A Service Oriented Reflective Wireless Middleware

Bora Yurday[1] and Halûk Gümüşkaya[2]

[1] TUBITAK-MAM, ETCBASE Yazılım
Gebze, Turkey
bora.yurday@etcbase.com
[2] Department of Computer Engineering, Fatih University
34500 Istanbul, Turkey
haluk@fatih.edu.tr

Abstract. The role of middleware has become increasingly important in mobile computing, where the integration of different applications and services from different wired and wireless businesses and service providers exist. The requirements and functionalities of the wireless middleware can be achieved by Service Oriented Computing which can be an ideal paradigm for mobile services. Reflective middleware responses are optimized to changing environments and requirements. In this paper a Service Oriented Reflective Wireless Middleware (SORWiM) is proposed. It provides basic (Event, Messaging, Location, and Redirection) and composite services for efficient and reliable information discovery and dissemination in ad hoc mobile environments. One of the primary goals of this research is to investigate how the construction of mobile services can benefit from the Service-Oriented paradigm.

1 Introduction

Middleware is distributed software that sits above the operating system and below the application layer and abstracts the heterogeneity of the underlying environment [1]. It provides integration and interoperability of applications and services running on heterogeneous computing and communications devices, and simplifies distributed programming. Middleware can be decomposed into multiple layers such as host infrastructure middleware (Java Virtual Machine (JVM)), distribution middleware (Java RMI, CORBA, Simple Object Access Protocol (SOAP)), common middleware services (J2EE, .NET), and domain-specific middleware services (developed for particular domains, such as telecom, or e-commerce) [2]. Conventional middleware technologies, such as CORBA and RMI have been designed and used successfully with fixed networks. However, there are significant challenges to design and optimize middleware for mobile computing. Since conventional middleware platforms are not appropriate for mobile computing, because, first of all, they are too big and inflexible for mobile devices. Mobility, quality of service, security, management of services, service discovery, ad hoc networking and dynamic configuration are main middleware issues for mobile computing. It is therefore essential to devise new middleware solutions and capabilities to fulfill the requirements of emerging mobile technologies.

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 545 – 556, 2006.

Service-Oriented Computing (SOC) [3] is a distributed computing paradigm based on the Service-Oriented Architecture (SOA) [4], which is an architectural style for building software applications that use services. SOC and SOA are not completely new concepts; other distributed computing technologies like CORBA and RMI have been based around similar concepts. SOA and SOC are merely extensions of the existing concepts and new technologies, like XML, and Web Services, are being used to realize platform independent distributed systems.

The SOA appears to be an ideal paradigm for mobile services. However, it is currently focused only on enterprise and business services. In addition, most of SOA research has been focused on architectures and implementations for wired networks. There are many challenges that need to be addressed by wireless middleware based on SOA. Wireless middleware will play an essential role in managing and provisioning service-oriented applications. In this paper a Service Oriented Reflective Wireless Middleware (SORWiM) is proposed. It provides a set of services for efficient and reliable information discovery and dissemination in ad hoc mobile environments. One of the primary goals of this research is to investigate how the construction of mobile services can benefit from the Service-Oriented paradigm.

This paper is structured as follows. In section 2, the properties affecting the design of wireless middleware and why conventional middleware platforms are not appropriate for mobile computing are presented. In section 3, the service oriented approach to wireless middleware and SORWiM are introduced. In section 4, the detailed architecture and services are given. In section 5, service orchestration and mobile application scenarios are introduced. In section 6, the performance evaluation of SORWiM is presented, and finally in section 7, our future work is given.

2 Wireless Middleware for Mobile Computing

Limited resources, heterogeneity, and a high degree of dynamism are the most common properties that usually exist in mobile devices such as pocket PCs, PDAs, sensors, phones, and appliances. Although limited resource availability varies from device to device; mobile devices generally don't have powerful CPUs, large amount of memory and high-speed I/O and networking compared to desktop PCs. Different hardware and software platforms, operating systems imply changes in some parameters such as byte ordering, byte length of standard types, and communication protocols. The degree of dynamism present in ubiquitous computing does not exist in traditional servers and workstations. A PDA, for example, interacts with many devices and services in different locations, which implies many changing parameters such as the type of communication network, protocols, and security policies. Therefore, because we can't predict all possible combinations, the software running on a PDA device must be able to adapt to different scenarios to cope with such dynamism.

All these properties affect the design of the wireless middleware infrastructure required for mobile computing. Conventional middleware platforms are not appropriate, because, first of all, they are too big and inflexible for mobile devices [5], [6]. A wireless middleware should be lightweight as it must run on hand-held, resource-scarce devices. Conventional middleware platforms expect static connectivity, reliable channels, and high bandwidth that are limited in resource-varying wireless networks.

Wireless middleware should support an asynchronous form of communication, as mobile devices connect to the network opportunistically and for short periods of time. It should be built with the principle of awareness in mind, to allow its applications to adapt its own and the middleware behavior to changes in the context of execution, so as to achieve the best quality of service and optimal use of resources. Hiding network topologies and other deployment details from distributed applications becomes both harder and undesirable since applications and middleware should adapt according to changes in location, connectivity, bandwidth, and battery power.

New wireless network middleware is required to increase performance of applications running across potentially mixed wireless networks (from GPRS to WLANs), supporting multiple wireless devices, providing continuous wireless access to content and applications, as well as to overcome periods of disconnection and time-varying bandwidth delivery. Wireless middleware could also ensure end-to-end security and dependability from handheld devices to application servers.

We cannot customize existing middleware platforms manually for every particular device. It is not flexible or suitable for coping with dynamic changes in the execution environment. Reflective middleware presents a comprehensive solution to deal with ubiquitous computing [7]. Reflective middleware system responses are optimized to changing environments or requirements, including mobile interconnections, power levels, CPU/network bandwidth, latency/jitter, and dependability needs. Reflection is the ability of a program to observe and possibly modify its structure and behavior. In the SORWiM architecture, reflection was used in the service level. A fully reflective middleware was not implemented initially; instead we worked on reflective services that take decisions according to context information.

Principles and guidelines for designing middleware for mobile computing have been published in literature [5], [6], [7], [8], and some wireless middleware projects have been developed for some specific areas, such as sensor networks [9], [10]. A few researchers have also published service oriented computing imperatives for wireless environments [11], [12]. To the best of our knowledge, there are no or a few real implemented wireless middleware platforms based on SOA.

3 A Service Oriented Approach to Wireless Middleware

A SOA-based service is self-contained, i.e., the service maintains its own state [4]. A service consists of an interface describing operations accessible by message exchange. Services are autonomous, platform-independent and can be described, published, dynamically located, invoked and (re-)combined and programmed using standard protocols. SOA promotes loose coupling between software components.

The building block of SOA is the SOAP [13]. SOAP is an XML-based messaging protocol defining standard mechanism for remote procedure calls. The Web Service Description Language (WSDL) [14] defines the interface and details service interactions. The Universal Description Discovery and Integration (UDDI) protocol supports publication and discovery facilities [15]. Finally, the Business Process Execution Language for Web Services (BPEL4WS) [16] is exploited to produce a service by composing other services.

The high level architecture of SORWiM using the reference model of the web services standards stack is shown in Fig 1. There are four services, Event (Notification), Messaging, Location, and Redirection. These are some of the first important and basic services in a typical wireless environment. We have also developed some example composite services using these basic services in our implementation.

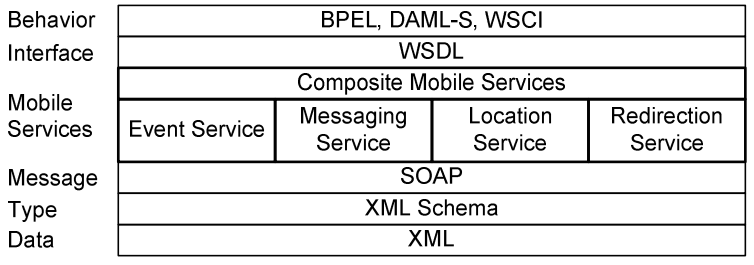


Fig. 1. The high level architecture of SORWiM based on the Web Services Standards Stack

4 The Architecture of SORWiM and Its Services

The architecture of SORWiM is shown in Fig 2. Each service is depicted by a WSDL document, which describes service accessing rules. Web services are registered to the central UDDI database. The client searches the UDDI to find out the service it needs, fetches the WSDL file, and generates the stub with the WSDL to stub code generator provided by the web service toolkit, and starts calling remote methods.

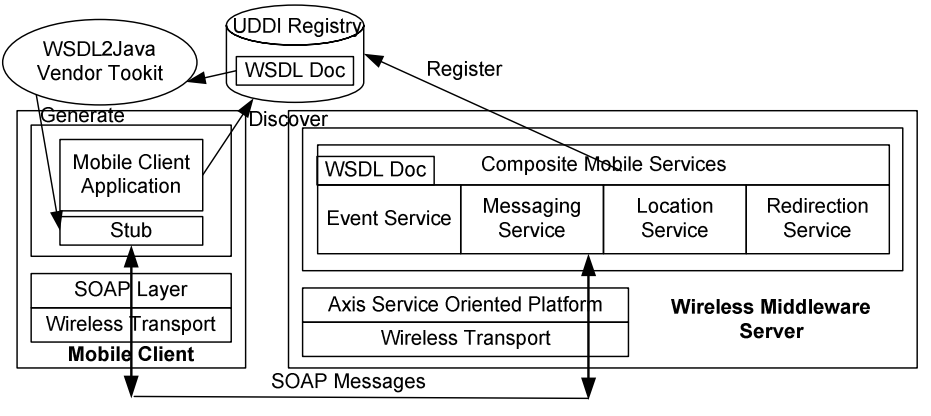


Fig. 2. The architecture of SORWiM

The package diagrams of SORWiM basic services are shown in Fig 3. A package describes a stateless service which can communicate with other services through service interfaces. That is there is no dependency between services which is one of the

primary goals of SOA. By defining relations between services we create composite services that form new application scenarios in our wireless framework. We extensively used the design patterns [17] in the implementation of the SORWiM architecture. Design patterns codify design expertise, thus providing time-proven solutions to commonly occurring software problems in certain contexts [17], [18].

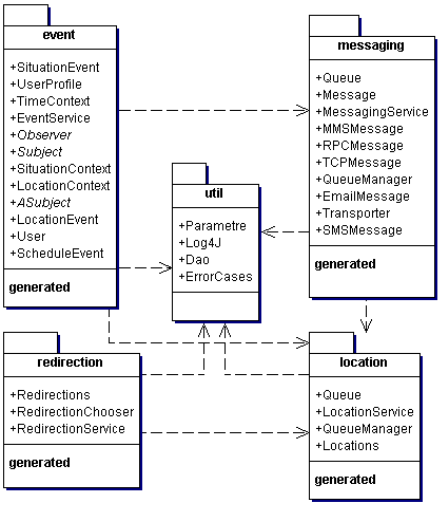
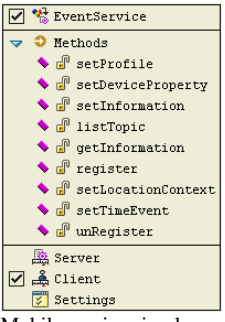


Fig. 3. The UML package diagrams of SORWiM basic mobile services

All basic services use the util package which organizes logging, error cases, database operations, initializing parameters. Since we need single instances of the classes in this package and a global point of access to them, the Singleton pattern [17] is used in the implementation.

A mobile client application, TestPad, which aims to test the basic services, was written using the PocketBuilder, the Sybase's the rapid application development tool for building mobile and wireless applications running on Microsoft Pocket PC devices [19]. In the middleware implementation we used the Apache Axis which is a proven service oriented platform to develop Java web services [20]. In the following, first, the basic services are explained. Then how composite services are orchestrated using these basic services are presented.

The EventService is one of the first basic services of SORWiM. It provides an interface schema for four main functions, authentication, profiles and preferences, notification (alert), and system and user information as shown in Fig. 4. A behavioral pattern, the Observer pattern [17], is used in the implementation of the notification engine. This pattern defines a one-to-many dependency between a subject object and any number of observer objects so that when the subject object changes state, all its observer objects are notified and updated automatically.



Mobile services implemented in the Axis 1.2

Authentication

- register: Mobile user logs in the system
- unRegister: Log off the system

Profiles and Preferences

- setProfile: Detailed user profile and preferences

Notification

- setDeviceProperty: Device info such as battery power, memory, and communication bandwidth
- setLocationContext: Location info
- setTimeEvent: Time based-notification event

System and User Information

- getInformation/ listTopic: get system and user information such as profiles, preferences/list a topic

Fig. 4. The services in the EventService

The MessagingService package has basic messaging services to create, send, receive and read XML based messages for mobile applications. This service can send messages using RPC, SMTP, SMS, and MMS types. The simplified class diagram of MessagingService is shown in Fig. 5. We used the Factory pattern [17] for creating different message types. This pattern deals with the problem of creating objects without specifying the exact class of object that will be created. The asynchronous communication is provided by the message oriented middleware approach of SORWiM using a messaging queue structure. The MessagingService class provides the functionality of the Facade pattern [17] for the messaging service package. It decouples the other classes of the MessagingService package from its clients and other subsystems, thereby promoting service independence and portability.

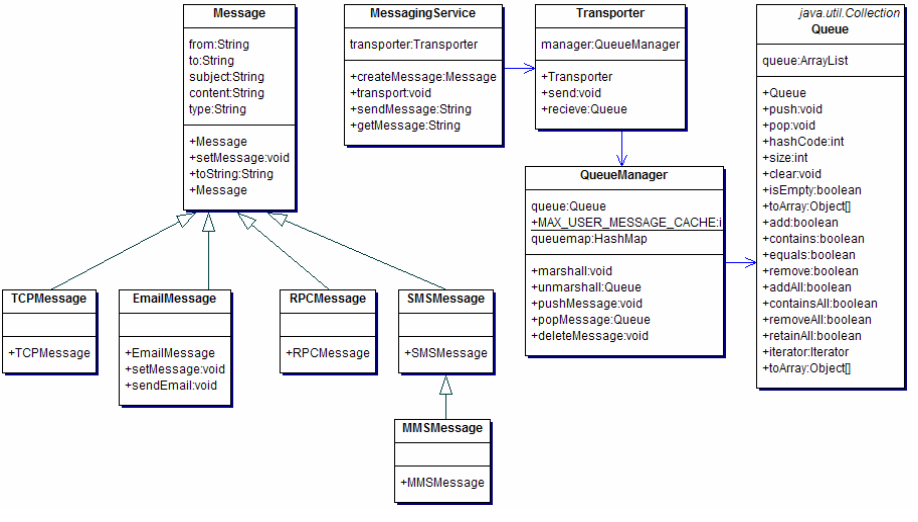


Fig. 5. The simplified class diagram of the MessagingService

The Location Service handles the problem of searching and updating locations of mobile nodes in a wireless network. This service is also used by other services for giving decisions and providing active context-awareness that autonomously changes the application behavior according to the sensed location information.

We designed and implemented an indoor positioning system, WiPoD (Wireless Position Detector), which locates and tracks a user having an IEEE 802.11 supported device across the coverage area of a WLAN [21]. With high probability, WiPoD can estimate a user's location to within a few meters of his/her actual location. The location information for the SORWiM indoor applications will be provided by WiPoD.

The services such as redirectionRequest and replicationRequest in the Redirection package are responsible for client redirection to different servers and server reference translation. There are two main problems dealt with in redirection: Address migration and data replication, both provide migration transparency. Migration transparency allows components to change their location, which is not seen by a client requesting these components. Server translation and redirection can be done transparently or non transparently. The Redirection Service needs to be aware of the state of connectivity and the location of the client at any point in time. The steps of redirection:

- Request- sent from a client to a server to signal that the client wishes to carry out a request on the server
- Response-sent from the server to a client in response to a request
- Redirection – Sent from a server to a client signaling that the client should send its request to another server whose reference will be embedded in redirection response.

In an asynchronous replication one server acts as the master, while one or more other servers act as slaves. The master server writes updates to log files, and maintains an index of the files to keep track of log rotation. These logs serve as records of updates to be sent to any slave servers. The slave receives any updates that have taken place, and then blocks and waits for the master to notify it of new updates. These replication functions are provided by the replication servers of different vendors.

5 Service Orchestration and Mobile Application Scenarios

Composite services provide standardized interfaces for state transfer between basic services. The first step in SORWiM Services was to create the stateless basic services as Event, Messaging, Location and Redirection, and their descriptions (WSDL) as explained above. The second step was to aggregate the functionality provided by each independent service and create the composite services which receive client requests, make the required data transformations and invoke the component web services.

The basic services and their usage in service composition are shown in Fig 6 using two example composite services, Location Based Redirection Composite Service (LBRCS) and Context Aware Notification Delivery Composite Service (CANDCS). To evaluate the development of mobile applications on SORWiM, TestPad was implemented to emulate several wireless application scenarios. The TestPad screen for basic services is shown in Fig 7(a).

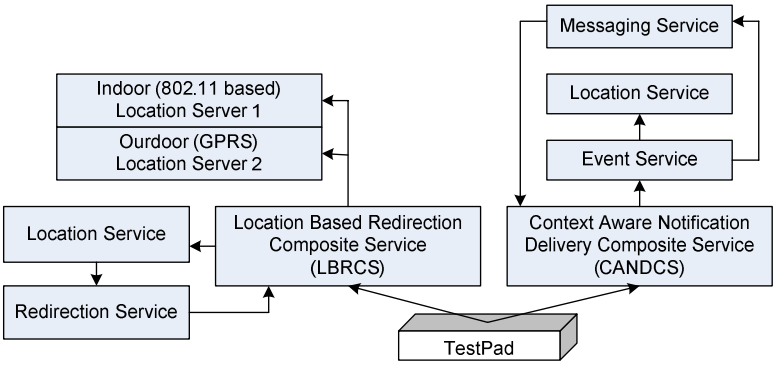


Fig. 6. Two example composite mobile services in service orchestration

LBRCS maintains transparent redirection according to changes in the mobile location using basic Location and Redirection services. This service first obtains the device location (Location Service) and redirects to the available addresses (Redirection Service). We will give a scenario to explain how LBRCS can be used in a mobile application. In this scenario, there are three locations. These are the user's home and user's office which both are covered by a Wireless LAN, and the outdoor location between the home and office where GPRS connection is available. In all these three locations the user can connect to these networks using Pocket PCs or laptops. When the user has left his/her office and moved out from the office WLAN coverage area, his mobile device keeps connected to SORWiM through the GPRS connection. The WLAN-to-GPRS handover is generally initiated by the mobile user. In our approach, the mobile user sets the location information based on the local measurement of WLAN signal quality. When LBRCS decides that a WLAN-to-GPRS handover is required, it sends a handover-required message and an available endpoint (IP, port number, and service name) of the server to the mobile device. The mobile user then performs a handover and switches from the WLAN connection to the GPRS connection. When the user arrives in home, and discovers a wireless access point, the mobile device sends the location information to LBRCS through the GPRS connection. SORWiM then sends the response of the endpoint of the server that is accessible in the office. The user performs a handover again and switches from GPRS to the WLAN connection again. The TestPad screen for this last scenario is shown in Fig 7(b). The user sets the new location as Home and gets a response message from SORWiM. The response is a new connection end point for the WLAN network. The old and new end points are also shown in the lower part of the screen in Fig 7(b).

CANDCS which composes the services of Messaging, Event and Location, notifies all registered users who are interested in a specific event. Imagine the following scenario using this composite service: Haluk is waiting for Bora's arrival and types in his Pocket PC as "notify me as soon as Bora appears on the campus". This message is sent to SORWiM using the Pocket PC's window shown in the upper part of the screen in Fig 7(c). When Bora arrives and enters the campus, immediately a notification is sent to Haluk's Pocket PC as a message or SMS as shown in the lower part of the screen in Fig 7(c).

(a) Basic services: Event, Message, Location and Redirection (ReDR)

(b) Location Based Redirection Composite Service (LBRCS)

(c) Context Aware Notification Delivery Composite Service (CANDCS)

Fig. 7. Basic services and composite services TestPad screens

CANDCS makes a decision using three basic services for requests similar to "I want to be registered for notifications about whether person X is in Y location". The Event Service provides an interface for registering the notifications. The Location Service gives location information and can query the nearby objects as shown in Fig 7(a). The Messaging Service that has the ability of sending different types of messages is responsible for delivering the notification messages.

6 Performance Evaluation of SORWiM

There are a number of research studies for the performance evaluations of web services and SOAP compared to middleware technologies, including Java RMI, CORBA. In these studies, a comparison is made based on the performance of a web service implementation that calls a simple method and an RMI, CORBA, or other implementation that calls the same method [22], [23], [24], [25], [26], [27].

We performed similar benchmarking tests to measure the performance of SORWiM to see if its performance is acceptable for a wireless middleware. The technologies used in this study were: Jakarta Tomcat 5.0.25, Axis 1.2, Java RMI from Java 1.4 SDK. The server specification is Intel Pentium M Processor 1.6 GHz 591 Mhz. 504 MB RAM. The client is an HP IPAQ h6340 Pocket PC. The Messaging Service in SORWiM is implemented using two middleware technologies, RMI and Web Service. The remote interface for performance tests is given below:

```
public interface MessagingService extends Remote {
  public String getMessage(String from)
     throws RemoteException;
  public String sendMessage(String to, String from,
     String  subject, String content, String type)
     throws RemoteException;
}
```

The single request and response CPU-times were measured for both the client Pocket PC and the server. A sample of our initial test results is given in Table 1.

Table 1. The server and client average CPU times for Web Service (WS) and RMI implementations of the Messaging Service methods, getMessage and sendMessage

Method	Calls	CPU Time [ms] Server WS	CPU Time [ms] Client WS	CPU Time [ms] Server RMI	CPU Time [ms] Client RMI
getMessage	10	30	80	10	20
sendMessage	10	50	180	20	30

Our test results and other research studies showed that web services implementations performed slower than a Java RMI implementation. The large amount of XML metadata contained in SOAP messages is the main reason that Web services will require more network bandwidth and CPU times than RMI. We have concluded that although the web service implementations are slower, the performance is acceptable for many real time operations of a wireless middleware. The performance can be made better and improved if the following issues are known and taken into consideration in middleware server designs.

All numeric and other data in web services are converted to text. Meta-data, defining structure, are provided as XML mark-up tags. XML parsers allow client and server implementations to construct their distinct but equivalent representations of any data structures The use of HTTP, and XML text documents, supports increased interoperability but also represents a significant increase in run-time cost for web service solutions as compared with Java-RMI solutions. The XML formatted documents are inherently more voluminous than the binary data traffic of the other approaches. More data have to be exchanged across the network, and more control packets are required.

When considering performance alone, web services provide value when the overhead of parsing XML and SOAP is outweighed by the business logic or computation performed on the server. Although web services generally don't provide value in performance, but do provide a convenient way to provide user interface, automatic firewall protection (because they use HTTP for transport and HTTP traffic can normally pass through firewalls), mobility and heterogeneity of applications, transparent proxies, and thin clients. The most natural designs for distributed objects are easy to use but scale poorly, whereas web services have good scaling properties.

The performance studies in literature generally measure RPC-style communication and do not consider the possibilities of document-oriented designs that demonstrate the strengths of web services [28]. Web services are not intended to be used RPC-style like

other distributed object technologies. Web services provide a literal encoding that can be used in a document-centric paradigm rather than an RPC-centric one. If the document-centric nature of web services is used in implementations, web services can outperform other traditional implementations when compared with an RPC-centric approach. According to [28] the pure-object RMI or CORBA implementation is faster for small batches of documents and low-latency networks, but performance degrades rapidly with larger batches and higher latency. The web services have a high initial cost, but show little or no change with larger batches. Higher latency creates a greater initial cost, but performance is still independent of batch size. As latency increases, the performance benefits of the document-oriented approach increase significantly. This is relevant when in some real world wireless communication scenarios, latency may even be minutes, or hours, as for disconnected or asynchronous operations.

7 Conclusion

In this paper the design issues of a SOA based reflective wireless middleware, SORWiM have been presented. We will work on the research and development problems of different mobile location based services (LBS) applications based on service oriented computing using the basic and composite services provided by the middleware. SORWiM currently has four basic services and two composite services which can be used as building blocks of such wireless applications. We need develop some more new basic and composite services for more complex mobile applications and LBS.

The integration of SORWiM and WiPoD will be also one of our next projects. The WiPoD client application currently has a decentralized architecture and does not need a server. It will be connected to the SORWiM server to provide the indoor location information for the mobile user.

References

1. Schantz, R., Schmidt, D.: Middleware for Distributed Systems: Evolving the Common Structure for Network-Centric Applications. In Encyclopedia of Software Engineering, J. Marciniak, J., Telecki, G., (Eds): John Wiley & Sons, Inc., New York (2001)
2. Schmidt, D.: Middleware for Real-Time and Embedded Systems. Communications of the ACM, Vol. 45, No: 6 June (2002)
3. Papazoglou, M. P., Georgakopoulos, D.: Service-Oriented Computing. Communications of the ACM. Vol. 46, No: 10 (2003)
4. Papazoglou, M. P., Heuvel, W. J. van den: Service Oriented Architectures: Approaches, Technologies and Research Issues. The VLDB Journal (2005) Available at: http://infolab.uvt.nl/pub/papazogloump-2005-81.pdf
5. Mascolo, C., Capra, L., Emmerich, W.: Mobile Computing Middleware. Lecture Notes In Computer Science, Advanced Lectures on Networking, Vol. 2497. Springer-Verlag (2002) 20–58
6. Vaughan-Nichols, S. J.: Wireless Middleware: Glue for the Mobile Infrastructure. IEEE Computer, Vol. 37, No. 5 (2004) 18–20
7. Roman, M., Kon, F., Campbell, R.: Reflective Middleware: From your Desk to your Hand. IEEE Communications Surveys, 2 (5) (2001)

8. Mascolo, C., Capra, L., Emmerich, W.: Principles of Mobile Computing Middleware. In Middleware for Communications, Wiley (2004)
9. Heinzelman, W. B., Murphy, A. L., Carvalho, H. S., Perillo, M. A. Middleware to Support Sensor Network Applications. IEEE Network, January/February (2004) 6 – 14
10. Yu, Y., Krishnamachari, B, PrasannaIssues, V. K.: Designing Middleware for Wireless Sensor Networks. IEEE Network, January/February (2004) 15 – 21
11. Sen, R., Handorean, R., Roman, G.-C., Gill, C.: Service Oriented Computing Imperatives in Ad Hoc Wireless Settings. Service-Oriented Software System Engineering: Challenges And Practices, Stojanovic, Z. and Dahanayake, A., eds., Idea Group Publishing, Hershey, USA, April, (2005) 247 – 269
12. Thanh, D., Jørstad, I.: A Service-Oriented Architecture Framework for Mobile Services. IEEE Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/ELearning on Telecommunications Workshop, September (2005)
13. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, M., Nielsen, H., Thatte, S., Winer, D.: Simple Object Access Protocol 1.1. , available at http://www.w3.org/
14. Chinnici, R., Gudgina, M., Moreau, J., Weerawarana, S.: Web Service Description Language (WSDL), version 1.2. Technical Report (2002)
15. W3C. UDDI Technical White Paper. Technical Report (2000)
16. Andrews, T. et al. Business Process Execution Language for Web Services (BPEL4WS), ver. 1.1. Technical Report (2003)
17. Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
18. Schmidt, D., Stal, M., Rohnert H., Buschmann, F. Pattern-Oriented. Software Architecture: Patterns for Concurrent and Networked Objects. John Wiley (2000)
19. PocketBuilder: http://www.sybase.com/products/developmentintegration/pocketbuilder
20. Apache Axis web site: http://ws.apache.org/axis/
21. Gümüşkaya, H., Hakkoymaz, H.: WiPoD Wireless Positioning System Based on 802.11 WLAN Infrastructure. Proceedings of the Enformatika, Vol. 9 (2005) 126–130
22. Davis, D., Parashar, M.: Latency Performance of SOAP Implementations. IEEE Cluster Computing and the Grid (2002)
23. Demarey, C., Harbonnier, G., Rouvoy, R., Merle, P.: Benchmarking the Round-Trip Latency of Various Java-Based Middleware Platforms. Studia Informatica Universalis Regular Issue, Vol. 4, No. 1, May (2005) 724–
24. Elfwing, R., Paulsson, U., Lundberg, L.: Performance of SOAP in Web Service Environment Compared to CORBA. in APSEC. IEEE Computer Society (2002) 84–
25. Juric, M. B., Kezmah, B., Hericko, M., Rozman, I., Vezocnik, I.: Java RMI, RMI Tunneling and Web Services Comparison and Performance Analysis. SIGPLAN Not., vol. 39, no. 5, (2004) 58–65
26. Gray, N. A. B.: Comparison of Web Services, Java-RMI, and CORBA Service Implementations. Fifth Australasian Workshop on Software and System Architectures (ASWEC 2004), Melbourne, Australia, April (2004)
27. Juric, M. B., Rozman, I., Brumen, B., Colnaric, M., Hericko M.: Comparison of Performance of Web Services, WS-Security, RMI, and RMI–SSL. The Journal of Systems and Software 79 (2006) 689–70
28. Cook, W. R., Barfield, J.: Web Services versus Distributed Objects: A Case Study of Performance and Interface Design, Proc. of the IEEE International Conference on Web Services (ICWS), September 18-22 (2006)