Review

Middleware Technologies: Chain Web Grid Services

Zeeshan Ahmed, Saman Majeed
Department of Bioinformatics
Biocenter, University of Wuerzburg
Am Hubland 97074, Wuerzburg. Germany
{zeeshan.ahmed, majeed.saman}@uni-wuerzburg.de

ABSTRACT: *In this review, we address the importance of field i.e. Middleware Technology; a mediate between more than one distributed web application and client based desktop application for efficient data exchange and resource sharing. While describing the technological contributions of Middleware Technologies in detail, we present some of its major technologies i.e. Web Service, GRID Computing, Open Service Grid Infrastructure, which are involved in web, embedded and enterprise application development e.g. Portals, E-Science and E-Learning Platforms, Games etc. Furthermore, in this review paper, we present new Middleware Technology concepts i.e. Web Grid Service and Chain Web Grid Service. Going into the details, we present the conceptual architecture of newly proposed concepts along with information of implementation using existing web technologies, scripts and programming languages.*

Keywords: Grid, Middleware, OSGA, Security, Web Services

Received: 11 August 2011, Revised 29 September 2011, Accepted 3 October 2011

1. Introduction

Technologies are growing like a dense tree, whose every branch is with a new way towards further advancements of existing solutions, every leaf is becoming a new idea of innovation and benefiting with different fruits. One of the branches of this tree is Middleware Technologies and this branch is becoming dense enough with more leaves and fruits. Middleware technologies are mainly distributed online web applications promoting the concept of sharing resources and information amongst other web and desktop applications [1] [2]. A lot of development has already been performed and future research is in progress, towards different areas of this field to improve and to invent.

One of the most recent topics of today is encompassing technologies to connects computer systems and provide a protocol to interact with each other to share the information between them. The term "Middleware" carries the meaning to mediate between more than one distributed web applications to efficiently exchange data. This is a decentralized approach towards centralized application development for minimizing the load of data and service exchange by making them heterogeneous. Middleware services help the software developers in a way that they need not to implement the some important modules during their web applications e.g. *data transactional module for transactional services, online security module, event management module, over network communication management module, system maintenance module, perseverance module for non volatile data and request handling module* etc., as shown in Figure 1.

The presented online distributed web application architecture in Figure 1, can be adopted into service oriented web based middleware application development. Self facilitating, a Web application just has to use middleware technologies (MDLW Tech.) to access provided services and need not to take care of the communication protocol(s), operating system(s), concurrency service(s) and hardware compatibility issues. Using MDLW tech software development methodology, a software developer can save his time by implementing reliable web resource sharing applications in a possible short period of time. Most of the MDLW Tech based applications follows the client server architecture following the Remote Procedure Protocol (RPC) [6] and marshalling (conversion of actual to network transmission data format) and de-marshalling (conversion of network transmission data format to actual). Due to progressive reputation in international software markets, most of the major organizations like Microsoft or Sun Microsystems etc. are also adopting, developing and using Middleware technologies [3] i.e. SOAP[7], Web Services [8] [9], CORBA[10] [11], Grid Computing [12] [13], OSGI [14], SNMP [15] etc.

There are three sub-braches of middleware technologies i.e. *Web service, Grid Computing and Open Grid Service*. Web service is one of the new sub-branch of this middleware technologies, which is growing more rapidly aiming the promotion of sharing resources between web applications by creating online resource sharing environments, with the involvement of the concept of artificially intelligence e.g. Multi agents system (MAS) has also been involves in the web services to implement an automated online work environment. Another sub branch of the middleware technologies is Grid Computing, and the aim of this branch is to implement the concept of virtualization to heterogeneous entities to provide distributed infrastructures. This sub branch is further extending some other sub branches, one of them is Open Service Grid to provide ubiquitous access to the systems with the implementation of Grid Computing and Web (Grid) Service as the combination of both the Grid Computing and the Web Service to have the benefits of both the Grid computing and the web services in web applications for improved resource sharing in a fast and secure way with in virtual environments by ubiquitous having access.

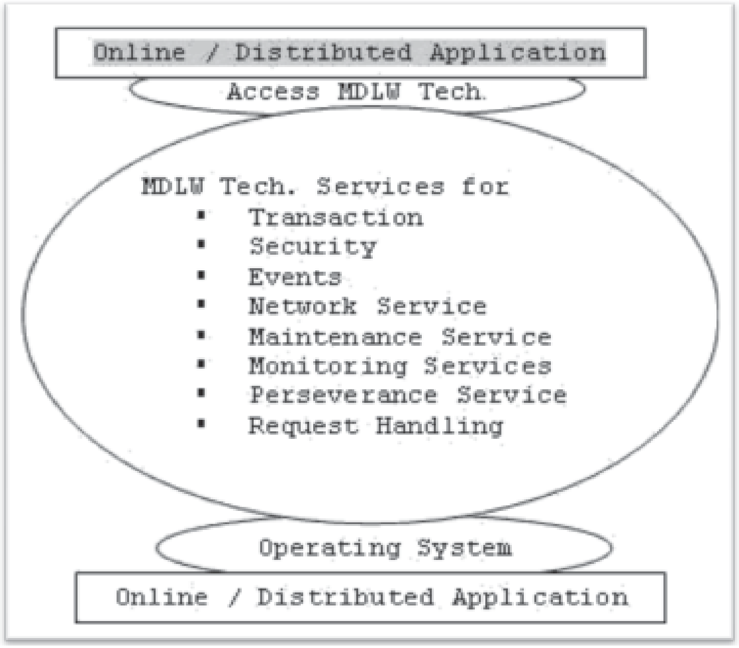


Figure 1. Online Distributed Web Application Architecture

Figure Legend: Online Distributed Web Application Architecture Implementation using Middleware Technologies. This architecture consists of five main layers i.e. *online Application, Access MDLW Tech., MDLW Services, Operating System and Online Application*.

Since almost 10 years, Middleware has appeared to be an imperative and conceptually implementable component based architecture with the provision of prevailing tools especially towards the distributed and enterprise application development [16]. This Middleware is providing high quality values in special purpose, intelligent and data management based web applications belonging to the different field's e.g. *Online experimental data management [31] and sharing application development for quantitative computational analysis [17, 18, 25] and data classification [19], online massive multiuser games development [20, 21, 22], web based product data management system implementation [23, 24], middleware mediation software development [26], message oriented middleware application development [27], service oriented web based enterprise application development and reliable autonomous Web Service composition [28] and Product Line Architecture [29, 30] development using middleware [32, 33]* etc.

The remainder of this review is organized as follows: present three sub braches of middleware technologies, section 2 presents Web Services, section 3 describes Grid Computing and section 4 states Open Service Grid Infrastructure. Section 5 presents our newly proposed middleware Web Grid Service concept and shows how this can be implemented with existing middleware tools and technologies. Section 6 concludes the discussion.

2. Web Service

Before the proposition of the concept "*Web Service*", Distributed Common Object Model (DCOM) was the most recent solution (since 1995) promoting the idea of creating Common Object Model (COM) components to communicate across local and wide area networks by using at first Remote Procedure Call (RPC: mechanism to send and receive the data between the COM components) and later then Common Object Request Broker Architecture (CORBA: mechanism to enable separate pieces of software programmed using different programming languages and running on different computing systems but to work with each other like a single application or set of services) was proposed.

Web Service is a platform and language independent middleware technology, based on XML standards for interoperability and some transport protocols to exchange data between the applications. Implementation of Web Service (renowned as Application Service) helps web application developer by providing better data transactional processes, reliable system messaging, pragmatic interface across the networks, secure web environment and standard way to integrate web applications in distributed real time, embedded and online systems. Furthermore describing more benefits of Web Service, it is economic to afford, efficient and quick in performance than other middleware technologies.

Web Service Description Language (WSDL) is the main language for Web Service development with the application of dynamic behavior at run time including message formats, data types and protocols. Furthermore WSDL is capable of understanding XML schemas and requires no such web application browser to execute but to be accessed by any operating system as a service. Basic work flow of the Web Service provides a simple way of performing RPCs over hyper text transfer protocol (HTTP) using Simple Object Access Protocol [4] (SOAP – a protocol for structured information exchange using Web Services in computer networks).

Web Service is mainly used in two ways i.e. *Web Service Exposition and Web Service Consumption*. During Web Service Exposition, incorporated services are meant to be shared with external web servers and user can access these services in the form of exposed methods. Whereas Web Service Consumption is the process of using the services at client side by implementing proxy (send calls to the web server and reduces communication over head) using WSDL. To make Web Services more autonomous, intentional communication agents (software which is responsible for performing the autonomous behavior is certain environment) are also involved in it, using Foundation of Intelligent Physical Agents (FIPA) protocols.

Web Service is requested by some destination from service provider. The request sending and data receiving procedure is mainly divided in to four iterative steps, as shown in Figure 2, i.e. *Initiate to each other by IDs, Agreement to common WSDL, Input service description & semantic and SOAP messages exchange*. Initiate to each other by IDs; the first step to connect and start resource sharing (between Web Service provider and requester), by showing valid identification numbers (IDs). Agreement to common WSDL is the service (defining) contract declaration for licensed and legal resource sharing. Input service description & semantic is consists of the actual meaningful shared resource or information (can be encrypted). SOAP messages exchange is the communication protocol (earlier discussed).

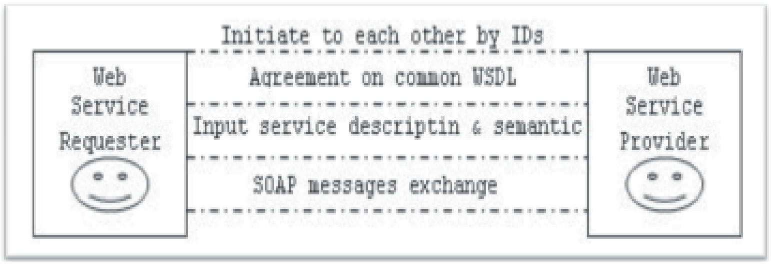


Figure 2. Web Service Communication Architecture

Figure Legend. Web Service Communication Architecture consists of two main identities i.e. *Web Service Requester and Web Service Provider*, with four intermediate communication messages.

One of the essential needs of today's web applications is also to provide security against broker including policies to secure distributed web applications, valid user authentication, control access to the defined roles, reliable data integration and data confidentiality. Web Service provides security in following ways i.e.

1. Implements SOAP for user authentication, data protection and message confidentiality.

2. Provides cryptographic token system for secure communication with the inclusion of some good mathematical algorithms e.g. *Kerberos, Key management for encryption, authorizing, signing and verification* etc.

3. Uses Security Assertion Markup Language (SAML) to manage user sessions (time period after user in login) by implementing three components i.e. *Assertion, Protocol and Binding*. Assertion is used in user authentication process, Protocol is used for sending and receiving data and Binding is used for the SOAP message exchange.

4. Writes access control rules and policies using eXtensible Access Control Markup Language (XACML) for different kind of applications.

5. Create, discover and update private and public identities using Identity Web Services Framework (ID-WSF).

3. Grid Computing

Grid computing is a middleware technology implementing the concepts of virtualization to heterogeneous entities to provide distributed infrastructure by increasing the computational power, capabilities, storage capacity and network speed [1].

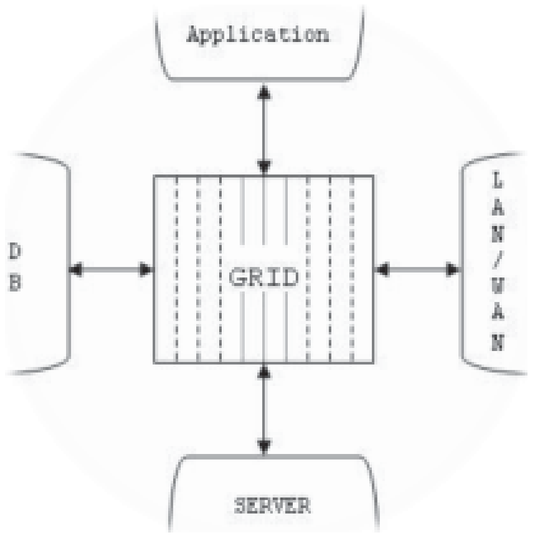


Figure 3. Grid Computing Conceptual Architecture

Figure Legend. Grid Computing Conceptual Architecture, based on four centrally connected units i.e. *Application (itself), LAN/WAN (network), DB (database) and Server*.

The ongoing research and development in the grid computing has been stated since 1990s for the advancement of science and engineering technologies. Grid Computing promotes distributed system development allowing dynamic resource sharing at run time, depending upon the availability of resources. As shown in figure 3, the basic grid computing conceptual architecture is consists of four integrated units i.e. *Application, LAN/WAB, DB and Server*. Application is the web based platform independent software program using Grid computing services, LAN (local area network) / WAN (wireless area network) is the connected network, DB (database) is the attached repository and Server is the computer program running to serve the requests of other programs (client applications). This architecture helps in to enhancing the concept of resource distribution by enabling different technologies capable of performing their tasks in a perfect way by decreasing the time limits, increasing resource sharing and synchronizing communication in an abstract way.

Practically the Grid is the combination of various resources from assorted companies. Main components of the Grid are Cluster of computer, Supercomputers, Internet and Job schedules. It uses general purpose protocols and interfaces for the communication. As shown in figure 4, Grid layer architecture is divided in to three layers i.e. *Layer 1; Application, Layer 2; Data and Commutation Protocols, and Layer 3; Routines and resources*. Starting with layer 3; it is the low level layer of Grid containing the operating system (OS), OS libraries, routines and network resources. Second is the Middle layer which contains the actual data, security mechanisms (SM), processes and languages (Compilers/Interpreters). Third and the last layer is the actual web application.

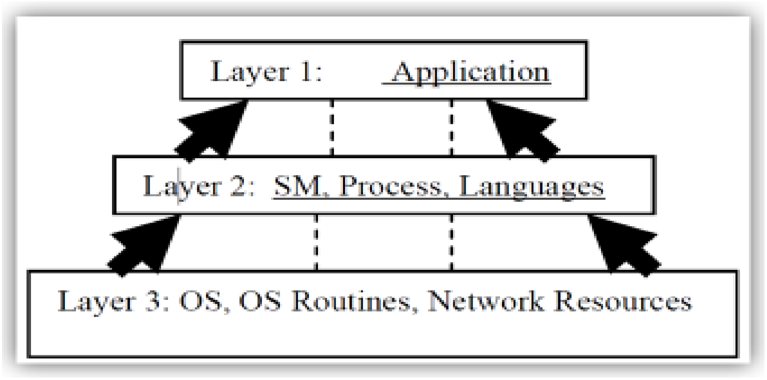

Layer 1: Application
Layer 2: SM, Process, Languages
Layer 3: OS, OS Routines, Network Resources

Figure 4. Grid Layer Architecture

Figure Legend. Grid Layer Architecture consisting of three main Layers i.e. *Application layer*, *Process layer* (SM, Processes and Languages) and *Operational layer* (Operating System, OS Routines and Network Resources)

Presented layer architecture in Figure 4 provide several benefits to the Grid computing web applications, as at third layer operating system runs and provides all its functionalities based on the its libraries and routines along with the provision of network resources initialization and maintenance of communications over the network. Then control moves from third layer to second where actual data is managed, languages along with their compilers or interpreters are installed to process web application's preprocessed source code and security mechanisms are implemented and handled. At the first layer, actual web application runs, for which the whole layered architecture is created.

One very important feature of Grid Computing is the provision of a reliable security mechanism with the capabilities of performing valid user authentication, secure communication and secure resource sharing. To provide these aforementioned security features, Grid implement the concepts of Cryptography (data encryption, data decryption, issuing of public keys, private keys and the certificates). Due to the implementation of this security mechanism, Grid Computing is considered as a secure data transfer technology renowned as *Asymmetric Grid Computing*.

To process semantic based data using Grid Computing, a new concept is introduced i.e. *Grid Semantic*; an approach towards information (Meta data) processing. Furthermore it helps in the establishment of connection in an automated way to provide quick, reliable and efficient resource sharing mechanism by using or creating virtual organizations, by making an intelligent network aware of all its network (free and in use) components and the network addresses (domain).

4. Open Service Grid Infrastructure

Open Service Grid Infrastructure (OSGI) is the enhanced version of Grid Computing promoting the concept of virtualization by managing the services and resources for distributed applications based on heterogeneous entities [5]. OSGI plugs the concept of service oriented architecture in Grid, which leads to the concept of combination of service oriented architecture and the Grid architecture i.e. *Open Service Grid Architecture (OSGA)*; able to manage and share the network resources, database (relations) and web servers. OSGA provides ubiquitous access to the systems and manages different kinds of work loads by reducing the cost and increasing resource utilization. Furthermore to have more ubiquitous access OSGA provides the implementation of Grid Computing with the integration of web services, security module, Ontology on Grid, virtual catalo access, logging and session maintenance and configuration management, so called Grid Service.

It is a service based platform independent distributed system model with key feature of Grid computing i.e. *Grid Service*, with the ability of efficiently locating servers and providing access to data in more consistent and integrated way. Furthermore it is able to implement the concept of virtualization, abstraction, searching entities and system monitoring mechanisms. Grid Web

Service Description Language (GWSDL: extended version of WSDL) is mainly used for OSGA development. OSGA provides well secure environment including valid user authentication process, integration, resource access, cross domain interactions, isolation, delegation and operations monitoring along with the implementation of optimized behavior at both the ends (client and server) by improving the quality of services. Technically speaking, the mechanism to resolve Grid Service handles Grid Service Reference renown as *Handle Resolver*. The grid service created at client's request for the instance of Grid service for a particular life time period and when that time finishes, instance is automatically destroyed (instance can also explicitly be destroyed). To have a secure interaction between the client's object and the requested service object, an abstract communication protocol is followed.

5. Web Grid Service

Concluding aforementioned research about Middleware Technologies, we can say that the current concept of Grid Services (OSGI and OSGA) is more successful because of the inclusion of some of the features of both Web Service and Grid Computing. Combining these two middleware technologies, we propose a new concept for the implementation of distributed online (web based) technology, including all behaviors of Grid Computing in a Web Service i.e. *Web Grid Service*, as shown in Figure 5.

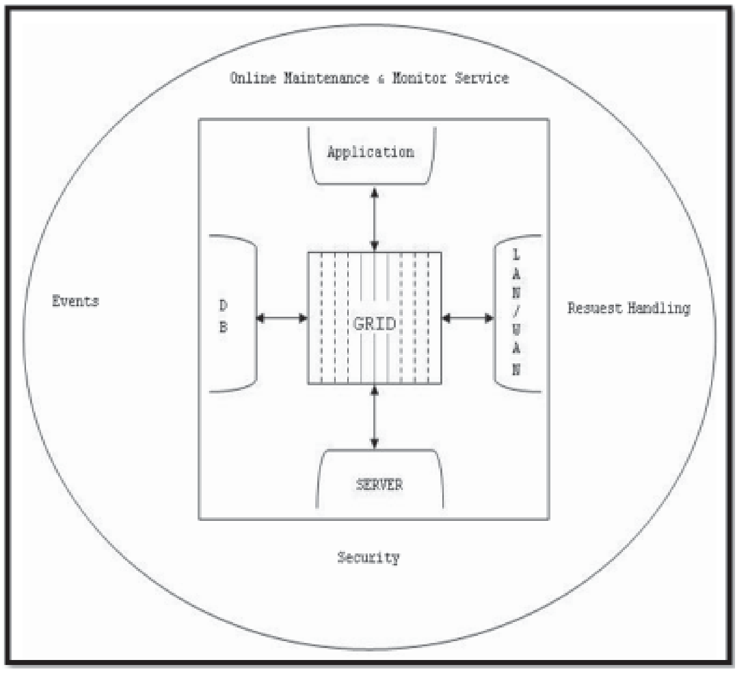


Figure 5. Web Grid Service; Conceptual Architecture

Figure Legend. Web Grid Service Conceptual Architecture consists of four main interconnected independent modules providing different services i.e. Application, Local/Wireless Area Network, Database and Server.

The conceptual architecture of Web Grid Service is based on the joint capabilities of Grid Computing and the Web Services, implementing three layered architecture with the use of RPCs over HTTP for the communication, preferring SOAP for communication purposes with the use of GWSDL. The conceptual architecture of Web Grid Service claims of benefiting distributed web systems with implementation of virtual environment between the heterogeneous entities, efficient resources sharing, cryptographic security towards data confidentiality, defined role access to users, policy implementation and reliable messaging in possible limited time and cost. Furthermore the inclusion of multi agent system (MAS) can bring much autonomous behavior to the Web Grid Service, as multiple agents can be involved responsible of meeting several tasks individually.

Extending the conceptual architecture of Web Grid Service, we also propose a chain of Web Grid Services (like a network) as shown in Figure 6, where more than one Web Grid Services are linked to each other in the form of a network with the implementation of MAS in it. The proposed and extended concept of Web Grid Services can be implemented with the use of existing Middleware tools and technologies for the development of distributed, service oriented and enterprise applications.

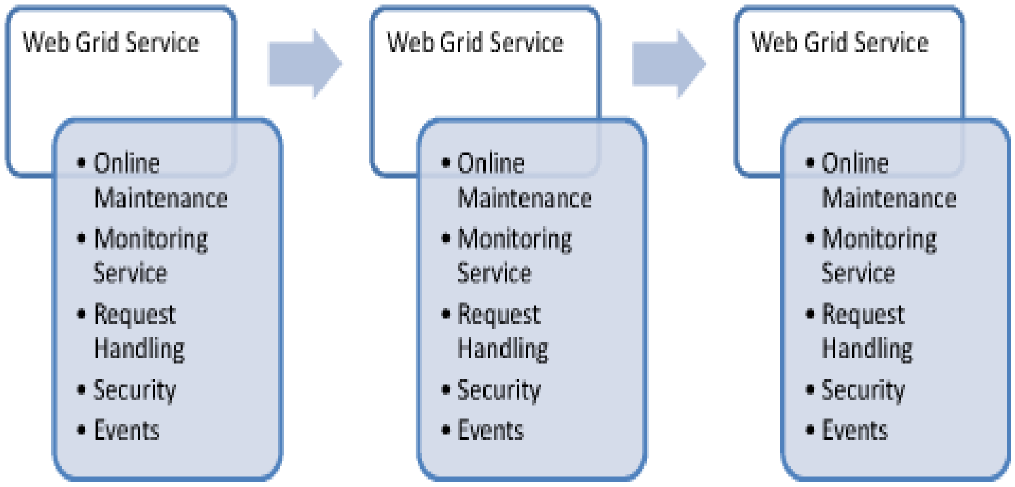


Figure 6. Chain Web Grid Service Conceptual Architecture

Figure Legend. Chain Web Grid Service Conceptual Architecture; a network of Web Grid Services.

6. Conclusions

A review research has been conducted in the field of Middleware Technology and its importance has been elaborated in detail in this paper. Concluding the discussion, in this paper, we have presented three major building blocks of Middleware Technologies i.e. *Web Service, GRID Computing, Open Service Grid Infrastructure*, some implementation tools and technologies i.e. *Web Services, CORBA, Grid Computing, OSGI, SNMP*, and some communication protocols i.e. *RPC, HTTP, WSDL, SOAP and GWSDL*. We have also introduced a new concept i.e. *Web Grid Service*, and briefly presented its conceptual architecture, we have shortly described how this concept can be extentended and implemented in the form of network sequence i.e. Chain Web Grid Service, to take advantage in the implementation for extreme, secure and efficient heavy (amount of) data exchange and resource sharing amongst different web applications and client based desktop applications. In future the conceptual architecture of Web Grid Service and Chain Web Grid Services can be implemented using available Middleware implementation supporting tools and technologies, and its effectiveness and real time potential can be tested as well.

7. Acknowledgement

We (authors) would like to thank to academic administration and research supervisors belonging to the University of Wuerzburg (UW) and Blekinge Institute of Technology (BTH) Sweden. We would like to pay out special gratitude to Prof. Dr. Martin Fredriksson from BTH for his guidance during this research and Prof. Dr. Thomas Dandekar from UW for his generous support and help.

References

[1] Wang, GuiLing., Li, YuShun., Yang, ShengWen,, Miao. ChunYu,, Xu. Jun,, Shi. MeiLin, (2005). Service-oriented grid architecture and middleware technologies for collaborative e-learning, *In*: IEEE International Conference on Services Computing, 2, 67 - 74, 21 November.
[2] Scott Walker, Alan Dearle, Stuart Norcross, Graham Kirby, Andrew McCarthy, (2006). RAFDA: A Policy-Aware Middleware Supporting the Flexible Separation of Application Logic from Distribution. EuroSys 2006. University of St Andrews CS/06/2.
[3] Khan Sumair, Qureshi Kalim, Rashid Haroon. Performance Comparison of ICE, HORB, CORBA and Dot NET Remoting Middleware Technologies. *International Journal of Computer Applications*. 3 (1) 15.
[4] Martin Kuehnhausen, Victor S. Frost, (2010). Framework for Analyzing SOAP Messages in Web Service Environments. 5 (1) 1-9.
[5] Grimshaw, A., Morgan, M., Merrill, D., Kishimoto, H., Savva, A., Snelling, D., Smith, C., Berry, D. (2009). An Open Grid Services Architecture Primer, Computer, Vol 42 (2).

[6] Andrew, D., Birrell, Bruce Jay Nelson, (1984). Implementing Remote Procedure Calls. *ACM Transactions on Computer Systems,* 2 (1) 39-59, February.

[7] Brian Travis, Mae Ozkan, (2002). Simple Object Access Protocol (SOAP), Chapter 7, Web Services Implementation Guide, Architag Press.

[8] Phillip, J., Windley. Enabling Web Services, <http://www.windley.com/>

[9]Phil Wainewright. Web Services Infrastructure, The global utility for real-time business, Pages 20, <http://www.philwainewright.com/pubs/wp/WSIpaper.pdf>

[10] Corba primer, A., <http://www.omg.org/news/whitepapers/seguecorba.pdf>

[11] Sim, A., Nordberg, H., Bernardo, L. M., Shoshani, A. , Rotem, D. (1999). Storage Access Coordination Using CORBA. *In:* Proceedings of the International Symposium on Distributed Objects and Applications (DOA '99). IEEE Computer Society, Washington, DC, USA, 168.

[12] Parashar, M. and Lee, C. (2005). Grid Computing - Introduction and Overview. Proceedings of the IEEE, Special Issue on Grid Computing. IEEE Press, 93 (3) March.

[13] Parashar, M. and Lee, C. (2005). Grid Computing. An Evolving Vision., *In*: Proceedings of the IEEE, Special Issue on Grid Computing. IEEE Press, 93 (3) March.

[14] About the OSGi Service Platform, Technical Whitepaper, Revision 4.1, OSGi Alliance, 7 June (2007).

[15] Murray, Peter., Stalvi, Paul (2005). SNMP: Simplified, F5 White Paper, <http://www.f5.com/pdf/white-papers/snmp-simplified-wp.pdf>

[16] Qilin Li, Mingtian Zhou, China, P. R. (2010). The Future-Oriented Middleware Technology, *Journal of Commuters*, 5 (2) 250-257, February.

[17] Tha, K., Elmasri, R. (2011). IRTG: A Grid Middleware for Bioinformatics, *In:* IEEE International Conference on Services Computing (SCC), 440-447, 4-9 July.

[18] Hajo, N., Krabbenhöft, Steffen Möller, Daniel Bayer, (2008). Integrating ARC grid middleware with Taverna workflows, Bioinformatics, 24 (9) 1221-1222.

[19] Kyu, Cheol Cho., Jong, Sik Lee, (2011), Grid-Based and Outlier Detection-Based Data Clustering and Classification, Ubiquitous Computing and Multimedia Applications, *Communications in Computer and Information Science*, 150,129-138

[20] Denault, Alexandre., Kienzle, Jörg., Dionne,Carl., Verbrugge, Clark (2008). Object oriented Network Middleware for Massively Multiplayer Online Games. Technical Report SOCS-TR-2008.5, School of Computer Science, McGill University, December.

[21] Denault, A., Kienzle, J. (2011). Journey: A Massively Multiplayer Online Game Middleware. *Software IEEE*, 28 (5) 38-44, October.

[22] Tsun-Yu Hsiao, Shyan-Ming Yuan, (2005). Practical middleware for massively multiplayer online games, *IEEE Internet Computing,* 9 (5) 47-54, October.

[23] Ahmed, Z., Gerhard, D. (2007). Contributions of PDM Systems in Organizational Technical Data Management. *In:* The proceedings of The First IEEE International Conference On Computer, Control & Communication (IEEE-IC4 2007), 12-13 November.

[24] Huang, M. Y., Lin, Y. J., Hu Xu, (2004). A framework for web-based product data management using J2EE, *International Journal of Advanced Manufacturing Technology*, 24 (11-12) 847-852.

[25] Ahmed, Z., Saman, M. (2011). Machine Learning and Data Optimization using BPNN and GA in DOC, *International Journal of Emerging Sciences*, 1 (2) 108-119, June.

[26] Eric Wohlstadter, Stefan Tai, Thomas Mikalsen, Isabelle Rouvellou, and Prem Devanbu, (2004). GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions, *In:* Proceedings 26th International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, UK, May.

[27] Tai, Stefan., Mikalsen, Thomas., Rouvellou, Isabelle(2003). Using Message-Oriented Middleware for Reliable Web Services Messaging. *In:* Proceedings of WES 2003, Klagenfurt, Austria, June.

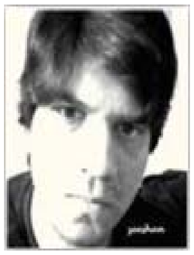
[28] Mikalsen, Thomas., Tai, Stefan., Rouvellou, Isabelle (2002). Transactional Attitudes: Reliable Composition of Autonomous Web Services. *In:* Workshop on Dependable Middleware-based Systems (WDMS 2002), part of the International Conference on Dependable Systems and Networks (DSN 2002), Washington D.C., June.

[29]Ahmed, Z. (2010). Towards Performance Measurement and Metrics based Analysis of PLA Applications, *International Journal of Software Engineering & Applications*, 1 (3) 66-80.

[30] Ahmed, Z., Saman, M. (2010). Towards Increase in Quality by Preprocessed Source Code and Measurement Analysis of Software Applications, International Science & Technology Transactions on Information Technology- Theory and Applications, 1 (1).8-13, September.

[31] Saman, M. (2011). Towards the Contribution of NEMDBs in Global Genetic Heterogeneity, *International Journal of Emerging Sciences* (IJES), 1 (3) 433-443, September.
[32] Silva de Oliveira, D. J., Rosa, N. S. (2010). Evaluating Product Line Architecture for Grid Computing Middleware Systems: Ubá Experience, IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 257-262, 20-23 April.
[33] Sven Apel, Klemens Böhm, (2005). Towards the Development of Ubiquitous Middleware Product Lines, *In:* ASE'04 SEM Workshop, V. 3437.

Author Biography



Zeeshan Ahmed is the software engineer, researcher at the Department of Bioinformatics, Biocenter, Am Hubland, University of Wuerzburg Germany. His interests include developing intelligent software systems towards product data analysis and visualization management.

Saman Majeed is the doctoral scientist in the Department of Bioinformatics, Biocenter, Am Hubland, University of Wuerzburg, Germany.