

Application Middleware: A Case Study

Siniša Srbljić*, Darko Huljenić, Björn Dellas, Antun Carić, Ivan Benc, Ivan Skuliber, Mario Štefanec, Daniel Šimić, Andro Milanović*, Franjo Plavec*, Igor Grudenić*, Domagoj Krišto*, Goran Radić*, Dražen Klarić*, Miroslav Mihaljević*

Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb
{sinisa.srbljic, andro.milanovic, franjo.plavec, igor.grudenic, domagoj.kristo, goran.radic, drazen.klaric, miroslav.mihaljevic}@fer.hr

Ericsson Nikola Tesla
Krapinska 45, 10000 Zagreb
{darko.huljenic, bjorn.dellas, antun.caric, ivan.benc, ivan.skuliber, mario.stefanec, daniel.simic}@etk.ericsson.se

1 The Beginnings – Organization Chart

In the last decade we are witnesses of merging telecommunication and computer industries, and the birth of the information and communications technology (ICT) business. So far, telecommunication industry provided infrastructure and voice-based services. On the other hand, computer industry provided applications and services based on data transfer. With an extensive growth of the Internet and Internet-oriented services and applications, merge between the two worlds became an imperative. Newborn ICT industry is focused on bringing services to clients wherever they are and whichever device they are using.

Services are becoming even more interesting with the introduction of mobile access to the global network. New protocols, such as WAP, enable 24/7 access to the services that were primarily designed for non-mobile users. Since service and content providing is becoming main source of revenues, rapid service development is an imperative in merciless market competition. Every content provider that hosts services which offer content to the users must have an infrastructure that enables content offering, customer care and billing. Development of the required infrastructure is long and expensive process that yields with low or no profit.

As for service providers and developers, non-collaborative services have negative impact on end users. A good example is user registration and authentication. While every content provider must implement its own user management functions, users must register with every content provider whose services they use. Every registration is followed with choosing of user login and password for that service. After a while user has more logins and passwords than she can remember. To avoid endless development of common functions that every service needs (user authentication, registration, access control, and so on), we propose development of MidArc, an “all-IP” platform that will provide common functionalities to a number of content providers.

Development and deployment of a comprehensive “all-IP” platform is challenge in both research and development areas. Success of such a demanding project requires synergy of university sited knowledge and industry resources. Following worlds best practice models, cooperation between Faculty of Electrical Engineering and Computing (FER), University of Zagreb and Ericsson Nikola Tesla (ETK) was established in order to best utilize research efforts. Team of experts form both FER and ETK was formed with responsibility to steer and oversee project execution. In order to gain fresh ideas and educate coming generations of developers and researchers, parts of the project are delegated to undergraduate and graduate students. They participate in the project through their undergraduate and graduate thesis under guidance of the expert team. Following widespread practice of summer internships, ETK organized Summer Camp as an opportunity for undergraduate students to gain experience working on real problems in real-world environment. Described model of university-industry collaboration achieved substantial results and surfaced as a worthwhile research model.

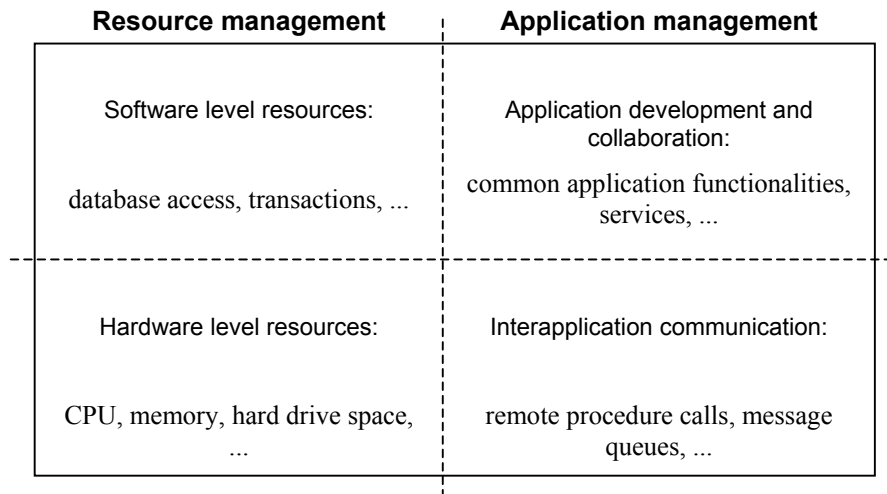


Figure 1. Classification of middleware systems

Rest of the paper is organized as follows: Section 2 classifies middleware systems and defines application middleware. In Section 3 related work in the field of middleware systems is described. Section 4 brings an overview of MidArc functionality, architecture and implementation, as well as a model of building distributed applications that utilize MidArc application middleware. Section 5 describes real world test case of the MidArc platform, while Section 6 concludes the paper and points out possible steps to take in the future work.

2 The Myth – What is Middleware

The term "middleware" has become a buzzword in the ICT industry in the recent years. Everyone in the industry uses it, wants to have it, because middleware is "in", it guarantees sales. Thus, the true legacy of middleware's past has been forgotten, and new middlewares of different types and purposes emerged, seeking their way toward customers. However, it is important to understand how everything began, to know the roots of the middleware. Our work is based on the original concept of middleware, the Internet platform, how it was called.

Geoplex [1], the first research project that began to explore the uncharted waters of middleware originated at AT&T Labs, IP Technology Organization at San Jose, California in the mid '90-es of the previous century. It was a visionary milestone for the Internet, a concept ahead of its time. In summary, the idea revolved around building a complex Internet platform for enabling cooperation between applications running on heterogeneous networks. Nowadays, very little information can be found about it, because the project was carefully buried under the ground and tightly sealed. However, project resulted with envisioned ideas, out of which several [2][3] were accepted as patents. We present summarized information about it that can be found scattered around the Internet in the next Section.

Today, the situation has changed. Many companies are offering middleware products. Middleware today means exactly what its literal meaning is - an intermediate between two counterparts. So, in order to present our concept properly, we must first classify existing middleware and specify the place where our work resides. Current middleware products and projects can be divided into two categories: resource management and application management (Figure 1). Middleware in the first category acts like a resource management facility. It can be separated into two groups, the lower and higher resource management, depending on which types of resources middleware manages. The lower resource management middleware manages hardware level resources like CPU, memory and hard drive space [4]. Since these are typical computer resources, we call this type of middleware resource middleware. The higher resource management middleware manages software level resources like database access, and transactions processing [5]. Because these resources are related to databases in general, this type of middleware we classify as database middleware.

The second middleware category deals with application management. Like the first category, it is also divided into two groups, the lower and higher, depending on the level of service abstraction middleware offers to applications. Interapplication communication middleware, the lower application management middleware, operates with remote procedure calls in general [6][7][8]. The higher application management middleware fully abstracts the network by substituting procedure calls with direct service calls. Applications invoke specific services that perform certain tasks for them. Thus, this type of middleware we call the application middleware.

According to given classification, our project lies in the field of application middleware. Hence, we see the middleware as an “all-IP” platform that, roughly described, enables interconnectivity and interoperability of different applications, systems and devices. Our goal is to create a platform that provides common application functionalities and bridges the differences of heterogeneous networks.

3 The Promises – Related Work

Nowadays there are a lot of research projects focused on middleware systems. However, most of these projects aren't at all related to application middleware or some of its parts. Majority of today's middleware projects are in the fields of the resource and communications middleware. In this Section, we present three middleware systems that are either very similar or at least have some common ground with our work.

Geoplex [1][9] is the first middleware project originated at AT&T Labs. The main goal of the project was to enable hybrid services in the emerging data networks. Hybrid services span different network technologies like the public switched telephone network (PSTN) and the Internet. Project researchers noticed that, at the time, network applications and services were self contained with little dependence on the network for anything other than access and transport. However, their analysis of those services revealed that much of the development cost and service complexity is common and already replicated across a wide range of service types. They identified these common issues as service supporting functions for all service and user related operations. The functions included registration, authentication, authorization, usage tracking, billing, encryption, etc. Researchers proposed to move the implementation of common function from service servers to somewhere in the network and enable access to functions by standard application programming interfaces (APIs). It is in this manner that our work tries to follow this concept and to continue the original middleware idea. According to our classification it is obvious that the original middleware is in fact an application middleware.

Globus [10] is the most advanced Grid computing related project up to date. It's main goal is establishing new approaches to distributed computing that bridge the differences of heterogeneous networks. Globus offers all common Grid features like processor, file and space sharing, but goes step further than other Grid projects by bringing additional functionalities layers on top of Grid layer. These layers define typical Grid application programming interfaces (API) for resource allocation and work distribution. However, there is also a layer that defines APIs for authentication. It is one of the crucial layers because authentication is the basis for authorization and encryption functionalities also offered by Globus. Despite authentication, and accompanying authorization and encryption, Globus'es primary focus is resource reservation and allocation, and thus it is a typical representative of resource middleware.

Current middleware project that in a way resembles to our work comes from Microsoft. The project .NET My Services [11] offers applications easy access to various user information. The goal is to provide the same information regardless of used input device, PC, PDA or a mobile phone. Only the graphical context through which the information is presented differs. User information discussed here contains e-mail, personal calendar, contacts, documents, wallet etc. A large portion of this information is of private nature, and thus, Microsoft's puts a lot of effort into creating secure authentication and authorization mechanisms. Judging from its features, .NET My Services belongs to application middleware.

4 The reality – MidArc middleware

Following the principles of middleware platforms development, we designed and implemented MidArc application middleware. The goals of the MidArc are simplification of development, deployment and usage of Internet based services, through the common use of MidArc functions, such as single sign on, security, privacy, usage tracking and usage billing. In order to insure mentioned goals of the MidArc platform we empowered the MidArc with the common functions found in almost every Internet service. The identified common functions placed within the MidArc define the framework for development of MidArc aware services and client programs. In this Section we present the benefits of using the MidArc platform and describe its architecture and framework.

4.1 Reasons to use MidArc platform

First question that has to be answered is why should an average Internet user or service provider use MidArc. First of all, using MidArc should be transparent to the end user. Burden of dealing with MidArc framework is set upon service provider's backs solely. Once adapted to the MidArc framework, service and end user can benefit from inherited MidArc characteristics explained below.

Non collaborative services present in nowadays networks impose a problem of remembering various user names and passwords for various applications. MidArc solves this problem by introducing user sessions. User has to authenticate herself while beginning user session, providing single user name and password. Access to the services within the session is authorized by the middleware, eliminating the need to authenticate on every service call. Single sign on (SSO) mechanism insures usage of services without remembering thousand and one different password.

Security issue is important aspect of every distributed system. Since middleware system is designed to enable development and deployment of distributed services in the Internet environment, security issue is given special consideration. There are two basic security problems. The first one is protecting user names and passwords in order to disable unauthorized access to sensitive information. The second security problem is preventing malicious users from collecting sensitive data by monitoring network traffic. In order to provide secure way of communication between client applications, middleware system and services, various security mechanisms are incorporated into middleware architecture. Symmetric and asymmetric encryption mechanisms accompanied by using of certificates insure security of the data transferred between entities of the middleware system.

While MidArc stores sensitive user and service related data, appropriate privacy mechanism has to be implemented in order to ensure privacy of personal data. Middleware system handles privacy issues through implemented authorization mechanisms. There are two types of authorization interfaces: authorization of the service access used in SSO mechanism and authorization of the access to registered entities of the middleware system. In order to ensure privacy of data stored in profiles, user or service administrator can define set of access rights rules. For example, right to access data such as credit cards numbers and home addresses can be given only to trusted services or individuals. Architecture of the middleware system guarantees that defined rules will not be bend.

The ability to track users' activity down to the single mouse click has always been appealing for the vendors of the Internet content. Questions how to optimize service appearance, which merchandise to advertise and how to offer personalized content have been driving forces in the usage data collection and mining field. Middleware system provides mechanism for filtering, collecting and partial analysis of the usage tracking data. Middle tier role of the middleware system is crucial for usage data collection process. MidArc features two types of usage tracking data. General usage tracking data represents information about time of service call, amount of transferred data, used protocol and so on. This data type does not depend of the service semantics. On the other hand, service specific usage data depends on the semantics of the called service. Middleware system has an ability to extract and store service specific data without getting into semantics and meaning of the collected data.

Middleware acts like glue between Internet applications. Although some Internet applications can be used free of charge, majority of them isn't intended for free use. In order to facilitate application development, middleware offers sophisticated billing system based on the above described usage tracking. Thus, billing system offers the ability of charging time spent using the application, time spent on specified parts or functions of the application, the number of service calls, the number of specified

service part of service function calls. Hence, all usual types of present e-commerce applications like electronic shopping basket are supported by the middleware.

A need to provide a service personalized according to user preferences is appealing to the most of the service providers. Middleware system enables provision of user oriented personalized services utilizing user profile data. Service with appropriate authorization level can access user preferences and provide content on user acceptable way.

4.2 Basic interaction of MidArc entities

Up to now, three types of MidArc entities could be observed: users, services, and the MidArc itself. Users communicate with services using client programs. Client programs are programs running on user computers that users use in order to access services. In most cases, Web browser is used as a user program for accessing services, although a specific program may be used if a service requires one.

Services, also called applications, are programs running on remote computers – Web or application servers. Web servers are powerful computers that run special software that enables storing of service data and running of service functionalities. An example of service program is a Web searching engine.

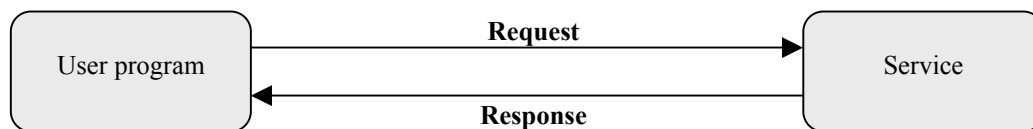


Figure 2. User – server program communication

A Web searching engine (i.e. <http://www.google.com>) is a service program that received a term as input, returns Web pages related to that term. Service data of a Web searcher service is a database of related terms and Web addresses, and the service specific functionality is logic that based on a term searches the database and generates the results. Typically, server programs are made out of many functionalities, out of which some are service specific and the other are general.

User and service collaborate using requests and responses (Figure 2). When a user wants to use a particular service he/she indicates this to its user program. User program issues a request to the service. Service running on the Web server receives the request, runs the logic of the service, and sends response with service relevant data. User program receives the response, possibly does a part of the service logic on the received data, and displays the data to the user. Usually, using of one Web application involves exchanging of multiple requests and responses between user and service. Additionally, service can issue requests to other services, which allows building of complex inter-tied services.

The MidArc is a set of functionalities running on a network computer that simplifies collaboration of user programs and services. MidArc acts as a middle tier in request-response chain (Figure 3). All user issued requests flow unconditionally through the MidArc to the servers, as do all the responses from the servers to the users. This enables moving of the part of the service functionalities from the server programs to the MidArc.



Figure 3. MidArc as a middleman in request-response chain

4.3 MidArc functionalities

The first of the functionalities implemented by the MidArc is **Registration**. Registration is a one-time process during which entities (users and services) register with the MidArc node. During registration entities supply the MidArc node with information about themselves. Prior to service use, service

administrators have to register services to the MidArc node. While registering services, service administrators supply the node with the following service related information: service access point (URL), service description, usage terms, and other. In order to use the MidArc system users register with the MidArc node. During registration users supply the node with their personal information such as: name and family name, date of birth, sex, address, telephone numbers and other. Registration process needs to be done only once, but an entity can reregister, in which case all the entity data is reentered to the MidArc node. Entities register with the MidArc node using MidArc's Registration interface.

The **Authentication** is a process conducted by the entities (users and services) during logging on to the MidArc system. During authentication process entities prove to the MidArc node their identity. After an entity is authenticated, a session key is generated and governed to it. A period between consecutive logging in and logging out of the MidArc node is called a session. A session key governed to the entity is guaranteed to be unique during a session, and it is used as entity identification during that session. Entities logon, and logoff the MidArc using the MidArc's Authentication interface.

Service access authorization is a process conducted during processing of the requests on the MidArc node. As mentioned earlier all the service requests traverse MidArc node. MidArc node checks the authorization rights of the issuing entity, prior to forwarding the request to the service programs. If the entity has rights to access the requested service, MidArc forwards the request to the service, receives the response and forwards the response to the entity program. If the entity does not have the rights to access the service, an error response is generated on the MidArc node, and sent to the entity program. MidArc's Request Handler interface manages requests and responses exchanged between user and service programs. Service access authorization is one of the functionalities implemented within Request Handler interface.

As service access authorization process, **Service Usage Tracking** process is conducted during processing of the requests and responses on the MidArc node. For each request and response flowing through the MidArc node, the following notes are made: who issued the requests, the time request arrived at the MidArc node, the time response was sent to the request issuer, authorization info of the request (authorized or not), the size of the request and response and service specific data. The notes are grouped together and stored to the usage-tracking database as a usage record. Users can examine notes about service request they issued, and services can examine notes about services request targeting them. Collecting of usage records is implemented as a part of Request Handler interface. Entities can extract and examine usage records related to them using MidArc's Usage Information interface.

Identification is a process by which service programs identify issuers of the requests. Identity of the entity (service or user) issuing a request is known to the MidArc node, but not to the service programs. Service programs can interrogate the MidArc node about the identity of the entity issuing the request. The retrieved identity can be used as a base for service personalization. Services interrogate MidArc about identities of the accessing entities using the Entity Data interface.

Service personalization can be enhanced using the service and user data stored in the MidArc node. For this reason, **browsing of user and service data**, stored in the MidArc is enabled. Services can interrogate MidArc node about user personal information and adopt their offering based on it. Due to the privacy reasons, data browsing is a subject to the authorization. Browsing user and service data is implemented as a part of Entity Data interface.

Administration is functionality used for administrating of the other common functionalities MidArc node implements. It enables authorization rights management, checking of usage records and other tasks. Administration is managed through Administration interface.

4.4 Building Web applications using MidArc

When developing with MidArc, one has to have in mind MidArc architecture in order to empower all benefits offered by the platform. MidArc consists of two major components: MidArc Core and MidArc Tentacle (shaded parts in Figure 4). MidArc Core is a complex middleware proxy component residing in the network node, while MidArc Tentacle is a light middleware agent component situated on the users' computer. The collaboration of these two components executes all MidArc functionalities. Core part of the MidArc middleware implements common functions stated earlier in this section. Beside common functions, it also implements distributed database for storing profiles of registered users, services and usage tracking data. Tentacle is MidArc component present on end user's device for

accessing services. Tentacle is crucial for ensuring Single Sign On, security, and authorization mechanisms. Although it is possible to develop user programs without Tentacle component, that is not recommended. With architecture of the MidArc in mind it is easy to develop MidArc-aware services and client programs.

The server side of the application in MidArc middleware oriented network comprises of three distinct parts as shown in Figure 4. These parts are: the core of the server, specific business logic interfaces exposed to the client part of the application, and communication logic that uses publicly exposed MidArc interfaces for management of common functions. The core of the server program contains specific business processing logic and, if necessary, specific business related databases. All business related operations accessible by clients are exposed as business interfaces. Servers utilize functionalities implemented by the MidArc in two distinct ways: explicitly by accessing MidArc's interfaces, or implicitly during MidArc's handling of the requests-response chain. Servers explicitly call registration, authentication, entity profile and usage-tracking interfaces, which handle corresponding common functionalities. Common functionalities of registering and authenticating users, authorizing service access and tracking usages does the MidArc Core without intervening of the server programs.

The client side of the application in MidArc middleware oriented network consists of three parts as well: Tentacle, core, and MidArc accessing communication logic. The Tentacle is MidArc component distributed to user computers. All client programs running on a single user computer, share one

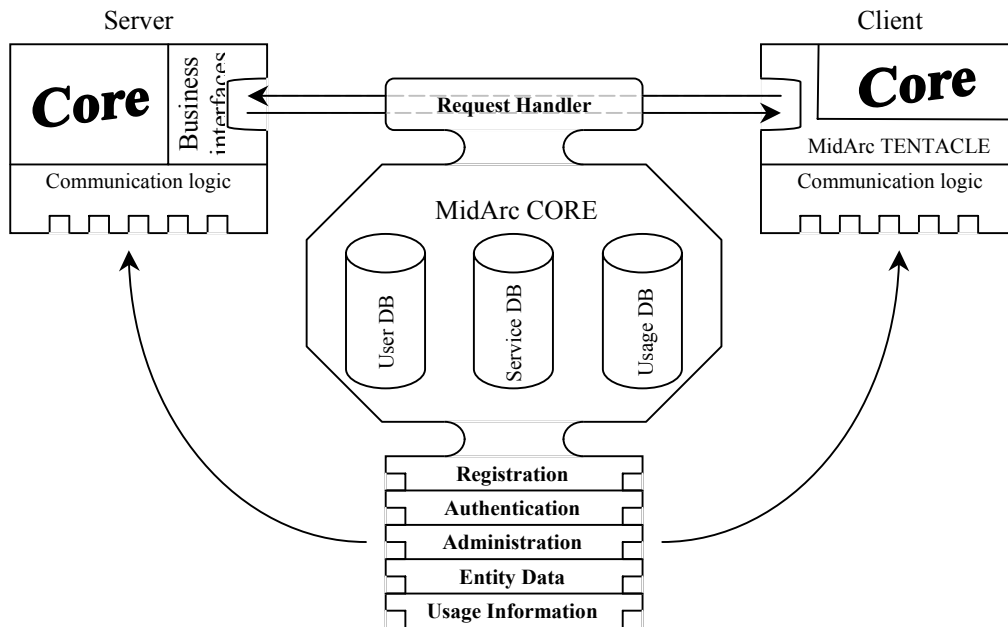


Figure 4. Collaboration of user, and service programs using MidArc

Tentacle component. The responsibilities of the Tentacle component are authentication of the user to the MidArc core, and managing of user request between the user programs and the MidArc core, as stated in previous sections. The core of the client possesses processing logic to analyze data received from the server and present it visually to the user. The client calls server's specific business logic interfaces, through the Tentacle and Request Handler interface of the MidArc middleware, and receives business data from them. Client programs may also use MidArc interfaces to customize user interface presented to users, or to enhance logic conducted in the client program core. MidArc accessing communicating logic is a part of the client program responsible for handling communication with MidArc interfaces.

MidArc platform enables development of peer-to-peer applications as well. Development of a peer-to-peer application in MidArc middleware oriented network resembles the development of the server side of the application, but with a few additional typical client side functionalities. This is because each peer has the functionality of a server and a client. Thus, a peer is constituted of parts that are present in both

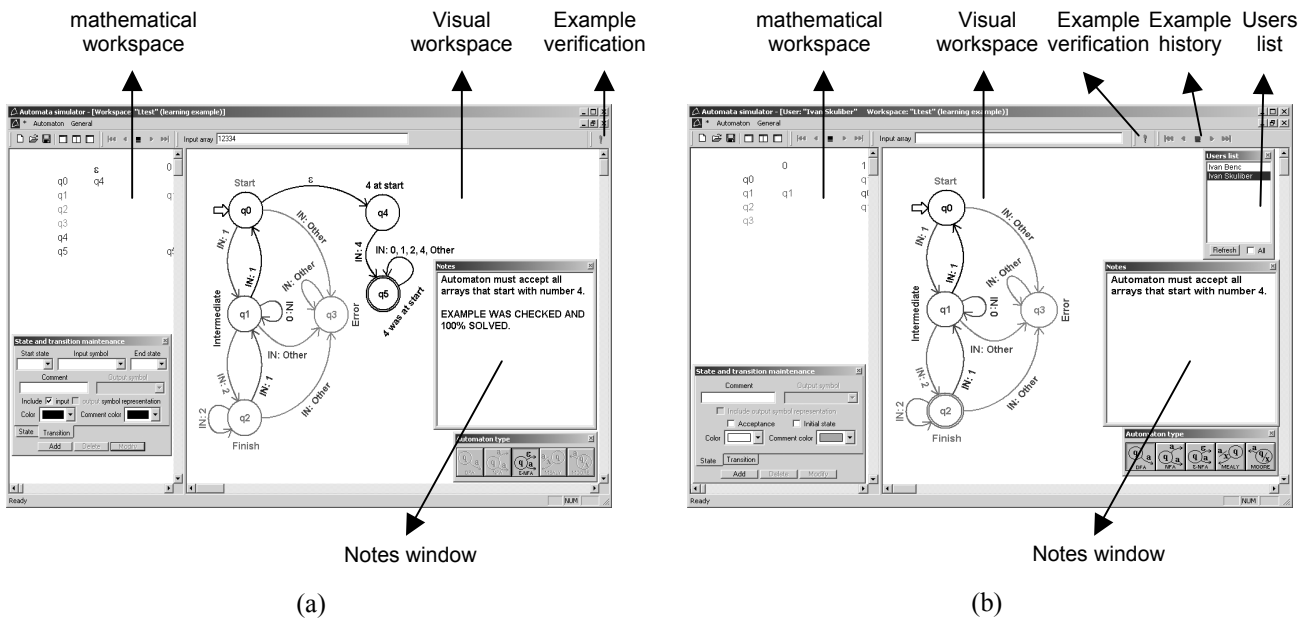


Figure 5. SoftLab applications – (a) student's client side, (b) supervisor's client side

the server and the client of a client-server application model in the MidArc middleware oriented network. The core part of a peer must be able, on the one hand, to process incoming requests, and on the other hand, to analyze received results and visually present them to user. Each peer must expose the same business logic interfaces. Also, each peer must have integrated support for accessing MidArc middleware's interfaces.

5 Blast Off – Real World Application Testing

The biggest obstacle in testing modern systems and applications is ensuring respective number of test users. Since a part of the MidArc middleware development is conducted on Faculty of Electrical Engineering and Computing, it was realized that students could be used as test users for real world application testing. The advantages of using students as a test group are threefold. First, faculty has good computing infrastructure, which can run MidArc and MidArc based applications. Second, students are eager at noting disadvantages and flaws of tested applications, and finally if applications are built into student courses, students can be obligated to test the MidArc platform. As the result of previous observations it was decided that a distance learning application upon MidArc platform will be developed.

Developed distance learning application – SoftLab [12] is a learning tool, which helps students comprehend complex mathematical models of formal automata theory (Figure 5). From user's perspective, SoftLab application consists of two different modules: student module (Figure 5-a) and supervisor module (Figure 5-b). While using student module, students can examine examples of automata both in a visual and mathematical workspaces, modify automates, solve test examples and check their rightness, and communicate with the supervisor. Supervisors (professors, teaching assistants) use the supervisor module to add new automata examples, and examine student's work in progress. While examining student's work supervisor can mark noted errors and help the student to overcome obstacles in the learning process. Architecturally, SoftLab service consists of three parts: SoftLab service program, SoftLab client program, and the MidArc node (Figure 4). Server side of the SoftLab consists of database, which holds information about users and their work, as well as interfaces that enable access to that information. All generic functionalities are located in the middleware layer – the MidArc node, and are accessible as described in previous sections.

The MidArc system and the SoftLab service were deployed on two computer sets of the Ericsson-laboratory at the Department of Electronics, Microelectronics, Computing, and Intelligent Systems (ZEMRIS) of the FER. One computer set served as the MidArc node, while the other one served as the

SoftLab service. The SoftLab service was introduced as an obligatory exercise of the “Introduction to automata theory 2” course during the academic year 2001/2002. During that year, two hundred students used the SoftLab application as a part of their laboratory work. Students were given student client module and MidArc Tentacle, which they installed on their personal computers.

Based on the test results three categories of observations can be made: application development related, user impressions and performance measurement. Development of SoftLab application proved the benefits of the middleware concept. SoftLab application is an extension of an existing stand-alone application AutomataSimulator [13]. Since most of the specific functionalities of the SoftLab application were already developed in the AutomataSimulator, and all network related generic functionalities are implemented by the MidArc, the development process of SoftLab application was extremely short. Students that used the service have reported numerous deficits of application, and shown directions for further developments. However, generally they agreed that application is a helpful resource in better understanding of automata theory. Performance measurements have shown that even low computer resources (couple of Pentium III PC’s connected with Ethernet) are sufficient for handling hundreds of users. Furthermore, measurements have shown that although testing involved more than two hundred students, workload on none of the computers never exceeded ten percent. This indicates that with similar equipment even larger groups of students can be served.

6 The New Beginning – Conclusion and Future Work

Since middleware architecture becomes more and more important part of the application development, through this paper we tried to give an overview of the state of the art in the field and present our concept and implementation the MidArc middleware platform. Following the classification of the middleware systems we characterized MidArc as an application middleware platform. Its intention is to enable rapid development and deployment of distributed applications on “all-IP” platforms (such as Internet), as well as to simplify their usage and maintenance.

This paper describes benefits that can be gained from the MidArc platform, functionalities it provides and high level platform architecture. Functionalities of the platform and its architecture define framework for developing MidArc-aware applications. Utilizing defined framework we developed SoftLab testing application in order to prove MidArc concept and test the platform in real world environment. Since SoftLab was deployed at Faculty of Electrical Engineering and Computing we engaged undergraduate students as test users. Test results proved straightforward development of distributed applications and stability of the MidArc implementation.

Due to complexity and broad technical area MidArc encompasses, project was conducted in cooperation between Ericsson Nikola Tesla and Faculty of Electrical Engineering and Computing, University of Zagreb. We are sure that the synergy of both the university and industry worlds ensures inevitable project success.

MidArc as is, can be assumed as basic application middleware platform upon which new values and functionalities can be built. For the future work we plan to conduct more real world and stress tests in order to improve performance of the platform. From the research aspect we are eager to define and build specialized application middleware platforms upon existing MidArc, such as distance learning application middleware.

7 References

- [1] G. Vanacek, N. Mihai, N. Vidovic and D. Vrsalovic: “Enabling Hybrid Services in Emerging Data Networks”, IEEE Communication Magazine, July 1999.
- [2] S. Srbljić, P.P. Dutta, T.B. London, D.F. Vrsalović, and J.J. Chiang: ”Scalable Distributed Caching System and Method”, United States Patent 5,933,849, issued August 3, 1999. (<http://www.uspto.gov/>)
- [3] S. Srbljić, P.P. Dutta, T.B. London, D.F. Vrsalović, and J.J. Chiang: ” Scalable network object caching”, United States Patent 6,154,811, issued November 28, 2000. (<http://www.uspto.gov/>)

- [4] I. Foster, C. Kesselman, S. Tuecke: "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International J. Supercomputer Applications*, 15(3), 2001.
- [5] I. C. Yoon, H. H. Kim, D. H. Bae and C. Youn: "Reliable Transaction Design Using MTS", the Twenty-Fourth Annual International Computer Software and Applications Conference, October 2000.
- [6] IBM Web Services Architecture Team: "Web Services architecture overview", October 2001.
- [7] Object Management Group: "The Common Object Request Broker: Architecture and Specification", July 1995.
- [8] Sun Corporation: "Remote Method Invocation Specification", <http://java.sun.com/products/jdk/1.2/docs/guide/rmi>, November 2001.
- [9] M. Lerner, G. Vanecek, N. Vidovic and D. Vrsalovic: "Middleware Networks: Concept, Design, and Deployment of Internet Infrastructure", Kluwer Academic Publishers, 2000.
- [10] I. Foster, C. Kesselman: "Globus: A Metacomputing Infrastructure Toolkit", *Intl J. Supercomputer Applications*, 11(2), 1997.
- [11] .Net Services, <http://www.microsoft.com/netservices/default.asp>
- [12] I. Skuliber, S. Srblic and A. Milanovic: "Extending the Textbook: A Distributed Tool for Learning Automata Theory Fundamentals", International Conference on Electronics, Circuits and Systems, Dubrovnik, October 2002.
- [13] I. Skuliber, S. Srblic, and I. Crkvenac: "Using Interactivity in Computer-Facilitated Learning for Efficient Comprehension of Mathematical Abstractions" in Proc. IEEE Int. Conf. Trends in Communications EUROCON'2001, Bratislava, SK, July 2001.