A Service Oriented Architecture Framework for Collaborative Services

Ivar Jørstad
Norwegian University of Science and Technology, Dept. of Telematics, O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
ivar@ongx.org

Schahram Dustdar
Vienna University of Technology, Distributed Systems Group (DSG), Information Systems Institute, A-1040 Wien, Argentinierstrasse 8/184-1, Austria,
dustdar@infosys.tuwien.ac.at

Do Van Thanh
Telenor R&D & Norwegian University of Science and Technology
Snarøyveien 30 N-1331 Fornebu, Norway
thanh-van.do@telenor.com

Abstract

The demand for flexible, efficient and user-friendly collaborative services is becoming more and more urgent as the competition in the current market-oriented arena is getting more intense and fiercer. Enterprises have to be more dynamic in terms of collaboration with partners and even competitors. The Service Oriented Architecture is a promising distributed computing paradigm offering solutions that are extendible, flexible and compatible with legacy systems. This paper proposes and investigates the use of SOA in the construction of collaborative services. The paper starts with a short introduction of the Service Oriented Architecture and then gives a description of collaborative services. A generic model of a collaborative service is analysed to identify the basic collaborative functions. A Service Oriented Architecture Framework for collaborative service is presented.

1. Introduction

Organizations constantly search for innovative applications and services to improve their business processes and to enrich the collaborative work environments of their distributed and mobile knowledge workers. It is increasingly becoming apparent that a limiting factor in the support of more flexible work practices offered by systems today lies in:

1. Their inherent assumptions about technical infrastructures in place (hardware, software, and communication networks),
2. Their assumptions about interaction patterns of the users involved in the processes.

Emerging new ways of flexible and mobile teamwork on one hand and dynamic and highly agile (virtual business) communities on the other hand require new technical as well as organizational support, which current technologies and infrastructures do not cater for sufficiently. Service Oriented Architecture (SOA) is a new paradigm in distributed systems aiming at building loosely-coupled systems that are extendible, flexible and fit well with existing legacy systems. By promoting the re-use of basic components called services, SOA will be able to offer solutions that are both cost-efficient and flexible. In this paper, we propose to investigate the feasibility of using SOA in the construction of innovative and advanced collaborative services. The paper starts with a short introduction of the Service Oriented Architecture and then gives a description of collaborative services. A generic model of a collaborative service is analysed to identify the basic collaborative functions. A Service Oriented Architecture Framework for collaborative service is presented thereafter.

2. Overview of the service oriented architecture

There are currently many definitions of the Service Oriented Architecture (SOA) which are rather divergent and confusing. The World Wide Web consortium [1] defines as follows:

A Service Oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following properties:

Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)
0-7695-2362-5/05 $20.00 © 2005 IEEE

IEEE COMPUTER SOCIETY

- *Logical view: The service is an abstracted, logical view of actual programs, databases, business*
- *Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.*
- *Description orientation: A service is described by machine-processable meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.*
- *Granularity: Services tend to use a small number of operations with relatively large and complex messages.*
- *Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.*
- *Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint.*

A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities. To be used, a service must be realized by a concrete provider agent.

The mentioned definition is very generic and we choose to adopt the definition inspired by Sayed Hashimi in [2]:

In SOA, software applications are built on basic components called services.

processes, etc., defined in terms of what it does, typically carrying out a business-level operation.

A service in SOA is an exposed piece of functionality with three properties:

1. *The interface contract to the service is platform-independent.*
2. *The service can be dynamically located and invoked.*
3. *The service is self-contained. That is, the service maintains its own state.*

There are basically three functions that must be supported in a service-oriented architecture:

1. Describe and Publish service
2. Discover a service
3. Consume/interact with a service

3. From groupware to collaborative services

Groupware systems have the potential to offer and consume services on many levels of abstraction. Consider a typical scenario of team work: (Distributed) Team members collaborate by using messaging systems for communications. In most cases the “work-space” metaphor is used for collaboration. This means that team members have access to a joint workspace (in most cases a shared file system), where files (artifacts) and folders may be uploaded and retrieved. In many cases (mobile) experts are part of such teams and their workspaces. One can argue that a workspace can be seen as a community of team members working on a shared project or towards a common goal. The aim of Groupware systems is to provide tool support for communication, collaboration, and to a limited extent, for coordination of joint activities [3]. Collaborative Services or Collaboration Services facilitate collaboration between people, teams or organisations.

Current collaborative Services can be classified as follows:

- *Knowledge and resource sharing:* In collaboration it is crucial to share knowledge and resource together. By share it is meant several things:
    - *Presentation:* the same knowledge or resource is presented such that all the collaborators can view, experience and interpret it in the same way.

IEEE COMPUTER SOCIETY

    - *Generation and modification:* All the collaborators should be enabled to generate and modify knowledge and resources in such way that consistency and integrity are preserved
    - *Storage:* the knowledge or resource must be stored safely without affecting the availability.
- *Communication and personal interaction*: In collaboration, communication and interaction between collaborator are crucial to avoid misunderstand and mismatch. Communications can be classified in several ways:
    - *Synchronous* (e.g. telephony, chat, sms, etc.) versus *asynchronous* (mail, newsgroup, forum, etc.)
    - *Channels:* Voice, text, multimedia
- *Virtual rooms:* All the collaborators shall share the same context in the same manner as they are located in the same room. Examples are Team room, Electronic Classroom.
- *Organisation management:* It is also necessary to manage dynamically the collaboration team itself and to distribute information about the team to all the members in an efficient manner. The services includes:
    - Project/ team management
    - Calendar, time scheduling, milestones, mail exploder.

It is worth noting that the collaboration "service" has larger granularity than the ones defined in SOA. In fact, a collaboration service can be mapped to a SOA software application that consists of multiple SOA services.

4. A generic model of collaborative services

Since the Service Oriented Architecture is currently the most promising architecture for building services, we propose to use SOA in the construction of collaborative services. We start with the modelling of a generic collaborative service and then map it to a SOA environment. A SOA framework will then be elaborated based on the findings identified from the mapping.

Figure 1 depicts the logical architecture of a collaborative Service that is used by three users. Following the model of a generic service proposed in [4], a collaborative service can be represented by four components: *ServiceLogic, ServiceData, ServiceContent* and *ServiceProfile*.

ServiceLogic is the program code that constitutes the dynamic behavior and provides the functions of a service. The logic of a mobile service can be subject to various distributions, as in any distributed system [5].

ServiceData are used in the execution of the service logic and reflecting the state of it. They are for example variable values, temporal parameters, register values, stack values, counting parameters, etc.

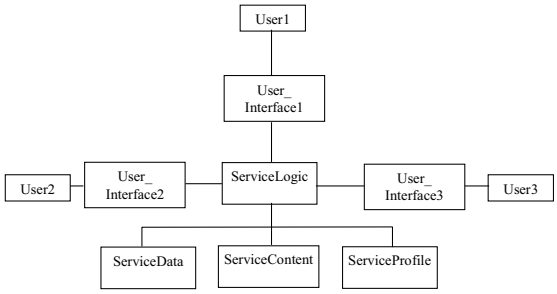

Figure 1. Logical architecture of a collaborative service

ServiceContent refers to data that are the product of service usage. For example it can be a document written in a word processor or the entries in a calendar. Service content can be produced or consumed by the user

ServiceProfile contains the settings that are related to the user or/and the accessing device.

Each user employs a *User_Interface* to collaborate with the other users via the Collaboration Service. The *User_Interface* can be a generic component that can be used to access several services like a browser. It could be a dedicated component that is especially built for a specific service. Each user can use different instances of the same implementation, e.g. different instances of Internet Explorer. These components can be referred to as *identical components*. They can also use different instances of different implementations. These components can be referred to as *equivalent components.*

To let several users to participate simultaneously, the *ServiceLogic* must be equipped with specific collaborative functions that we are going to examine successively.

IEEE COMPUTER SOCIETY

4.1 Locking mechanism

For knowledge and resource sharing services it is necessary mechanisms to prevent the corruption of knowledge and resources. These mechanisms are similar to the one in database systems, *locking mechanisms*. There are several types of locks that can be chosen for different situations:

- *Intent:* The intent lock (I) shows the future intention of a user to acquire locks on a specific resource
- *Shared:* Shared locks (S) allow several users to view the same resources at the same time; however, no user is allowed to modify it.
- *Update*: Update locks (U) are acquired just prior to modifying the resource
- *Exclusive:* Exclusive locks (X) completely lock the resource from any type of access including views

The majority of collaborative services will require a variety of locks being acquired and released on resources.

4.2 Presentation control

Quite often, the users want to experience the same resource together from their own computer, e.g. viewing the same document or the same presentation, listening to the same music song, etc. These resources are presented to the users by different applications such as word processor, presentation reader, etc. All the users may be allowed to manipulate these applications as for example scroll down, go to other page, etc. Alternatively, the control can be given to only one user. In any case, it is necessary to have a presentation control component that collects all the inputs from the different users and delivers them to the respective applications according the pre-selected presentation scheme. The outputs from the applications should also be controlled by this component. This component should also support different navigation devices such as mouse, scrolling button, joy stick, etc.

4.3 User Presence management

The user belonging to a collaborating organisation should be reserved the right to decide when to participate to a collaborating activity such as viewing a multimedia documentation, editing a document, etc. It is,therefore, necessary to provide a registration or login mechanism and deregistration or logout mechanism to the different activities. It should be also possible for the user to subscribe for different log services i.e. information about the dates and times of the different activities, information about the participants, the resources produced or modified by the activities.

4.4 Organisation management

There should also be a management function that allows the user in charge of the collaborative organisation to add, remove and assign rights to the participants. The responsible user can also define different collaborative activities. Each collaborative activity may incorporate different applications and contents. For example, in activity **workingGroup1**, a word processing with access to folder *working_group_1* is used together with chat. In activity **workingGroup2**, a presentation reader is used with SIP [6] (Session Initiation Protocol) IP telephony. In addition, it is useful to have other functions like calendar, time scheduler, milestones.

4.5 Communication means

In any collaboration, communication between the collaborators is decisive for the success. It should be possible to select the appropriate mean e.g. chat, email, sms, plain old telephony, voice IP telephony, multimedia IP telephony, etc. for each activity. To make things even easier, it should be possible to define an email exploder to send email to a group of persons, a telephone conference to initiate telephone call to a group of persons, etc.

5. SOA framework for collaborative services

In Service Oriented Architecture, the fundamental components are the services that applications are built upon. As depicted in Figure 2, a collaboration service **CollServ X** is an application consisting of a main component that invokes the **ServiceA_client** and the **ServiceB_client**. The clients in their turn will call their respective service. Alternatively, a service can be an orchestration of services as in the case of **CollServ Y**, which is an orchestration of the **ServiceC**, **ServiceD** and **ServiceF**. In the case where the Web service technology is used to implement the Service Oriented Architecture, there are many languages that can be used for the specification of the **CollServ Y** such as BPEL (Business Process Execution Language) or BPEL4WS (Business Process Execution Language for

IEEE COMPUTER SOCIETY

Web Services) [7], BPML (Business Process Management Language) [8].

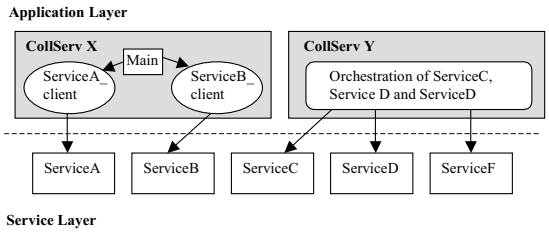

Figure 2. Examples of SOA collaborative service

In order to alleviate the construction of collaborative services, we propose the followings:

- To transform the basic collaborative functions identified in the previous section to SOA services.
- To gather them in a Collaborative Service layer which is available to all the developers.

As shown in Figure 3, a SOA Framework for Collaborative Services is hence obtained. The Collaborative Services Layer consists of the services: *Locking, Presentation Control, User Presence Management, Organisation Management and Communication Control*. A collaborative application can then be built based on these basic collaborative services and on other selected services. They can be either an application or an orchestration of services.

6. Conclusion

In this paper the Service Oriented Architecture is investigated and found feasible for the construction of collaborative services. The generic model of collaborative service is mapped to the Service Oriented Architecture. To alleviate the tasks of the developers, the basic collaborative services, *Locking, Presentation Control, User Presence Management, Organisation* Management and Communication Control are gathered into a Collaborative Service Layer and made available to the applications. A collaborative service can be built by composing or by orchestrating the collaborative services together with other services.

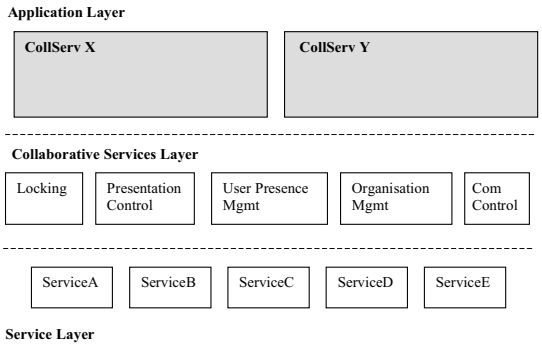

Figure 3. Service oriented architecture framework for collaborative services

7. References

[1] Booth, D., et. al. (editors), "W3C Working Group Note 11: Web Services Architecture", World Wide Web Consortium (W3C), February 2004, http://www.w3.org/TR/ws-arch/#stakeholder

[2] Hashimi, S., "Service-Oriented Architecture Explained", O'Reilly ONDotnet.com, August 2003, http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html

[3] Dustdar, S. Gall, H., Schmidt, R., "Web Services For Groupware in Distributed and Mobile Collaboration", 12th IEEE Euromicro Conference on Parallel, Distributed and Network based Processing (PDP 2004), A Coruña, Spain, February 11-13, 2004, IEEE Computer Society Press

[4] Jørstad, I., Thanh, van D. and Dustdar, S., "An analysis of service continuity in mobile services", Proceedings of the 2nd International Workshop on Distributed and Mobile Collaboration (DMC 2004), IEEE Computer Society Press, ISBN 0-7695-2183-5 (pp. 121-126), University of Modena and Reggio Emilia, Italy, June 14-16, 2004

[5] ITU-T X.901 | ISO/IEC 10746-{1,2,3,4}, "Open Distributed Processing Reference Model Part 1,2,3 AND 4"

[6] Rosenberg, J., et. al., IETF - Multiparty MUltimedia SessIon Control (MMUSIC) Working Group, "RFC 3261: SIP: Session Initiation Protocol",

[7] Curbera, F., et. al., "Business Process Execution Language for Web Services", Version 1.0,,, http://www-106.ibm.com/developerworks/library/ws-bpel/

[8] O'Riordan, D., "Business Process Standards For Web Services", published by Tect, 29 South Lasalle Str. Suite 520, Chicago, Illinois 60603, USA