Analyzing the Web Services and UniFrame Paradigms[1]

Natasha Gupta[2] Rajeev R. Raje[2] Andrew Olson[2] Barrett R. Bryant[3] Mikhail Auguston[4] Carol Burt[3]

Abstract

The software realization of today's distributed systems often require combining of heterogeneous software components, each offering a specialized service. This heterogeneity necessitates a paradigm for the interoperation of different components. Various models and approaches have been proposed to facilitate a smooth interoperation. Web Services and UniFrame are two such paradigms. This paper presents analyses of these two alternatives, thereby, indicating their similarities and differences.

1 Introduction

The evolution in the field of computing has shifted its paradigm from a centralized one to a distributed one. Hence, the target environment is no more a centrally managed, but concerned with collaboration, data sharing, and other new modes of interactions involving distributed resources. This necessitates the availability of technologies and solutions that can effectively and efficiently integrate services across disparate systems. This integration can be challenging because of the need to achieve various qualities of services when running on top of different native platforms [1]. Innovations in this field have led to developments of many paradigms including Web Services (WS's) [2], and UniFrame [3]. Each of these approaches has associated pros and cons. Web Services have emerged as a new "Web Development Tool" which enables a web application to become more interactive, by providing means to make it communicate at the middle-tier lever (business logic level) and provide a new platform to build software for a distributed environment. UniFrame is a research project that aims to provide a framework that allows a seamless interoperation of heterogeneous components. The purpose of this paper is to compare and contrast the Web Services framework and the UniFrame.

2 Related Work

2.1.1 Enterprise Application Integration (EAI) Solutions

The EAI [4] solutions provide the infrastructures for an organization that take the integration technology from the traditional point-to-point connections to a level that links multiple applications and databases internal to the organization to share information and business processes. EAI typically uses middleware to connect to different applications. A custom interface is built to link each separate application in the EAI system. Most EAI systems use adaptors to connect applications. Several types of EAI exist, including data integration, business process integration and method integration. However, the integration that EAI solutions provide tends to be complex and expensive, despite improving the overall communication. In addition, the EAI interfaces are not reusable and cannot be used by a company to connect to their business partners whose applications fall outside the boundaries of the organization. Web Services overcome this limitation by providing a set of reusable interfaces to applications, which enables them to interoperate with any other application (Web Service) using SOAP.

2.1.2 Business-To-Business (B2B) Solutions

The Internet has given birth to a "digital economy" [5]. In such an economy, B2B e-commerce provides a company with an effective and efficient end-to-end process communication to buy and sell services in an economical way. B2B relationships are often characterized by stringent requirements for security, auditability, availability, service level agreements and complex transaction processing flows [1] in addition to the large technical differences that arise between different organizations. B2B Integration has long been accomplished with the use of technologies like Enterprise Data Interchange (EDI). EDI is a relatively arcane technology that requires substantial overhead on the

[1] This material is based upon the work supported by the U.S. Office of Naval Research under award number N0014-01-1-0746.
[2] Department of Computer and Information Science, Indiana University Purdue University Indianapolis, 723 W. Michigan St., SL280, Indianapolis, IN 46202, USA, {nsgupta, rraje, aolson}@cs.iupui.edu.
[3] Department of Computer and Information Sciences, University of Alabama at Birmingham, 1300 University Blvd., CH 115A, Birmingham, AL 35294, USA, {bryant, cburt}@cis.uab.edu.
[4] Computer Science Department, Naval Postgraduate School, 833 Dyer Rd., SP517, Monterey, CA 93943, USA, maugusto@nps.navy.mil.

part of the participants, and a clear understanding of the semantics of the messages exchanged. EDI implementations, despite their “standardized” nature vary dramatically from business to business [6].

2.1.3 Open Grid Services Architecture (OGSA) for Distributed Systems Integration

OGSA builds upon the concepts and technologies from the Grid and Web Services communities. It [1] defines standard mechanisms for creating, naming, and discovering transient Grid service instances; provides location transparency and multiple protocol bindings for service instances; and supports integration with underlying native platform facilities. It aligns the Grid technologies with the WS technologies, in particular the WSDL, to provide mechanisms required for creating and composing sophisticated distributed systems, including lifetime management, change management, and notification. OGSA has adopted Globus Toolkit as the underlying Grid technology solution.

Each of these above mentioned approaches have specific objectives and are, aimed typically at particular application domains. In the next section, two other approaches, Web Services and UniFrame that are generic in nature are discussed.

3 The UniFrame and Web Services Nexus

3.1 UniFrame Overview

The main focus of UniFrame is to provide a comprehensive framework for the software realization of distributed computing systems. It consists of (a) a meta-model for components and associated hierarchical set-up for indicating contracts and constraints of the component, (b) an automatic generation of glue and wrappers, based on designer’s specifications to achieve interoperability, (c) a formal mechanism for precisely describing the meta-model, and (d) the formalization of the notion of the quality of service of each component and an ensemble of components.

3.2 Web Services (WS) Overview

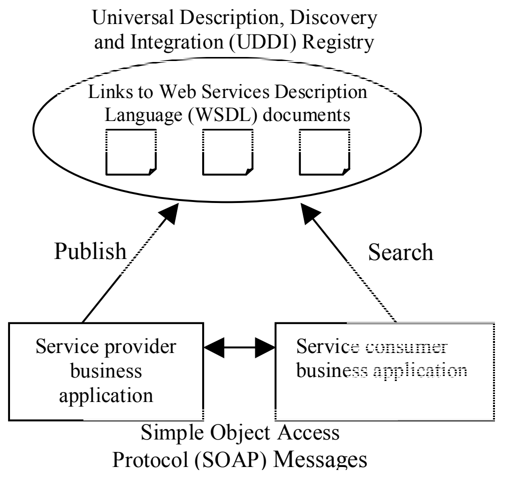
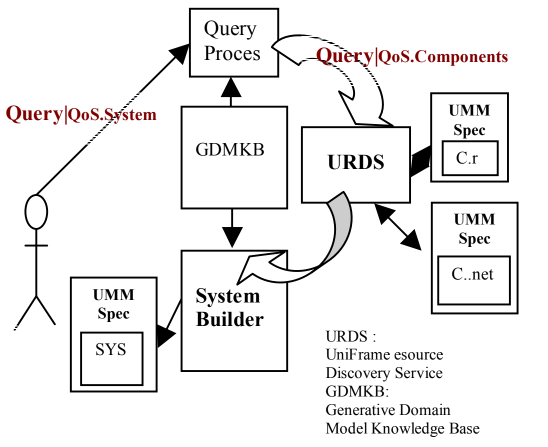
WS are based on existing protocols and technologies and provide a greater flexibility with respect to the interoperability, the reuse and the development of applications in a distributed environment. The underlying idea behind WS is to promote the “software as a service” paradigm. The use of open standards enables interoperability between components. These standards are based on XML, which enables WS to communicate with other applications in a programming language-, programming model-, and system software-neutral manner. XML forms the basis of the three standards: SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) and UDDI (Universal Description, Discovery and Integration) [5].

The next section indicates a comparison between the architectural aspects of the two frameworks, i.e., UniFrame and Web Services, and then the section 4 describes a model-based comparison.

3.3 Architectural Comparison

The following table shows the architectural comparison between the two paradigms:

	WEB SERVICES FRAMEWORK	**UNIFRAME**
OBJECTIVE	To provide a set of related standards which allow building of dynamic, loosely coupled systems composed of services, not bounded to any implementation and can be published, described, located and invoked over a network, more generally World Wide Web	To create a comprehensive framework that unifies the existing and emerging distributed component/service models under a common meta-model that enables the discovery, interoperability, and collaboration of components via generative software techniques [3,7]

GENERAL ARCHITECTURE	 Universal Description, Discovery and Integration (UDDI) Registry Links to Web Services Description Language (WSDL) documents Publish Search Service provider business application Service consumer business application Simple Object Access Protocol (SOAP) Messages	 Query Proces Query|QoS.Components Query|QoS.System GDMKB URDS UMM Spec C.r UMM Spec C..net UMM Spec SYS System Builder URDS : UniFrame esource Discovery Service GDMKB: Generative Domain Model Knowledge Base
BASIC TASKS/ PROCEDURES INVOLVED	▪ Service Development and Deployment (leveraging different platforms to one standard of web services using different Web Services development tools and software provided by vendors) ▪ Formal description of services (WSDL) ▪ Registration of services with UDDI (publish) ▪ Discovery of services (Find) ▪ Binding with the Service (Bind)	▪ Developing components using different current and future object models, such as, Java-RMI/CORBA/.Net/Web Services ▪ Informal and formal UMM (Unified Meta-Component Model) specifications of each component ▪ Querying the UniFrame for the system with desired Quality of Service parameters ▪ Creation of an integrated system made out of discovered components ▪ Incorporation of necessary glue and wrappers for QoS measurements and interoperation ▪ Checking to see if the test results of the integrated system satisfy criteria or not ▪ Refine Query or select alternate components to re-build and retest the integrated system
SERVICE/ COMPONENT DEVELOPMET DEPLOYMENT	▪ Development using frameworks that support them (e.g. .NET) or using different object models, which are then leveraged as services using the toolkits that support the technology ▪ Registering Services with the UDDI public/private registry	▪ Components are developed using different standard object models ▪ Deployment also under the same model with extra infrastructure provided by UniFrame to support seamless interoperation and system generation
DESCRIPTION OF SERVICES/ COMPONENTS	Web Service Description Language Document (WSDL file – XML)	UniFrame Meta-Component Model Description (UMM Specifications – informal text and XML)
DISCOVERY	Discovery through the UDDI Business or private registries (static registries)	Discovery through an search process involving active entities – headhunters and active registries [UniFrame Resource Discovery Service (URDS) Framework]
INTER-OPERABILITY OF SERVICES/COMP ONENTS	XML (standard for data exchange) and SOAP (Simple Object Access Protocol)	Automatic generation of glues and wrappers
SYSTEM INTEGRATION	▪ A hand-crafted approach wherein the responsibility of integration lies with the application developer by means of APIs of the WS ▪ Need to incorporate WS interfaces and	A comprehensive model-based approach forms the backbone of the system integration process right from the initial stages. The model follows an architecture-centric, domain-based and a technology-independent approach. The process

	integration capabilities within the existing "application integrating" tools and products	may be manual, completely automatic or a mix of both
RELIABILITY OF THE COMPOSED SYSTEM	Reliance on a third party (Web Service Auditors) which guarantees the reliability of a web service on basis of testing and certification during its creation as well as operational stage	Reliability based on test cases and formalism and a strong mathematical foundation of event traces and two level grammar
ADVANTAGES **ADVANTAGES (Contd..)**	▪ Builds upon open text-based standards (XML), thus aiding in interoperability ▪ Less additional cost involved in adoption, since employs existing infrastructure (Internet) and applications can be repackaged as Web Services	▪ No requirement of additional software tool to build components ▪ Automatic generation of glues and wrappers ▪ Quality of Service validation and assurance through event traces and formal domain knowledge; backed by a mathematical foundation ▪ Use of aspect-oriented programming to weave in the notion of QoS into the framework distinguishes UniFrame ▪ Active search process involving the notion of "headhunters"
LIMITATIONS	▪ Relatively new; standardization in progress, hence, Web Services created with current tools will not be compatible with the future technologies ▪ Use of text-based standards, XML, for communication may affect performance in some critical applications ▪ No standardized methods devised for assuring and validating Quality of Service; Use of third party "web service auditors"	▪ No standardization reached yet ▪ Experimentation and performance evaluation at a large scale and in a realistic domain not complete

Some of the important points tabulated above are described in detail in the next few sections.

3.3.1 Discovery Services

Web Services Discovery Process: The term discovery refers to the process of locating "Web Services" by means of registries. This process is carried out by businesses searching for services offering specific functionalities. WS Registries and Brokerages facilitate the discovery process and enable interactions between the service providers and requesters. The discovery process is classified into two categories [4]:

- Direct Discovery: This involves obtaining data from a registry, which is maintained by the service provider itself.
- Indirect Discovery: This involves obtaining data about a Web Service from a registry, which is maintained by a third party.

A service provider publishes the WSDL document containing the description of its Web Service, with the UDDI, which makes locations of such WSDL files available to a service requester. The Service Requester searches the UDDI based on certain criterion, such as functionality or a Quality of Service (QoS) attribute. Once it discovers a service, meeting its needs, it knows the method of accessing the Web Service by means of the WSDL file. It can now communicate with the Web Service directly via SOAP messages.

There are a few other discovery technologies, which support the discovery of Web Services apart from the UDDI specifications – ebXML and WS-Inspection for example. A Service developer/organization can combine these technologies with UDDI in order to take advantage of the features of both. For example, UDDI currently does not support a security model whereas ebXML does and so an organization can advertise its services through UDDI, on the other hand store its trading agreements and contracts through ebXML.

UniFrame Resource Discovery Service (URDS) Framework: The URDS architecture [8] provides a mechanism for an automated discovery and the selection of components meeting necessary QoS requirements. URDS is designed to act as a Discovery Service wherein new services are dynamically discovered while providing clients with a directory style access to services. The discovery process in URDS is "administratively scoped", i.e., it locates services within an administratively defined logical domain – in UniFrame a domain refers to industry specific markets such as Financial Services, Medical domain and Manufacturing Services, etc. The URDS infrastructure consists of two parts: (a) the Internet Component Broker (ICB) and (b) Headhunters.
The ICB, in addition to performing the functions of a conventional broker, also ensures the authentication of the principals of the system (Headhunters and Active Registries); cooperates with other ICB's deployed on the network to provide matchmaking between service producers and consumers; and acts as a mediator between two components adhering to different component models. A Headhunter is equivalent to a binder or trader in other models. However, unlike the trader, here the onus of registering components lies with the headhunter and not on the components themselves. Hence, the headhunter is capable of detecting the presence of service providers on the network, register the functionality of these service providers and return a list of service providers, which matches the requirements of the consumer requests forwarded by the Query Manager, to the ICB. The services are discovered by means of Active Registries (discussed later), with which the services are registered. The discovery process employed could vary from standard search techniques to broadcasts and multicasts to specific machines.

3.3.2 Service Descriptions

Web Service Description Language (WSDL) Document: It is an XML document for describing WS as a set of endpoints operating on messages containing either document-oriented (messaging) or RPC-payloads. Service interfaces are defined abstractly in terms of message structures and sequences of simple message exchanges and then bound to a concrete network protocol and data-encoding format to define an end-point. Related concrete end-points are bundled to define abstract end-points (services). The WSDL is extensible to allow description of end-points and the concrete representation of their messages for a variety of different message formats and network protocols [4].
UniFrame Meta-Component Model (UMM) Description: In UniFrame, components are autonomous entities. The UniFrame description of a component is more comprehensive and specified in a natural language-like manner. It indicates the functional (i.e., computational, cooperative and auxiliary aspects) and non-functional (i.e., QoS constraints) features of the component. These specifications are then refined into a formal specification based on the theory of Two-Level Grammar (TLG) and natural language specifications [9]. TLG specifications allow for a multi-level interface for the component. These levels are: Syntactic, Behavioral, Synchronization and QoS.

3.3.3 Registries/Repositories

Web Services Registries: The Web Services framework supports two kinds of repositories - UDDI and WS Brokerages.
UDDI: The UDDI standardization provides for "searchable Web Services Registries" which facilitate the storage, discovery and exchange of information about businesses and their Web Services. UDDI is implemented in two forms:
UDDI Business Registry: publicly accessible and maintained by Microsoft, IBM, Hewlett Packard and SAP.
UDDI Private Registry: accessible only to authorized users.
The various entities involved during the utilization of UBR (UDDI Business Registry) [4] are:

Operator Nodes: The organizations that host the implementation of the UDDI Business Registry are Microsoft, IBM, SAP, and Hewlett Packard. UBR operates on the principle of "register once and publish everywhere". This in turn implies a replication of the data within the operator nodes so that all instances of records are identical with each node. Operator nodes synchronize their information at least every 12 hours.

Custodian: The custodian for a company is the operator node with which it publishes its web services. A company can register and update its information only through its custodian. This prevents multiple versions of the data from entering in the four different operator nodes.

Registrar: These organizations do not host implementations of the UDDI but act as assistants for organizations in creating data (such as business and service descriptions) and publishing in the UBR.

Structure and Information Model of UDDI: XML forms the basis of the overall information structure of UDDI which can be broadly divided into following information levels:

White Pages: General information about the provider, such as its name, contact information and identifiers.

Yellow Pages: Categorization of the providers' information based on their services.

Green Pages: Technical information about the provider's services or products. Usually contains references to the WSDL documents of the services enabling the client to know as to how to interact with the Web Service.

UDDI supports certain APIs for the clients to use the registry. These include:
Publishing API – It supports the publish operation on the UDDI Registry. The access to this API is restricted to authorized users only. Operator nodes implement a form of Authentication protocol to allow legal organizations to access this API. By means of publishing API, an organization is able to execute commands to create and update information in its operator node.
Inquiry API: Supports the find operation in three different patters (browse, drill-down and invocation). This API is accessible to any individual on the UBR who wishes to locate a service or a kind of service.

WS Brokerages: The WS brokerages are web sites that house information about the available WS in the form of a list, along with their web addresses. These brokerages can also supply additional services, which can include advanced search capabilities based on category, organization name or schema type, service monitoring and service support, which can include services-related resources such as a tool that validates WSDL documents. Examples of some of the current Web Services Brokerages are: Allesta Web Service Agency, SalCentral Service, Xmethods and serviceFORGE.

UniFrame Registries: In the case of UniFrame, the entity that houses the information about components developed using a particular model is local to that component model. This entity is named "Active Registry", and is an enhanced version of the native registry of the corresponding object model. It has features such as *Activeness (*an ability to listen to multicast messages), *Introspection and a Capability to detect failures of the Headhunters.*

The conceptual difference that exists between registries of the two frameworks is in the way the registries participate in the discovery process of the components. In the case of the WS framework, the onus of locating components lies in the hands of the service requesters. While in UniFrame, the emphasis is on the automated discovery process provided by means of the URDS. Whether an organization needs to deploy one active registry per machine or one per many, is not decided and could vary depending on the size and necessity of the organization. While a service requester and publisher has to confirm to the underlying implementation of the UDDI registry as preferred by the company hosting it, the Active Registry is not as rigid and constraint since it builds upon the same native technology used for the development of components registered with it.

3.3.4 Quality of Service Assurances

Quality of Service Assurances in Web Services: Currently, service providers typically employ third parties to audit their web services during the creation stage as well as for reevaluation of the service on regular basis. An *auditor* achieves this in the form of testing and certification. Auditors may also be employed by the service requestors in order to gain a kind of guarantee about the level of service offered by the Web Service. The entire scenario employs "Service Level Agreements (SLA)" [4]. These are "legal contracts in which a service provider outlines the level of service it guarantees for a specific Web Service". When customers purchase the Web services subscription, they receive the services according to the quality-related contents specified by the SLAs. The service developer may maintain the SLAs. As the contents of the SLA are determined by the participating entities, there are no formal guidelines to specify the level of service a particular Web Service provides. The QoS requirements, which SLAs of WS's outline, include a*vailability, accessibility, integrity, performance, reliability, conformance to standards and security.*
Quality of Service framework of UniFrame: The approach followed by UniFrame can be stated as: building a precise model of the system's behavior (based on event traces) and then providing a programming formalism to describe the computations over these event traces. These are then applied in order to define different kinds of QoS metrics. UniFrame's iterative approach to system assembly from components meeting user's query specifications is based on constructive calculations of QoS metrics on representative set of test cases.
Quantifying the quality of service of the individual Commercial Off The Shelf (COTS) components, which compose to form an integrated system with a predictable quality, is one of the critical part of the UniFrame Approach. UniFrame provides a QoS Framework [5] for selecting, specifying and validating the QoS of components. The features of the UniFrame QoS framework are:

- An existence of a QoS catalog [10] containing detailed descriptions about QoS attributes, their classifications, their evaluation methodologies and the interrelationships with the other attributes.

- An integration of QoS at the individual component and distributed system levels.
- The validation and assurance of QoS, based on the concept of event grammars [11].
- An investigation of the effects of component composition on QoS; involving the estimation of the QoS of an ensemble of software components given the QoS of individual components.
- A QoS-centric iterative component-based software development process to ensure that the end product matches both the functional and QoS specifications.

UniFrame takes a domain-based approach in the classification and the discovery of components. Since every domain has its own constraints with respect to the QoS attributes, the QoS catalog aims to act as a checklist for any component developer/user interested in identifying and validating QoS attributes.

3.4 Model-based Comparison

- WS are all about XML and it being a text-based standard implies delays involved in parsing it, which may prove vital in performance-critical applications. XML uses two sets of redundant tags to mark up every piece of information it represents. The tags are usually written to be humanly readable, which makes the actual tags a lot longer than they need to be. Also, one character in a Unicode document can be up to four bytes. Four bytes in some other proprietary binary format used by technologies such as DCOM or RMI can hold a lot more information than just one character. The ability to serialize the data over a connection, parse it quickly and efficiently is what plays a vital role in applications interacting over the network [12]. UniFrame, on the other hand, leverages the components in a way so that they are a part of an application while remaining within their own object-model. This allows for more efficient ways of electronic communication.
- HTTP is the preeminent protocol to transfer WS content and is allowed a free access through firewalls. HTTP, although used almost everywhere because of its reliability and ubiquity, is also not the most efficient transport protocol [12]. HTTP relies on a constant connection between the client and server when a request is made. This constant connection causes an overhead in cases when the data that needs to be transferred is quite small. However, in the WS's universe, many transactions are essentially asynchronous. This in turn implies that the response of a web service request is not guaranteed. HTTP was not meant to deal with this kind of asynchronicity. It also relies on only one side initiating communication and the other side only responding to the request. This approach inhibits true peer-to-peer exchanges through Web services. A newer version of HTTP aims to fasten communications by making use of compression, but some of the previous issues still need to be pondered upon. Other protocols such as SMTP, over which Web Services can be implemented, still do not provide a major breakthrough in this respect. As UniFrame does not attach itself to a specific protocol, it avoids some of the drawbacks related to the usage of HTTP.
- The only guarantee that a service requester has about a Web Service is through its SLA. No other explicit mechanisms are mandatory in the WS world. Thus, the user of WS may or may not have a mechanism to validate the QoS claims made by the creator of WS. Hence, a requester can terminate its contract if the WS's fail to deliver what it claimed in its SLA. In contrast, UniFrame makes the notion of quality explicit during the creation of components. It also provides the user means (by the use of event grammars, glues and wrappers) to validate the QoS of any component made available by a supplier.
- In the world of B2B, Web Services prove to be a major benefit since they provide the needed flexibility and ability to operate across the Internet on completely disparate systems owned by completely independent entities. However, in EAI solutions, the major drivers are not only interoperability but also speed and efficiency, and with those requirements, Web services don't really seem to meet the need. Organizations globally are becoming aware of the importance and need of integration across disparate platforms. An organization with numerous applications needs EAI solution and corporations that are extending their processes with partners need B2B. The future holds potential for a solution set that provides the functionality for both the requirements frameworks. The UniFrame with its unbiased approach is an attempt in this direction.
- Although UDDI registries, both public and private, offer a great deal of advantage in terms of an application integration of the participating companies, they have their own set of limitations too. Firstly, because UDDI is fairly new, it has not reached standardization in a complete way, which holds true for UniFrame as well. Secondly, the UDDI Business Registry poses the question of data reliability. UniFrame does not involve the notion of publicly accessible registries. The Active Registries only allow authorized entities to publish components and interacts with the headhunter, thereby reducing the threats of data compromise. The discovery mechanism of the UMM Framework involves the headhunter storing the data about the components after it retrieves it from the Active Registries. The duration of the time interval after which this process repeats itself can be controlled so as to guarantee the freshness of the data within the meta-repository of the headhunters.

UDDI registries, although describe web services, do not evaluate them [4]. It does not house the Quality-of-Service information about a web service and requires an extensive search on the service-consumers part to do so. UniFrame on he other hand, provides an extensive Quality-of-Service framework to do so.

3.5 Integrating Web Services into UniFrame

As outlined above, the WS and UniFrame differ in their approaches and associated implementation techniques. However, they can complement each other to provide solutions for future distributed systems. UniFrame uses the Generative Domain Model [13] to describe the properties of domain-specific components and to elicit rules for assembling heterogeneous components. One possible approach to integrate WS in UniFrame could be to use WS as a mechanism to wrap heterogeneous components. Due to the open nature of WS, such an approach will ease the task of assembling heterogeneous components adhering to existing and new object models. Furthermore, since WS are weak in representing the business semantics of application domains, this will also lead to the enrichment of WS technology in terms of semantic representation by following a model driven approach for specific domain-specific component models. UniFrame can then automatically generate WSDL from the models with the help of generators.

4 Conclusion

Developing component-based software solutions for distributed systems is an inherently complex task. Any approach to tame these complexities must account for disparities that exist due to the existence of different object models. Web Services and UniFrame are two approaches that propose effective solutions for future component-based distributed systems. In this paper, an analysis of these two approaches has been presented. Although these two approaches differ from each other, they can also complement each other and provide a comprehensive solution for the creation of distributed systems. The proposed approach to integrate Web Services into UniFrame needs further investigation and is being currently explored.

5 References

[1] Foster, I., Kesselman, C., Nick, J., Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[2] World-Wide Web Consortium (W3C), "Web Services Activity", 2002, http://www.w3.org/2002/ws.

[3] Raje R., Bryant B., Auguston M., Olson A., Burt C., 2001, "A Unified Approach for Integration of Distributed Heterogeneous Software Components," Proceedings of the 2001 Monterey Workshop Engineering Automation for Software Intensive System Integration, pp. 109-119.

[4] Dietel, H., Dietel, P., DuWaldt, B., Trees, L., Web Services – A Technical Introduction, 2003, Prentice Hall, Upper Saddle River, New Jersey 07458

[5] Dhingra, V., "Business-to-Business Ecommerce," http://projects.bus.lsu.edu/independent_study/vdhing1/b2b.

[6] A Darwin Partners and ZapThink Insight, "Using Web Services for Integration", http://www.xml.org/xml/wsi.pdf

[7] Raje, R. R., Auguston, M., Bryant, B. R., Olson, A. M., Burt, C. C., 2002, A Quality of Service-based Framework for Creating Distributed Heterogeneous Software Components, *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1009-1034.

[8] Siram, N. N., Raje, R. R., Olson, A. M., Bryant, B. R., Burt, C. C., and Auguston, M., An Architecture for the UniFrame Resource Discovery Service, Proceedings of the 3rd Int. Workshop Software Engineering and Middleware, Springer-Verlag Lecture Notes in Computer Science, Vol. 2596, 2002.

[9] Bryant, B. R. and Lee, B.-S., "Two-Level Grammar as an Object-Oriented Requirements Specification Language," Proceedings of the 35th Hawaii International Conference on System Sciences, 2002, http://www.hicss.hawaii.edu/HICSS_35/HICSSpapers/PDFdocuments/STDSL01.pdf.

[10] Brahnmath, G. J., Raje, R. R., Olson, A. M., Auguston, M., Bryant, B. R., Burt, C. C., A Quality of Service Catalog for Software Components, Proceedings of the 2002 Southeastern Software Engineering Conference, pp. 513-520.

[11] Auguston, M., Tools for Program Dynamic Analysis, Testing, and Debugging Based on Event Grammars, Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering, 2000, pp. 159-166.

[12] Hudson, M. J., The Web Services Placebo, http://www.intelligententerprise.com/020917/515e_business1_1.shtml

[13] Czarnecki, K., Eisenecker, U.W., *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2000.