Integrating the Enterprise Service Bus in the Service Oriented Architecture for Middleware Extension

Guillaume Koum[1], Tam Sangbong[2], Raoul M. Tsamo[2],
Emmanuel Etoundi[2] and Augustin Yekel[1]

[1] Ecole Nationale Supérieure Polytechnique, Université de Yaoundé,
P.O. Box. 8390 Yaoundé, Cameroon
[2]Soft-Tech International Inc,
P.O. Box. 15412 Douala, Cameroon
g_koum@yahoo.fr; tam@soft-techint.com, rtsamo@yahoo.com; etoundiaboa@yahoo.fr;
a_yekel@yahoo.fr

Abstract. Our principal concern lies in designing and implementing a model of integration and communication of services within SOA environment, elaborated enough to allow several services to use a single input/output GSM (Global System for Mobile Telecommunication) channel. This model must offer a high flexibility in integrating new services (or new applications). This model is based on the characteristics of SOA and on communication of type publication/subscription from Event Driven Architecture (EDA).

Keywords: Service Oriented Architecture (SOA), Enterprise Service Bus (ESB), Event Driven Architecture (EDA), middleware.

1 Introduction

SOA is the philosophy that views software systems as multiple services connected to each other. In the context of SOA, a service is simply an autonomous application that runs and communicates to other services by exchanging messages. But an implementation of SOA suggests the use of an Enterprise Services Bus (ESB) because of complexity in managing nxn pairs of communication that could exist if we have n services. The goal of this article consists in designing and implementing an ESB that enable several services to use the same GSM channel for input/output and offers a high flexibility of integration of new services.

The ESB is something that plays a role of router between the services. More explicitly, the bus ensures the transport of messages between the various services which are connected to in order to carry out business functionalities. Because the data used by each service are not always in the same format, the ESB consequently needs to have a function of transformation, data conversion between the services. Moreover, as a router, it will have to ensure the connectivity and the transport of messages. These functionalities are in fact the necessary sub-set. For these purely functional aspects and since we are in a distributed environment, it would be necessary that this bus supports the needs for rise in load, huge volume of data, safety and reliability of the messages, also real time, etc.

Advances in Computer Science and Engineering.
Research in Computing Science 23, 2006, pp. 163-171

Received 07/07/06
Accepted 04/10/06
Final version 13/10/06

The communication model used for these services is of type event-driven. So, we will use the Event Driven Architecture (EDA) that proposes the development frameworks we have in the market. EDA is the type of architecture in which all the components are communicating to others by sending events. There is the publisher, the receiver and/or the broker of events. The publisher publishes the event; the receiver subscribes to the event before and specifies the action to be taken in the case of this event. The broker manages events. As such, it sends it to the good receiver after receiving it from the publisher. Event is like a signal. It naturally has its data. For example, you have a new mail is an event which has the mail content as its data. So, event transports simultaneously its data. Events are mostly caused by changing some data.

2 Our Context

In our case, connections to the bus are done with a SMS (Shorts Message Service). SMS is a technology in the GSM that allows sending small text to a mobile phone. SMS is used in multiple applications nowadays to ensure mobility of the users. [16] is the standard that specifies the format of SMS.

The figure 1 shows the conceptual architecture of this bus based on the use of SMS messages.

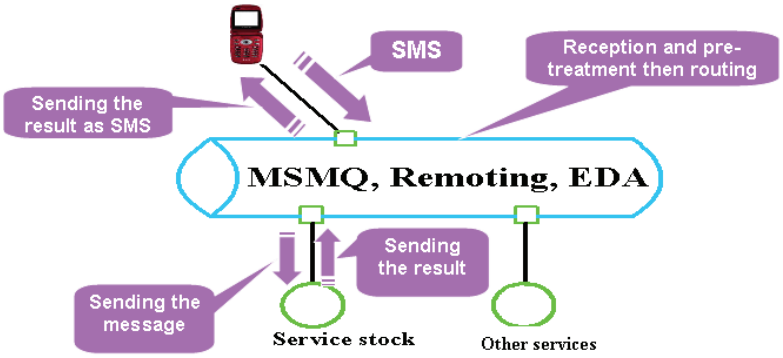

Fig. 1. Conceptual architecture.

The application that we built on this architecture meets the need of mobility of the users above. It will be possible to whomever to reach the information system (IS) of his company, with a certain number of rights naturally, to collect desired information. A specific case would be for example to give the possibility to an itinerant salesman to query the IS of his company in order to get for example the stock of a certain product. This query will be done from a mobile phone. Specifically, he will send a SMS to the system. Once the SMS is received, the ESB will send it to the suitable service which has been given the responsibility to analyze it and to carry out the required processing. At the end of the process the result from this service will

automatically be returned to the applicant. The applicant will at this time receive it directly on his mobile phone.

This form of communication is more advantageous because the user continues to have access to the IS of his company at anytime and anywhere (as long as the GSM network permits it). He is able to get the real time information which is essential for his business! In those IS where there are multiple updates, the user always has the possibility of getting actual data (for example case in the stock exchange place).

Our objective is to give to users the access to a multitude of services while passing by the same channel as figure 2 shows. Because there is only one channel and many services will access it, it is now necessary to have something (an application) that will manage this resource. We call it: a Manager. Its aim is to allow services to receive messages dedicated to them. These services can run as standalone applications. These services can be built by other teams; and, in order to use the channel, they will easily be grafted on the system. The mode of communication between these services and the manager should thus be by exchanging messages: a communication base on program interfaces would lay more problems and would be more constraining.

Indeed, while communicating by the publication of the interfaces, the services and the manager will have mutually to know themselves; this would oblige the manager to know the interfaces of all the services. One could imagine the amount of difficulties that arises when one wants to add a new service. Thus, the manager must be modified each time a new service is added. This adds maintenance problems and shows a considerable lack of flexibility in adding and removing services. In addition, it also poses a real problem of performance because the manager will be blocked at each service call since the communications would need to be synchronous in that case. If the performance is reduced then is also the quality of service (QoS). The huge volume of solicitation of the manager (what will be the case besides) will lead directly to freezing system what puts everything in danger!

Compared to the above discussion, we built a single manager that takes directly into account installation of the new services into the system. This manager is completely autonomous.

However, the access to the system needs to fulfill some criteria of security in order to guarantee the safety and the integrity of the system. For this purpose, it will be necessary to define rules of authentication and authorization. And given the fact that some data are confidential in the IS, it would be interesting to apply some concepts of cryptography and encryption to these data. These aspects were not largely applied here because of the prototype case of this application. But they must always be in mind once in a distributed or opened environment.

The figure below shows the necessary infrastructure that enables the system to operate. But it is interesting to present the models of communication used to make components communicate to each other. The dotted lines in this figure delimit the various circuits that exist for the transport of the requests from the mobile phone to the services; then answers from the services to the requester mobile phone. This figure shows also the technological options in circuits 2 and 3:

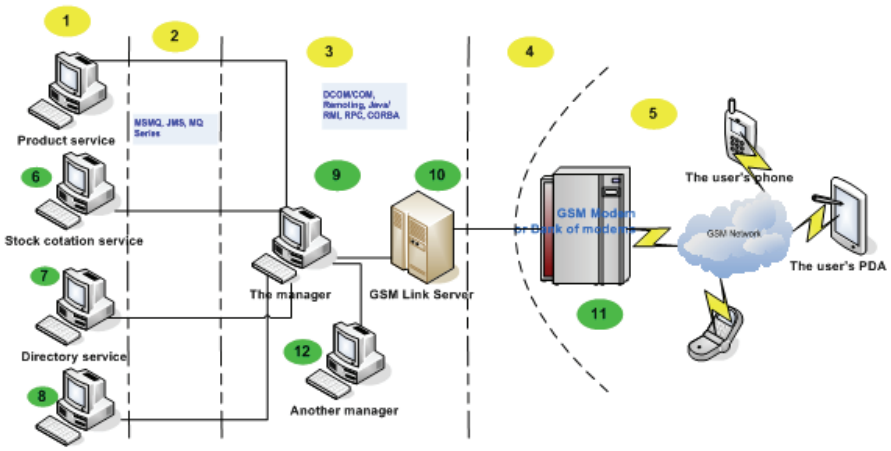

Fig. 2. Physical architecture and technological options.

The circuits sections in question are marked by yellow rounds 1, 2, 3, 4 and 5. They are described below:

1. This portion contains the various services which can execute requests from the users. New services can be integrated by using the infrastructure offered into 2.
2. This circuit represents the channel used so that these services communicate with the Manager. An analysis enabled us to note that this communication is not only of client/server type but should also be made by exchanging messages in order to meet the objectives referred above. The technological options are as follows: MSMQ (Microsoft Message Queue), MQ Series, and JMS (Java Message Queue). Services will subscribe to the events of arrived messages on these queues. So, the manager will write message on the queue. But the manager also subscribe to the same event on another queue in other to get results from services.
3. This circuit represents the channel used so that the Manager communicates with the GSM Link Server. It is a traditional type of communication between applications. Hence the use of directed object middlewares. The technological options are as follows: DCOM/COM+ (Distributed Component Object Model), Remoting, CORBA (Common Object Request Broker), Java/RMI (Remote Method Invocation). But since the GSM Link Server generates also events and that these events must be captured by the Manager, we implemented a model of EDA architecture between the two services that enables capturing events produced by GSM Link server. All communications are based on events and data are simply serializable objects that pass through the middleware.
4. This channel is used for the connection of the Modem to the serial port of GSM Link Server.
5. It is the GSM network which communicates with the GSM Link Server through the use of the Modem that serves as the interface with. The modem is connected to a COM port of the computer. And when an SMS arrives the modem writes data to the COM port and the listening program can then read the incoming data.

The data received on this port are analyzed against [16]. After this analysis, the program composes an object that is made up of the incoming SMS elements as stated in [16]. The system then fires an event and includes the composed object as the event's data. The event constitutes what the manager will the event.

On the figure above, applications involved in the system are identified by green rounds and each machine conceptually consists of one application.

Rounds 6, 7 and 8 contain respectively the Product service, Stock quotation service, the directory services. Rounds 9 and 12 are managers who send/receive messages for services; they consume also the services offered by the GSM Link Server (round 10). This machine has the functionalities that enable it to manage a bulk of modems represented by the component numbered 11.

The preceded sections enable us to show the physical architecture necessary to the implementation of our system having in mind the needs of mobility and real time. This architecture enables us to control the physical infrastructure and the ways that the various messages will get to their recipients. The following section presents the implementation of the prototype which turns for the execution of the main scenario by requesting a service from a SMS.

3 Implementation of the Prototype

3.1 Tools and Languages for Development

1. Our application is a multi-tiers architecture. We used:
2. Microsoft SQL Server 2000 as the Database Management System;
3. Microsoft MSMQ™ for the exchange of messages between the services and the Manager;
4. Microsoft Visual Studio.NET with the Microsoft NET Framework as the development environment;
5. The implementation language used for the business objects is Visual C#.NET;
6. Microsoft NET Remoting for the communication of the Manager and GSM Link Server;
7. A modem for the sending/receiving SMS.

3.2 Tests

The correct functioning of the system consists in launching GSM Link Server initially. This application can run in background. It communicates directly with the modem which can be connected at any time on the serial port of the computer it is running on.

After starting this server, one then now launches the Manager who will undertake to communicate with the server. The starting of the service is characterized by the presence of a notification icon Windows taskbar.

The launching of the Modem is done from the Manager; what shows the remote control we have on the Modem which is the material from which events are all from. Events from the modem can be displayed as follows:

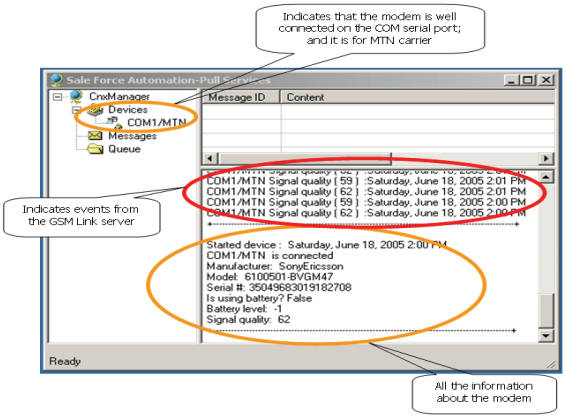

Fig. 3. The Manager main form.

The business services can be launched at any time.

We present the execution of the principal scenario described in the preceding sections:

1. Composition of the SMS by a user starting from his mobile phone. For example, the query to be sent is: "# STOCK#PRD00001 # " i.e. "I want the quantity in stock of the product code PRD00001.(STOCK: ID of the service, PRD00001: Produced ID, #: separator)
2. Reception by the modem, then by GSMLink Server;
3. Generation of the event corresponding to the reception of the SMS; and transfer of the data;
4. Reception of the data transmitted by the Manager and displaying; then insertion in the messages queue;

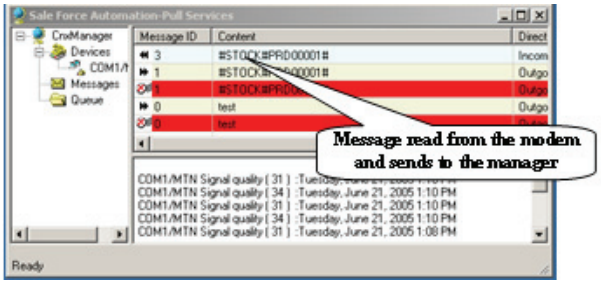

Fig. 4. Reception of the SMS by the Manager.

5. Automatic reception of the message by the concerned service and displaying, processing follows;

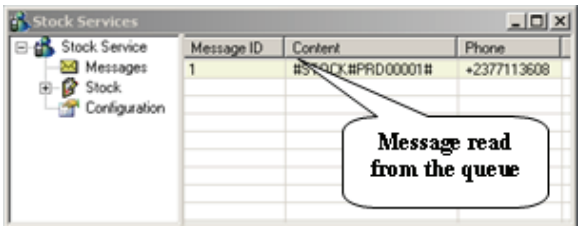

Fig. 5. Reading the message from the message queue.

6. Insertion of the result in the messages queue;
7. Automatic reception of the message by the Manager, then sending of the SMS to the server (GSM Lin Server);
8. Reception and sending of the SMS by the server (GSM Lin Server);
9. Reception of the result by the user through the portable. For example: # PR00001#45000#60 # From Sale Forces Automation # (PRD00001: Produced ID whose stock is required, 45000: Unit Price, 60: Quantity in stock, From Sale Forces Automation: For the signature of the source).

4 Conclusion

Services oriented architectures are today one of the elaborate solutions brought to solve the problem encountered during the integration of the components deployed in the different application servers. The services offered by this architecture are built on standards and because of that purpose could thus be consumed from any platform. This characteristic offers enormous advantages at the level of systems integration as well as at the level of constructing new solutions. The construction of new solutions will take less time because it is possible to compose a new service from those which already exist. This decreases considerably the T2M (Time-to-Market). However, it poses a small problem when one wants to integrate real time aspects in the solutions of company because the services are connected to each other and communicate in a traditional way (question/answer). This style of solution badly supports real time. However, company's IS are generally in a constant state of changes. Each change of state can be seen as an event. The fact that the quantity in stock of a product reaches a threshold is an example. This event can then automatically orchestrate a whole of consequent. Thus EDA are styles of architectures which implement this type of process offering solutions on a real time basis. Consequently, the coupling of the SOA with the EDA offers a more modern architecture than traditional ones.

In the context of the integration of services, the use of point-to-point can easily lead to a mesh network of services. This type of network is difficult to manage or maintain because of the high number of services and connections. The buses of services bring to this problem not only flexibility in the connection of the services but also reliability in sending message safely, the management of the priorities and the scalability (which is the capacity to be extended), etc. The research made in this direction enabled us to enrich the design of our system which is indeed a kind of bus.

Our research in this paper consisted in putting together the concepts of SOA, EDA, EAI and middlewares. These concepts enabled us to develop a model of bus of services allowing services considered as back-end to be able to use only one input/output GSM channel. This bus allows a flexible integration of new services (the services back-end). This system allows the services considered as back-end to give answers to queries carried out by the users of a given IS.

The problem we faced at the beginning of this work was that it was not possible that several services (or applications) use only one modem at the same time (or a bulk of modems). But now that is quite possible. This system can be distributed at the level of the Intranet as well as over the Internet i.e., the modem can be controlled from anywhere! The immediate economic impact is the resource sharing (modem in this case) and the market broadening.

With this purpose, we have exploited SOA, EDA and especially we could enable EDA to be distributed (and we called it d-EDA for Distributed Even Driven Architecture) by developing a certain architecture of classes on a middleware typically object-oriented.

The d-EDA has many advantages since from a network standpoint which generates events, several applications running on various machines can now process these events differently. These types of architectures are useful for example in Telecom, in hospitals (alarms of the medical equipment, etc), and in any field where one uses a device that generates events to be treated by an application.

The perspectives for this work go in the implementation of all the functionalities offered by a bus of services since we built only prototype not a complete bus. We used it only as a router with not as much intelligence as needed. We will enrich the system by the introduction of a messages broker and an intelligent routing process in order to be able to compose the results coming from several services. We will have to also integrate into this bus the capacity to receive Simple Object Access Protocol – SOAP messages. Naturally, it would be interesting to deal with the web services.

References

1. Duane Nickull et al.: Service-Oriented Structure. Whitepaper, Adobe Systems, Inc. http://www.adobe.com
2. Keen Martin et al.: Patterns: Implementation and SOA using year Enterprise Service Drunk. RedBook, IBM®, http://www.ibm.com
3. Mark Andrei et al.: Patterns: Service-Oriented Structures and Web Services, RedBook, IBM®, http://www.ibm.com
4. Hans-Erik Eriksson and Magnus Penker: UML Toolkit, John Wiley & Son' S, Inc, 1998.
5. J.P. Mueller: COM+ Developper' S Guides, Osborne, McGraw-Hill, 2000.
6. Davit S. Platt: Understanding COM+, Microsoft® Press, 1999.
7. Zoran Stojanović: A Method for Component-Based and Service-Oriented Software Systems Engineering. Doctoral Essay, Delft University of Technology, The Netherlands, January 2005, and ISBN: 90-9019100-3.

8. Raoul M. Tsamo: E-Intelligence integrated into the concept of e-marketplace in a distributed environment. Application to an e-procurement system. Master of engineering's end of course thesis in computer science. National Advanced Polytechnic School. Cameroon. 2001.
9. Service Oriented Structures Solution Accelerator Guide, www.bea.com, May 2005
10. Jeffrey R. Shapiro: SQL Server™ 2000: The complete reference, Osborne/McGraw-Hill, 2001.
11. Cesare Pautasso: Message Oriented Middleware (MOM), Computer Department Sciences, Federal Swiss Institute of Technology (ETHZ), http://www.iks.inf.ethz.ch /, April 2005.
12. William Tse: Enterprise Integration Application, www.cs.ucl.ac.uk/staff/W.Emmerich/lectures/3C05-03-04/EAI.pdf, May 2005
13. David Sprott and Lawrence Wilkes: Understanding Service-Oriented Structures, CBDI Forum, January 2004, http://www.cbdi.org/.
14. Distributed Systems, ITEC801. http://www.comp.mq.edu.au/units/itec801/lectures/. June 2005
15. Cyrille MBASSI: Design and implementation of a directed architecture service: News Aggregator Full Service. Master of engineering's end of course thesis in computer science. National Advanced Polytechnic School. Cameroon. 2005.
16. Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS) Point-to-Point (PP) (GSM 03.40). European Telecommunications Standards Institute.