Service Development and Application Integration with Public Information System Mediator

S. Srbljic*, I. Skuliber**, I. Benc**, M. Stefanec**, A. Milanovic*, B. Dellas**, S. Desic**, L. Budin*, N. Bogunovic*, D. Huljenic***, and A. Caric***

* School of electrical engineering and computing, University of Zagreb, Zagreb, Croatia
** Ericsson Nikola Tesla d.d., Zagreb, Croatia
*** KATE, A StarCapital Company, R & D, Zagreb, Croatia
{sinisa.srbljic, andro.milanovic, leo.budin, nikola.bogunovic}@fer.hr
{ivan.skuliber, ivan.benc, mario.stefanec, bjorn.dellas, sasa.desic}@ericsson.com
{dhuljenic, acaric}@starcapital.net

***Abstract* — Software development industry is facing two important issues: enabling rapid service development and integration of existing applications. Current approaches result in custom solutions that are not generally applicable. Therefore, we propose a *service development and application integration system* acting as a *mediator* between developers, users, services, and applications.**

In this paper, we present the prototype of the public information system mediator MidArc[1]. We describe the mediator's system architecture, the technology it is built upon, and the process of integration of distributed applications. The mediator prototype was created and used to *develop* the formal automata simulator as a public service, to *integrate* the automata simulator service into the distance learning application, and to *run* the distance learning system.

I. INTRODUCTION

The driving force of contemporary information and telecommunication technology (ICT) industry are applications. They can be either mainstream products and services or software systems designed for specific customer [1]. Software companies are eager to minimize development expenses, making it necessary to materialize new ideas into applications fast and to integrate proven existing solutions and make them act as unified system.

The simplest approach to the integration problem is development of *custom integration solutions*. Integration of various data organizations and protocols [2] is easier by developing custom wrappers and services, than by creating *reusable integration components* that can be used in general. However, although the initial cost is lower, maintenance and upgrades of the custom solutions can be very expensive [3].

On the other hand, *creating the reusable integration components* requires identification of often-used application parts or services, and their implementation in most general and open way. Thus, they can be used by any vendor on any kind of platform and in any kind of application [4]. User management facilities like registration, authentication and authorization are obvious parts of all applications. Other services like security, load balancing or usage tracking are also needed in contemporary applications. Reusability is the major feature of the *service-oriented architecture* (SOA) [3]. Often used application parts are implemented in components and offered as services to application developers. Moreover, components can also be bridges, used to integrate different applications and systems. While using SOA, developers include components into applications and program the ties between included components.

In order to facilitate service development and application integration, we develop the distributed, modular and scalable system that offers different services implemented as software plug-ins (SOA components). Our MidArc system mediates between developers, users, services, and applications by offering design-time support for development and run-time support for execution of its plug-ins and applications. Since mediation is its primary characteristic, we have named our system the Mediator of Public Information System [5].

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the architecture of the MidArc mediator seen from three distinct aspects: system, technology and application. In Section 4 we illustrate the development and integration of distance learning application by using the MidArc mediator.

II. RELATED WORK

Various software products are developed in order to create a system of reusable components for application development [6]. Since one type of system often builds upon other type in order to achieve certain properties, the layered products are often called middleware systems. The classification of the middleware systems is presented in Fig. 1.

The GRID technology [7] enables rapid development of *infrastructural middleware systems.* They provide uniform software layer residing on various hardware and operating system platforms for harvesting hardware resources of

[1] The MidArc mediator is developed by the School of Electrical Engineering and Computing, University of Zagreb, Croatia, and Ericsson Nikola Tesla d.d., Zagreb, Croatia; MidArc project is partially sponsored by the Ministry of science, education and sport, Croatia, and is a part of CRO-Grid national project, TP-01/036-29, http://ris.zemris.fer.hr

distributed system. *Database middleware systems* are the most common type of middleware systems. They are comprised of drivers, wrappers and transaction monitors that facilitate access to and operations over different types of database systems. *Communication middleware systems* assure platform independent communication enabling one application to be run on different platforms simultaneously. Web Services [8] are the latest communication standard that represents the essence of communication middleware systems.

The most sophisticated middleware systems are those that deal with distributed application integration. Due to their nature of integration, *application middleware systems* incorporate most of other previously described middleware system types. There are not many middleware systems operating at this level, most of them exist only in laboratory environment. However, application middleware systems have started to mature in the recent years due to high demands from the ICT industry.

The concept of application middleware system was introduced in the AT&T's IP Platform GeoPlex project. AT&T's GeoPlex project described the idea of hybrid services spanning across different networks [9]. In addition, it created the notion of transferring common parts of applications' logic from the network periphery to the network itself [4]. The network offers these common parts for use by applications. User management functions have been identified as common application parts, as well as a number of system management functions, like security, load balancing and data caching. Exposing these functions through public APIs assures faster application development. Integration aspect of application middleware consists of the ability to translate between different protocols and of a records database that can be used as a statewide records database. AT&T targeted the IP Platform toward ISPs as an intermediate between ASPs, businesses and customers.

Active Networks [10] operate on IP network level, deploying the features that are related to IP packets routing and filtering, QoS signaling, Web caching, reliable multicast mechanisms, etc. Applications utilize deployed features to obtain required support in terms of network and network management resources. For example, the main actors of the FAIN model (Future Active IP Networks, EU R&D IST project) [11] are the active network service provider (ANSP), the service provider (SP), and the consumer (C). The ANSP offers basic network resources for the deployment and operation of the network active components. The SP obtains network resources from the ANSP and creates services comprising active components. It then deploys these components in the network, and offers the resulting service to Cs. The C is the end user of the active services offered by an SP. Besides FAIN, several active network implementations have been released: ANTS [12], SANE [13], BOWMAN, and CANES [14].

Microsoft's Hailstorm project, later named ".NET My Services" [15], is a platform that is an equivalent of a statewide records database with three basic user management functions: registration, authentication and authorization. These user management functions can be included into application as already developed components, thus facilitate application development process. However, the emphasis of Microsoft's project is user centric database and its interaction with users. The database is optimized for specific user-related functions like management of user's calendar, tracking user's location, etc. Authorized applications can access these functions through SOA like services. Interaction is bi-directional, meaning that the database sends information and requests additional information or authorizations from users. Communication technology based on Web Services and environment based on .Net Framework hide implementation details and provide operating system independence.

Resources Management	**Applications Management**
Database Middleware Platform independent database access and management. Transaction monitoring and processing. *Microsoft, Oracle products, etc.*	***Application Middleware*** Rapid development and system integration. *.NET My Services, MS Biz Talk, Vitria Businessware, etc.*
Infrastructural Middleware Hardware and software resource management. *GRID projects (Globus, Condor, Legion, etc.)*	***Communication Middleware*** Platform independent communication. *WEB services, CORBA, etc.*

Figure 1. Middleware systems classification

The application middleware systems are commonly used as *enterprise application/business integration products* [16, 17]. Utilizing SOA, redesigning integration platforms as a collection of integration services and components, and including common services for rapid application development, presents new and more general approach to application middleware systems with emphasis on customization, extendibility and reusability.

III. MIDARC ARCHITECTURE

The MidArc mediator is an application level middleware system that mediates between clients and servers in a distributed, Internet-based public information system. The public information system based on the mediator is presented in Fig. 2. The role of the mediator, service, and client in public information system is similar to the role of the active network service provider (ANSP), the service provider (S), and consumer (C) in FAIN model [11], respectively. However, instead of dealing with network and network management functions that provide network independence and QoS in FAIN model, we focus on high-level application functions that support concepts

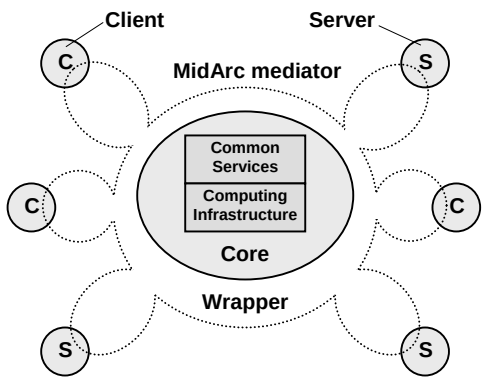

Figure 2. The MidArc mediator in public information system

closely related to the business processes, enterprise models, entertainment models, etc. In order to achieve this goal, the mediator supports service-oriented functions, like service description, discovery, delivery, and composition [18].

The MidArc mediator consists of *Core* and *Wrapper*. Core is physically isolated and secured part of the mediator, which provides infrastructure and execution environment for Wrapper. It also contains common services, which are used by clients and servers through Wrapper. Wrapper establishes presence of the mediator on clients and servers. Through its plug-ins, Wrapper provides various services and facilities that can be used by clients and servers.

A. *System architecture*

Since the MidArc mediator is an *application level middleware system*, it has to provide a wide range of services and facilities, which enable rapid distributed application development as well as maintenance during both design-time and run-time. In order to provide these services and facilities, the MidArc mediator builds upon and extends the other three classes of middleware systems.

Fig. 3 presents the *layered architecture* of the MidArc mediator. The vertical segment in the left portion of the Fig. 3 shows how four middleware system classes are organized and stacked in order to build the MidArc mediator. The vertical segment in the right portion of the Fig. 3 states the functions that are implemented at each level of the layered mediator architecture. The presented layered approach improves the flexibility and portability of the mediator. The bottom layer provides core functionality, and each successive layer is built on top of the lower layer, providing new services to the upper layers. The application level services are located at the top of the layered architecture.

Since the Core and the Wrapper constitute two distinct parts of the MidArc mediator, the architecture is further divided into two relating portions, as marked by the dashed lines.

The Core of the mediator consists of the computing infrastructure, distributed data infrastructure and common services. *Computing infrastructure* constitutes the bottom layer of the mediator system architecture. Its purpose is to provide dependable and scalable execution and hosting environment for the higher mediator layers. In order to meet these requirements, computing infrastructure layer is built using the infrastructural middleware. It contains services and facilities necessary for transparent deployment and execution of the mediator's components. One of the major benefits of using the infrastructural middleware is hardware and operating system independence.

Distributed data infrastructure extends the functionality provided by the computing infrastructure with the goal of creating physically distributed, but logically centralized data storage. It provides distributed data and transaction management that are developed by using the database middleware solutions. This layer stores user and service profiles, usage tracking and security related data.

Resource management provided by the computing infrastructure and data management provided by the distributed data infrastructure, make the foundation for the *common services*. Common services consist of functionalities necessary to develop distributed application, i.e. security, user and service management, auditing and usage tracking, etc. Since the developers of applications and services do not have to develop their own versions of these functionalities, they can speed-up service and application development by using the MidArc mediator.

The Wrapper of the mediator provides extended services and facilities, like application integration, interoperability and development. It is built upon mediator Core and consists of Wrapper services, service specific logic, and client specific logic.

The Wrapper services contain the logic for application integration and interoperability. Since the Wrapper services must meet various requirements including direct resource management and runtime efficiency, layer bridging is introduced into the mediator architecture. Layer bridging provides direct access to all core layers, so the Wrapper services can "bridge" one or more core layers. The layer bridging is primarily used to extend or improve the functionality of the common services. For

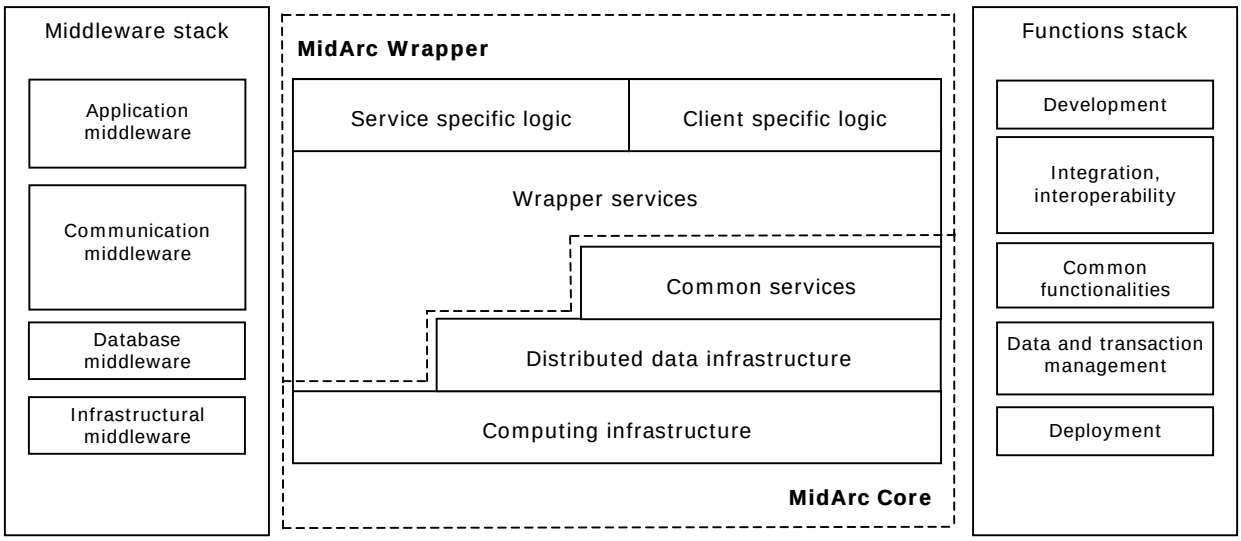

Figure 3. Layered view of mediator system architecture

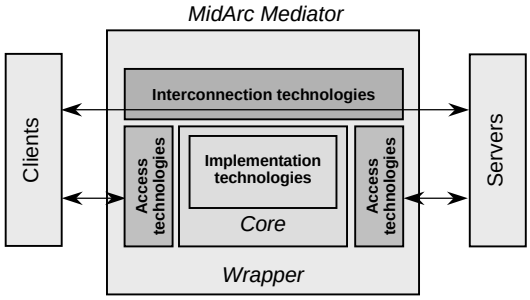

Figure 4. MidArc technologies

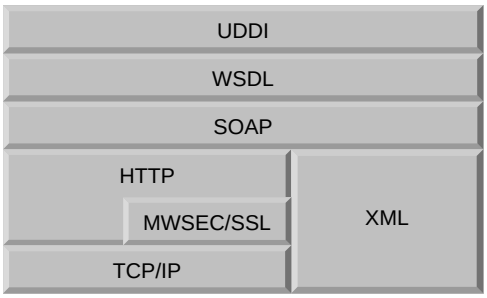

Figure 6. MidArc access technologies

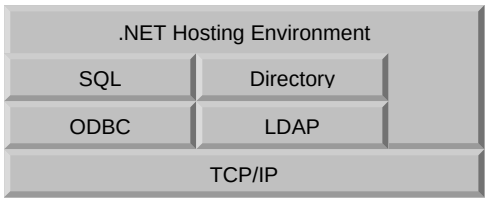

Figure 5. MidArc implementation technologies

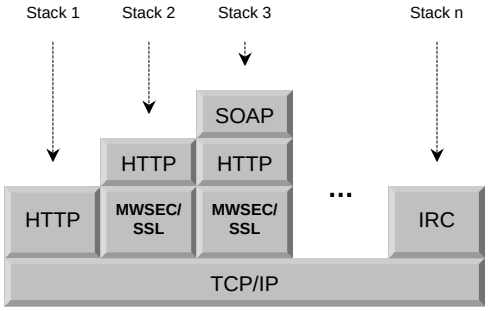

Figure 7. MidArc interconnection technologies

example, new designed service can inherit common security service and upgrade it according to the specific application requirements. Since Wrapper services should meet requirements set by a large number of different distributed applications, there is a large number of requirements, some of which can be contradictory. Therefore, it is impossible to predict, develop and deploy a fixed set of out-of-the-box Wrapper services. Instead of providing a huge number of pre-built Wrapper services, mediator Wrapper provides a mechanism to develop and deploy custom Wrapper services as plug-in components.

The service specific logic and *the client specific logic* of the Wrapper provide the functions needed for the client-server communication. When building client-server applications, developers use the development environment, which consists of development tools and communication functions necessary to access Wrapper services. The mediator wrapper is distributed on the client and server hosts in the form of the user and service agents.

B. Technology

The MidArc mediator is implemented using *three sets of technology classes*: implementation, interconnection, and access technologies. Fig. 4 presents the relations between technologies, mediator, clients and servers. While the implementation technologies are used to implement and operate the mediator Core, the access technologies and the interconnection technologies are used in the mediator Wrapper.

The *implementation technologies* are used in mediator Core in order to create the mediator's computing infrastructure. These technologies provide a *hosting environment* for all services and facilities of the mediator. Fig. 5 presents the technologies used in the implementation of the MidArc mediator. Microsoft .NET is used as the hosting environment of the mediator. All services and facilities are implemented as .NET classes. .NET hosting environment manages execution of .NET classes and supports their interconnection, synchronization and collaboration.

In addition, the implementation technologies are used to build the mediator *data management facilities*. Data management facilities consist of two types of databases used to store mediator data: a relational database and a directory. The relational database is based on SQL and accessed through the ODBC protocol. The data in the directory is accessed through the LDAP protocol.

The *access technologies* are used to expose mediator's common services and facilities to clients and servers. Clients and servers access all services and facilities of the MidArc mediator through remote procedure call (RPC) mechanism based on the Web Services RPC standard.

Fig. 6 presents the Web Services *protocol stack* used in the mediator. UDDI standard is located at the top of the Web Services protocol stack. UDDI is a *registry service* that stores the list of all methods exposed by the mediator as well as their descriptions. Since mediator can be expanded with new services and facilities, clients and servers use UDDI to discover the related access methods exposed by mediator.

All exposed methods are *formally described* using the WSDL language. Clients and servers use WSDL to gain the information on how to invoke the exposed methods. The remote methods of the MidArc mediator can then be invoked using the SOAP protocol. The SOAP protocol is a text-based protocol, which uses clear text and the XML to *encode* all parameters and other information of a remote procedure call into a single message. The MidArc mediator uses the SOAP over HTTP standard to transport the messages over the Internet.

Since both SOAP and HTTP messages are in clear text, a *security mechanism* is necessary to secure the information transmitted by SOAP. In addition to clear-text communication, the MidArc mediator offers the choice of custom developed MWSECURE protocol or the standard SSL protocol. These protocols provide message privacy,

confidentiality, sender authentication, and nonrepudiation. Finally, both clear-text messages and secure, encrypted messages are sent over the network using the TCP/IP protocol stack.

Interconnection technologies consist of various protocols used to connect clients and servers. The MidArc mediator provides the basic set of interconnection protocols, but this set can be extended through the Wrapper plug-in mechanism. The mediator includes a set of pre-built plug-ins that implement commonly used protocols. The supported protocols are HTTP, secure HTTP using either MWSECURE or SSL, SOAP over secure HTTP, and IRC. Individual protocols are bound together in order to form a protocol stack. A protocol stack is used to support complex and layered communication procedures used by clients and servers. For instance, when mediator connects Web server and a Web browser, the protocol stack consists of two protocols, TCP/IP and HTTP. Protocol stacks are the foundation of the MidArc mediator interconnection technologies, as presented in Fig. 7. New protocol stacks as well as individual protocol plug-ins are built using the mediator development facilities.

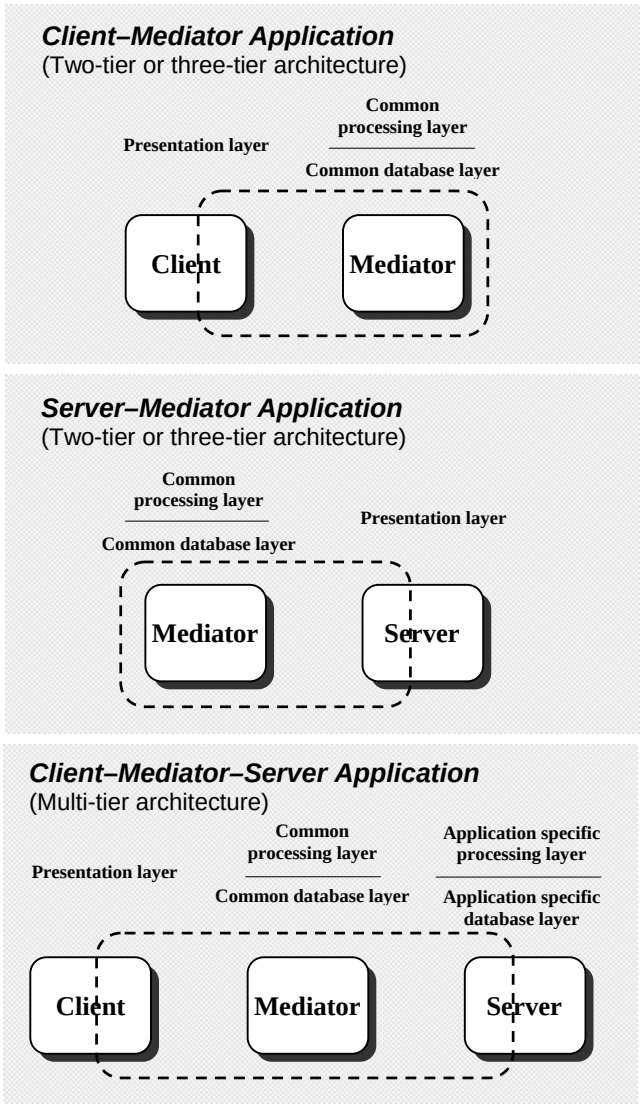

Figure 8. Typical architectures of applications integrated by the MidArc mediator

C. Distributed Application Architecture

Applications integrated by the MidArc mediator are structured into three layers in order to be distributed to multiple hosts: *presentation layer*, *processing layer*, and *database layer* [19]. These layers present the information to the users, implement the logic of the application, and perform database operations. Since often used application functionalities are isolated, modularized, and put into the MidArc mediator as common services, the processing and data management layers are additionally split into the four layers: *common processing layer*, *application specific processing layer*, *common database layer*, and *application specific database layer*.

Depending on (1) architecture of the computer infrastructure of the mediator Core, (2) complexity of the application, and (3) layer distribution through multiple hosts, the architecture of the integrated applications could be *two-tier*, *three-tier*, or *multi-tier* [7, 19]. Typical application architectures are presented in Fig. 8.

If an application only uses common services of the mediator, it has either two-tier or three-tier architecture, which depends on distribution of common processing layer and common database layer on the mediator hosts. Typical *Client-Mediator* applications perform client registration, billing statement checking, while *Server-Mediator* applications perform service registration, usage data analysis, etc.

Client-Mediator-Server applications use both common and application specific services. For example, common services could include client authentication and authorization, while application specific services could be operations of an e-banking system. These complex distributed applications have multi-tier architecture as presented in Fig. 8.

Involving more clients and services in a single application increases the architecture complexity. The MidArc mediator enables developers to integrate common services and application specific services at application design-time. However, we are exploring a possibility to request services on demand at application run-time, an approach described as demand-led application architecture [18].

IV. MIDARC IN PRACTICE

The applicability and usefulness of mediator system is continually being tested in practice. The SoftLab distance learning system [20], which is an integral part of computer science curriculum at School of Electrical Engineering and Computing, University of Zagreb, is developed and run on the MidArc mediator. The SoftLab distance learning system and the MidArc mediator is being used by more than 200 students and 10 system supervisors each semester in Automata Theory and Compiler Design courses.

Based on three years worth of testing, we make the following conclusions. The development of the SoftLab distance learning service from standalone application Automata Simulator [20] was very short. The integration of the SoftLab distance learning application proves the benefits of using the MidArc mediator. All network related services and user management services of the Automata Simulator application are implemented as the common services components of the MidArc mediator. Furthermore, the performance measurement [21, 22]

shows that low computer resources (a couple of Pentium III PCs connected with Ethernet) are sufficient to efficiently run the MidArc mediator and the SoftLab distance learning system. Measurements show that processor workload on the mediator computers never exceeded 10 percent, although testing involved more than 200 students. This indicates that with similar equipment larger groups of students can be efficiently served.

V. CONCLUSION

The concept of the application level middleware can be traced all the way back to the 1995 and AT&T Labs' project named GeoPlex. At that time, GeoPlex project has been literally ahead of time falling short of appropriate software technologies and lacking the terminology for its products. However, since then, the necessary software technologies have matured enough to support the development of an efficient application level middleware system.

In this paper, we have shown that, with selection of proper software technologies and by assembling them into the appropriate system architecture, it is possible to build efficient distributed application development and run-time support system. We have described the public information system mediator MidArc, the prototype of the application level middleware system. It is our solution to the problem of service development, application integration, and run-time support for distributed system execution. It is being developed by the School of Electrical Engineering and Computing, University of Zagreb, and Ericsson Nikola Tesla d.d., Zagreb, Croatia. We have described the architecture of the MidArc mediator through its system components, building technologies, and distributed application integration process. When compared to similar integrated middleware solutions, MidArc system offers openess for development of custom plug-in components and completely modular composition that assures simple system extendability.

MidArc mediator is used to run and develop the SoftLab distance learning system. Our experiences show facilitation of service development, application integration, and execution of the distance learning system. Currently, we are incorporating service-on-demand feature to the MidArc mediator.

REFERENCES

[1] M. Shaw and D. Garlan, *Software Architecture - Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[2] J. Pinkston, "The Ins and Outs of Integration", *EAI Journal*, August 2001, pp. 48-52

[3] R. Schmelzer, "Service-oriented integration", *Proceedings of the Boundaryless Information Flow: The Role of Web Services*, The Open Group, Boston, Massachutes, July, 2002.

[4] M. Lerner, G. Vanecek, N. Vidovic, and D. Vrsalovic, *Middleware Networks - Concept, Design and Deployment of Internet Infrastructure*, Kluwer Academic Publishers, 2000.

[5] S. Srbljic et al., "Application Middleware: A Case Study", *Annual of 2002 of the Croatian Academy of Engineering*, pp. 101-108, Zagreb, 2002.

[6] E. A. Gryazin, J. O. Tuominen, and O. Seppala, "Heterogeneous Middleware Structures for MyGrocer Project", *Proceedings of M-Business 2002*, Athens, July 2002.

[7] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, 1998.

[8] W3C, Web Services Activity, http://www.w3.org/2002/ws/

[9] G. Vanacek, N. Mihai, N. Vidovic, and D. Vrsalovic, "Enabling Hybrid Services in Emerging Data Networks", *IEEE Communication Magazine*, July 1999.

[10] D. Tennenhouse and D. Wetherall, "Towards an active network architecture", *Computer Communications Review*, 26, 2 (1996), pp. 5-18

[11] A. Galis et al., "A Flexible IP Active Networks Architecture", *Proceedings of Second International Working Conference, Active Networks*, IWAN 2000, Tokyo, Japan, October, 2000. pp 1-15

[12] D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols," *Proceedings of the 4th International Conference on OPENARCH*, San Francisco, California, April 1998, pp. 1–12.

[13] D. Alexander at al., "Safety and Security of Programmable Network Infrastructures, " *Communications Magazine*, pp. 84– 92. IEEE, October 1998.

[14] M Keaton at al., "Active Reliable Multicast on CANEs: A Case Study," *Proceedings of the 4th International Conference on OPENARCH*, Anchorage, Alaska, April 2001.

[15] Microsoft Corporation, *Microsoft .NET My Services Specification*, Microsoft Press, 2001.

[16] North Carolina Office of Information Technology Services, *North Carolina Statewide Technical Architecture*, 2003.

[17] B. Sumak, M. Hericko, I. Rozman, and M. Pusnik, "E-Businesses Integration Servers", *Proceedings of Mipro 2003*, Opatija, Croatia, 2003.

[18] M. Turner, D. Budgen, and P. Brereton, "Turning Software into a Service", *IEEE Computer*, October 2003, pp. 38-44

[19] I. Sommerville, *Softweare Engineering*, 6th ed., Pearson Education Limited, 2001.

[20] I. Skuliber, S. Srbljic, and A. Milanovic, "Extending the Textbook: A Distributed Tool for Learning Automata Theory Fundamentals", *Proceedings of ICECS 2002*, Dubrovnik, Croatia, September 15-18, 2002, pp. 1231-1234

[21] I. Benc, F. Plavec, and S. Srbljic, "Scalable Data Storage for Public Information System Middleware MidArc", *Proceedings of SCI 2003*, Orlando, Florida, USA, July 2003, Vol 3., pp. 241-246

[22] M. Stefanec, S Srbljic, and I. Skuliber, "Performance Evaluation of Distributed Objects Platform for Public Information System Implementation", *Proceedings of SCI 2003*, Orlando, Florida, USA, July 2003, Vol 3., pp. 259-264