then reconstructed into an equivalent object at the destination end. Any machine-specific representational differences are resolved through this process. CORBA. Whereas Java applications can be made to Interoperate with other Java applications using Java RMI, they can also interoperate with non-Java applications and Java applications as well, using CORBA’s ORB. As stated by Berg “Java and CORBA are extremely synergistic. Java solves the problem of code distribution; CORBA solves the problem of intercommunication between distributed components combined, they provide an architecture for creating distributed applications that can deploy themselves and run in a cooperative fashion across a network”[46].

RMI’s native protocol for communications between a client and server is called Java Remote Method Protocol (JRMP). Alternatively, RMI can use CORBA IIOP to affect the exchange of method parameters between applications. Java Message Service (JMS). With Java RMI or CORBA, a
designer can create a system federation capable of providing joint execution of tasks and information exchange among systems. As an alternative to the remote invocation protocol provided by Java RMI, JMS provides a messaging service that can be used for information exchange among federation systems. JMS provides both a point-to-point and publish-subscribe messaging model. In the point-to-point model applications send and extract messages from named queues. In the publish-subscribe model, applications publish messages to named “topic” channels and listen asynchronously for arriving messages on these topics. J2EE presents a competing approach to distributed computing to that provided by CORBA and the COM+ family. J2EE’s strengths include its support for both information exchange and joint task execution among federation systems, its use of a two-step translation methodology for resolving heterogeneities of hardware and operating systems, and its full support provided for both federation extension and modification. Its shortcomings include

i. failure to address the complete spectrum of system heterogeneity,

ii. lack of capability for assistance in establishing correspondence between different models of the same real-world entity,

iii. requirement to know the server’s name or identifying attributes in order to invoke the server’s methods from a client,

iv. requirement that both client and server applications be written in Java, and

v. lack of assistance in defining the translations needed to resolve system heterogeneities outside of the platform independence provided by the Java Virtual Machine.
4.7.5 SeeBeyond integration suite

The SeeBeyond™ integration suite provides an architecture and set of tools designed for the integration of incompatible legacy systems, databases, packaged applications, iddleware products, communication protocols, messaging standards, and data access paradigms. SeeBeyond targets to provide an eBusiness Integration (eBI) solution to the integration of incompatible and noninteroperable business applications. The two primary components of the SeeBeyond integration suite are e*Gate™ Integrator and e*Index Global Identifier™. SeeBeyond’s e*Gate Integrator provides the framework for their eBusiness integration solution [47]. Among the capabilities provided by e*Gate is the ability to

i. Manage information exchange between legacy systems and Web Servers;

ii. Integrate systems based on COM, CORBA, and Java;

iii. Serve as a universal gateway between Oracle, SQL Server, Sybase, Informix, DB2, and older-technology databases, and

iv. Provide an Enterprise Integration Backbone.

The e*Index Global Identifier application from SeeBeyond runs on the e*Gate Integrator platform and is designed to allow the sharing of customer information between disparate systems. The e*Index application employs a relational database of persons’ records with each record automatically cross-referenced to the various local identifiers used for that person in referenced local systems. This cross-reference provides a single global identifier for a person that corresponds to the multiple local identifiers that person may have in different systems.

The evaluation of interoperability characteristics is done for SeeBeyond’s eBusiness suite- their e*Gate Integrator and e*Index Global Identifier. The results of the evaluation are summarized below. SeeBeyond’s e*Gate Integrator and e*Index Global Identifier address each
of the eight classes of heterogeneity, although some are only partially dealt with. Specifically, these products provide for only partial resolution of heterogeneity of organizational models. SeeBeyond does provide limited support in correlating different models of the same real-world entity on different systems. Currently this support is limited to the correlation of customer ID’s through its e*Index component. SeeBeyond’s eBusiness suite is primarily focused on enabling information exchange among heterogeneous systems, but the capability for joint task execution can be provided through the application of CORBA, COM+, or Java RMI capabilities and limitations of it are

i. prior knowledge of remote system methods is required, and

ii. modification to existing systems is needed if they are not originally developed in compliance with the specified standards.

SeeBeyond does provide computer-aid to translation development; however, it does not support definition of an intermediate representation for translation, relying primarily on the point-to-point conversion between specified source and destination representation pairs.

4.7.6 The High Level Architecture (HLA) for modeling and simulation

The High Level Architecture (HLA) [48,49] is a software architecture designed to enable individual computer simulations to be combined into larger simulations. In the HLA, the combined simulation system created from a compilation of individual computer simulations is termed a federation. The elements of a federation consist of a number of federates, a Runtime Infrastructure (RTI), and a common object model of data exchanged between federates, the Federation Object Model (FOM).
Rules governing the interaction between federation elements are contained in the HLA standard, as are templates for defining the patterns to be followed by those interactions. An evaluation of HLA’s interoperability characteristics is summarized in the following. HLA, while providing facilities for combining individual computer simulations into larger simulations, does little to address most of the limitations identified with previously evaluated approaches. HLA does present an object-oriented model for capturing data shared between systems in a federation, increasing the developer’s visibility of system interaction. It also prescribes a publish-subscribe approach to data sharing, requiring interconnections between components be established only when one system has data of interest to another (or is interested in data from another system). The two primary limitations of the HLA are its capability for information exchange only, and its failure to address the possible heterogeneities among federate models of shared data. The prohibition of direct interaction between federates and the lack of behavioral information in the Federation Object Model (FOM) precludes the use of the HLA for joint task execution. HLA’s support for data exchange is limited by its leaving the burden of attribute and parameter interpretation to the federates and its failure to provide facilities in the Object Model Development Tool to facilitate model correlation and to develop translations to resolve differences between models.

4.7.7 Extensible Markup Language (XML)

The Extensible Markup Language (XML) has been highly publicized as a means for achieving system interoperability. XML, developed by the World Wide Web Consortium (W3C), provides a self-describing means for representing the data used by and shared among applications. It is an extension of the Standard Generalized Markup Language (SGML) and is designed to provide the flexibility and power of SGML while attempting to
capture the widespread acceptance and relative simplicity of another SGML derivative, the HyperText Markup Language (HTML) [50]. The cornerstone of the eXtensible Markup Language is the XML document, which consists of a sequence of data elements and element attributes with “tags” used to delimit and describe the meaning and use of the data to the user or relevant application. An XML document can consist of any permissible sequence of elements and their attributes. In order that the pertinent application might be better able to understand and utilize the data in a received document, XML utilizes a schema to structure and delimit the allowable values for XML documents. This schema can be in the form of a Document Type Definition (DTD) or of the newer XML Schema. Interfaced systems can utilize the DTD or XML Schema to specify the vocabulary of a system’s allowable data [51]. XML is actually a family of technologies. In addition to the DTD and XML Schema, XML’s immediate family includes the Document Object Model (DOM), the Simple API for XML (SAX), and the eXtensible Stylesheet Language (XSL). While not providing a distributed computing facility such as that offered by CORBA, COM+, or J2EE, or an architecture for combining independently developed systems such as that presented by SeeBeyond’s integration suite or HLA, XML does provide support for achieving interoperability among independently developed systems. Of the seven approaches presented, XML offers the greatest support for heterogeneity resolution, addressing. In addition, there are tools available that aid in the creation of XSLT stylesheets used for resolving heterogeneities of structure and meaning between application data models. Finally, XSLT can be applied using a rapper or middleware-based approach, reducing the requirement for system modification during heterogeneity resolution.

XML defines a mechanism for data definition and organization; it does not provide a method for remote invocation of methods between
applications. However, a companion specification, SOAP Version 1.2, provides a messaging framework for exchanging XML messages among heterogeneous platforms [52]. Included in the SOAP specification is a method for encapsulating Remote Procedure Calls (RPCs) within SOAP messages. This capability enables developers to embed CORBA, COM+, Java RMI, or other RPC calls within SOAP messages. Thus, in addition to enabling the use of XML’s significant capabilities for heterogeneity resolution during information exchange, SOAP also provides the means for the joint execution of tasks in an XML environment. While offering significant support for system interoperation, XML also shares some of the limitations found in the other approaches. These include failure to address the full spectrum of heterogeneities, and lack of facilities for correlating different models of the same real-world entity.

4.8 Semantic Web and Grid mining

Semantic Web technologies are based on open standards. If a computer understands the semantics of a document, it does not only interpret the series of characters that make up that document but also understands the meaning of the information in the document. It is also defined as the extension of the today’s web where the information is given the better meaning and allowing users to work in hand-in-hand to use the resources of web. Also allowing the software agents and programs to interoperate for extracting the useful knowledge and solving the complex problems in the internet.

The need of metadata (which annotates and describes Web content) to allow machine-to-machine operation, it is required to have automated processing to give semantic meaning to each Web resource. The aim of the Semantic Web is to provide machine processable metadata that describes
the semantics of resources to facilitate the search, filter, condense, or negotiate knowledge for their human users. Application areas like Knowledge Management and Ecommerce, is the field of Semantic annotation, which turns human-understandable content into a machine understandable form. [53].

Grid mining, In recent years, substantial effort has been made in researching and developing interoperable data management systems. As more and more special-purpose, non-interconnected software-intensive systems for application areas varying from embedded system software to logistic management systems are being used, the issue of interoperability is becoming increasingly important as a weapon for mining the Grid and also in software engineering research. Advances in computer communications technology, the recognition of common areas of functionality in related systems, and an increased awareness of enhanced information access can lead to improved capability and organizing data. The interoperable data management systems have major focus on data model mapping, communication protocols, and database integration, including schema integration, query processing, consistency management and security management[54].

The interoperability must also deal with real-time issues and justify the need for extending interoperability research into real-time. It is required to identify a number of research issues in integrating real-time data management systems. These issues do not exist in either traditional distributed database systems or multi databases; they must be addressed in the context of real-time system integration.

For interoperability between applications, the information processing must be supported by an architecture providing distribution technology and transparent access to information through heterogeneous networks and environments. The distribution of information has been done with a strong
connection with current concepts and advances in the field of object oriented technology.

The advances in the Internet has emphasized the need for interoperability among different and heterogeneous systems, i.e. their ability to exchange data and services in a transparent way. The interoperability in a GISs context where systems are geographically distributed which needs to exchange spatial data. Its mission is to unification of spatial data models and services in order to achieve interoperability among GISs. The interoperability at different levels of abstraction applied. At the highest level (the spatial data on the Internet) a set of common interfaces are adopted to represent both data and services and at the bottom level (database level), the concept of a specialized sewer (called OGIS/SQL server) is introduced, which is the entity that tries to obtain data integration among the data sets[55].

Grid interoperability involves not only the ability of systems to exchange information but also includes the capability for interaction and joint execution of tasks. Provides system interconnectivity along with interoperability among systems. A prime difficulty in achieving interoperability among heterogeneous components of a systems is that the component systems were developed independently, without any requirement for interoperation. Thus systems have different architectures, different hardware platforms, different operating systems, different host languages and different data models. Short of redeveloping a new system using the consolidated requirements from the various component systems and a common architecture, hardware platform, operating system, host language, etc. (a cost prohibitive approach), a means must be devised to achieve the goal of component interoperability in the face of expected
limited acquisition budgets. In order to overcome the limitations as above it is required to have new technologies and methods for achieving system interoperation and interoperability[56].

References


[27] EGA (Enterprise Grid Alliance), http://www.gridalliance.org


[36] Aphrodite Tsalgatidou, Eleni Koutrouli,” E-SERVICES INTEROPERABILITY ANALYSIS AND ROADMAP ACTIONS”.
[37] Paul Young, Nabendu Chaki, Valdis Berzins, Luqi,” Evaluation of Middleware Architectures in Achieving System Interoperability” 1074-6005/03 © 2003 IEEE.
[38] Paul Young, Nabendu Chaki, Valdis Berzins, Luqi,” Evaluation of Middleware Architectures in Achieving System Interoperability” 1074-6005/© 2003 IEEE.


