SOA-based Generic Architecture for CSCW Systems*

Mario Anzures-García[1], Patricia Paderewski-Rodríguez[2], and Miguel J. Hornos[2]

[1] Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla,
Avenida 14 Sur y Avenida San Claudio. Ciudad Universitaria, San Manuel,
72570, Puebla, México
anzures@correo.ugr.es
[2] Departamento de Lenguajes y Sistemas Informáticos, ETS Ingeniería Informática,
Universidad de Granada, C/ Periodista Saucedo Aranda, s/n,
18071, Granada, Spain
{patricia, mhornos}@ugr.es

Abstract. In this paper, we present an SOA-based generic architecture for CSCW systems, directed towards overcoming the shortcomings of other architectural models when developing this type of system, as well as tackling the lack of a generic architecture in web service-based collaborative applications. Since it is based on SOA, the resulting application has the following characteristics: modularity, reusability, interoperability, scalability, flexibility, adaptability, robustness and efficiency. In addition, the services can be published or requested from any device (including mobile devices) independently of the language or platform used. The architecture supports the development of collaborative applications with group awareness, notification, consistency and security mechanisms. This allows us to change the group size, the roles played by the user and the interaction policies, and to add new services without the need to modify existing services. This proposal is the result of the study we carried out into the main architectural models and environments for developing collaborative applications (which shows that they focus on the interaction aspects of the systems and have been designed for very specific applications) and the analysis of the most representative web service-based collaborative applications (which reveals that they only solve specific problems and that there is no generic architecture).

1 Introduction

In the last two decades, Internet, the web and the enormous growth of different technologies (mobile technology, networking and protocols, database advances, graphics, etc.) have given rise to an evolution in the structure of the market and in the form in which enterprises are organized, allowing the emergence of a global market. This new market requires software systems which support, contribute and strengthen groupwork, and these systems must simultaneously be supported by models, methodologies, architectures and platforms that allow groupware applications to be

* This work is financed by the Spanish project CICYT (TIN2004-08000-C03-02).

developed in keeping with current needs. This development has been based on the use of different approaches, including object-oriented, component-oriented, aspect-oriented and agent-oriented ones. In recent decades, emphasis has been placed on the implementation of web-distributed collaborative applications which are component-based.

There has recently been an increase in the use of SOA to build distributed applications based on web service technology. SOA applications are built by combining network-available services. These services take part in distributed co-operative processes, exchange messages, coordinate their executions, and enable ubiquity, interoperability and integration between applications. The resulting cooperative systems are potentially self-configurable, ubiquitous, adaptive and robust since they allow the dynamic incorporation of alternative services and avoid simple failure points. A service-based software system automatically discovers services, negotiates to acquire them, and composes, binds, executes, and unbinds them for each execution. This mitigates software evolution problems [16, 17] since the system is created as a suitable composition of services for a set of particular requirements at a given time and every service evolves or is maintained independently.

For the development of collaborative systems, however, a generic architectural model is necessary which supports the three key characteristics of CSCW systems (i.e. communication, coordination and collaboration [6]) and which allows third-party components to be consistently integrated. For this purpose, we present an architectural proposal that in addition to providing a generic architectural model for the development of collaborative applications allows third-party components to be added and takes advantage of the benefits provided by SOA.

In Section 2 of this paper, we present an analysis of the main architectural models and environments for the development of collaborative applications, as well as the conclusions of a study on a series of web service-based collaborative applications. Section 3 shows our architectural proposal. The final section presents our conclusions and future lines of research.

2 Related Work

A wide range of applications, prototypes and products have been developed to support groupwork. Each groupware system has been designed to support a particular form of cooperative work or a specific range of cooperative work forms. There are also a wide variety of architectural models and environments which help us to develop collaborative applications, such as the ones analysed in Section 2.1. Moreover, in recent years, web services have been taken as the technological base for the development of CSCW applications. The conclusions of the study carried out on nine of the most representative web service-based applications of this type are presented in Section 2.2.

2.1 Models and Environments for Developing Collaborative Applications

Architectural models attempt to model the system as a group of components and relationships between them, e.g. Interactor Models [13], PAC [4], and MVC (Model-View-Controller) [11]. Extensions have also been proposed for CSCW systems, e.g. PAC* [2] and Dewan's generic architecture [5], and new models such as Patterson's taxonomy [18], COCA (Collaborative Objects Coordination Architecture) [15], Clock [10], Clover [14] and the architecture proposed with the methodology AMENITIES (A MEthodology for aNalysis and desIgn of cooperaTIve systEmS) [9]. One disadvantage of these models is that all except COCA and AMENITIES focus on the interaction aspects of the system. COCA, however, is mainly directed towards designing independent coordination policies, and the implementations carried out with AMENITIES are platform dependent since they are based on Jini and JavaSpaces. None of these architectures provide a consistent model which allows third-party components to be added.

The analyzed development environments do not provide all the aspects required to develop flexible, dynamic, robust, open and secure collaborative applications. We shall now indicate the main disadvantages of each of them. Although Groupkit [20] provides a library of components for the construction of multi-user interfaces, these are not easy to customize and cannot interoperate with each other. Habanero [3] enables their single-user applications to be converted into collaborative applications and collaborative applications to be developed with group awareness mechanisms, but extra effort is needed to implement different group awareness mechanisms and it is not platform independent. JSDT [1] provides a set of APIs to construct collaborative applications with group awareness mechanisms, but these applications are tightly coupled with the environment and dependent on it. COAST [21] allows document-based synchronous collaborative applications to be developed by means of a general architecture and the corresponding classes; however, it does not offer consistency mechanisms, and this can result in conflicts and loss of work. TOP [12] enables the development of applications using the definition of ten objects, but it does not establish information safety and consistency mechanisms. NESSIE [19] mainly focuses on group awareness mechanisms, leaving other aspects to one side. ANTS [8] facilitates the development of generic CSCW systems by providing monitoring and group awareness services and a three-layer architectural model; its main disadvantage is that it is based on JavaBeans, and this prevents it supporting other interfaces and programming languages. CoopTel [7] defines collaborative applications based on a model of components and aspects; however, its platform independence is limited, since it is implemented in Java and RMI. Most of the mentioned environments do not define a specific architectural model for collaborative applications, and those which do (with the exception of ANTS) are based on architectural models orientated towards the interaction aspects of the systems. Nevertheless, ANTS does not provide a consistent model for the addition of third-party components.

2.2 Web Service-based Collaborative Applications

In terms of the use of web services for the development of collaborative applications, we have analysed a representative group of nine applications. The analysis was carried out according to the three main characteristics of CSCW systems (communication, coordination, and collaboration) and to the implementation, architecture and technologies used in web services. From this study, we have obtained the following conclusions: firstly, there is no generic architecture for developing web service-based CSCW systems; secondly, the use of services generally offers the possibility of establishing asynchronous and synchronous communication, but it would be convenient to define a communication layer that allows the establishment of floor control, notification and group awareness mechanisms; thirdly, for coordination, it would be advisable to develop a session manager that allows a session to be created, joined or left in run time, in addition to providing a repository that serves as the basis for group awareness and notification mechanisms in order to ensure the consistency of shared information.

3 Architectural Proposal

We propose an SOA-based architecture for developing collaborative systems, directed towards overcoming the shortcomings of current architectural models. Our proposal provides the underlying advantages of SOA. Figure 1 presents the general outline of the architecture proposed, showing its main layers, modules and services: a Communication Layer; (2) Group Layer, which comprises the Session Management module, the Shared Control Access module and the Group Awareness and Register services; and (3) Application Layer.

3.1 Communication Layer

The Communication Layer is fundamental in collaborative systems because it supports the mechanisms that allow groupwork. These mechanisms enable the sessions to be implemented in such a way that users can join and leave a session and different coordination policies to be established; they also provide group awareness and facilitate security. This layer manages the communication between layers, modules and services by interchanging documents through messages. The use of a document-based communication model provides relations which are loosely coupled between services, and this results in flexible and adaptable connections. The layer uses the HTTP protocol to provide external communication services (communication between layers) and internal communication services (communication between layer modules). By means of the communication protocol, the Application Layer will invoke the Session Management module and this in turn, will invoke the Shared Access Control and Group Awareness service.

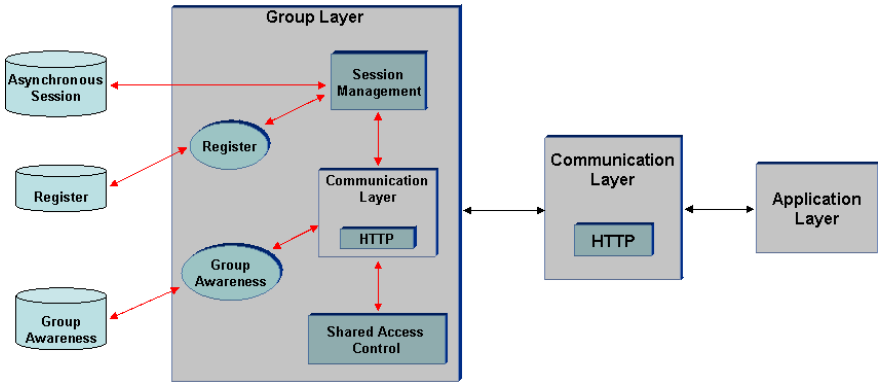

Fig. 1. SOA-based Architecture for the Development of CSCW Systems

3.2 Group Layer

It is important to provide the user with mechanisms that allow him/her to start and perform groupwork. Consequently, a shared workspace is firstly established and the created sessions are sufficiently flexible to be adapted to diverse forms of groupwork organization. Secondly, a set of policies to facilitate interaction between users and with the shared resources is defined; these policies avoid conflicts leading to information inconsistencies. And thirdly, mechanisms that give the user information about what the other users are doing and what is happening in the shared space are supplied. We shall describe the elements that carry out these tasks below.

- **Session Management Module:** This module manages the users (registration and group membership) and orchestrates the session, which includes the invocation of tools, the provision of information about the session state, as well as its initiation, suspension, resumption and stopping. Users initiate a session by invoking the Register service which stores session information (session name, user id) in the repository called Register. The Register service therefore provides the user with information about how to create a session, the currently open sessions, the number of users in each one and detailed information about each user (name, alias, occupation, photograph, etc.), as well as authentication mechanisms. Among other things, group membership enables users to create, join, withdraw from, invite someone to, and exclude someone from a session. The asynchronous session uses the Asynchronous Session repository to store information generated by the users, thereby providing the shared workspace necessary for interaction between them. The synchronous session provides a shared space that allows connected users to work together on shared sources in order to carry out a specific task in a certain time. We currently consider two session management policies. The first policy is a moderate session, whereby a moderator or president controls and coordinates the session. He/She selects the appropriate tools and establishes the turn in which each user can participate. The second is a brainstorming

session, which functions in a similar way to instant messaging applications. The group can change the session management policy in run time. This policy determines the roles that users can play and each role represents the set of access rights that users have on shared sources and the actions that they can perform. These access rights are defined in the Shared Access Control module, so the session manager must inform this module when a user joins or leaves a session so that the coordination policy can be adapted to the new situation. Since the roles are dynamic, one user may play several roles during a session. In conclusion, this module allows modifications to be made to the group size, its organization and the user roles so that the system can be adapted to group requirements.

- **Shared Access Control Module:** This session enables different users in various geographical places (or even in the same one) to interact and work with a variety of resources in a shared common context in order to reach a common goal. This module therefore coordinates the interactions to avoid conflicts in the shared workspace, due to the cooperative and competitive activities between users, by supplying dynamically generated, temporary permissions to collaborating users. In this way, the race conditions are lessened, the mutually exclusive resource usage is guaranteed and safety, timeliness, fairness, adaptively and stability are provided to session participants. The permissions granted to users depend on the roles that they can play and specify which user is allowed to send, receive, or manipulate shared data at a given moment. The default policy is “free for all”, where conflicts are resolved by serialization of access requests to shared resource on a first-come-first-served basis.
- **Group Awareness Service:** In order to be able to cooperate, users must be aware of the presence of other members in the session and of the actions that each one has carried out and is carrying out. One of the main tasks of any CSCW system is to provide the users with the necessary information to support group awareness, and this helps session participants establish a common context and coordinate activities, thereby avoiding surprises and reducing the probability of conflicts in the group. This service therefore stores each action carried out by the users in the session in the Group Awareness repository, and notifies the remaining participants of each performed action. Since each action is stored, it may be presented to any user joining a currently open session. The service also allows messages to be sent with information about what the other users are doing. One way of providing group awareness is to notify users when somebody joins or leaves the session or to show the user list of a session.

3.3 Application Layer

This layer communicates with the Group Layer by invoking the Session Management module and allows components or applications to be integrated whenever they fulfil the conditions specified in the service description. It has been given this name due to the fact that it is in this layer that the specific application must be developed, i.e. the

collaborative application which users are interested in and want to use, e.g. a shared whiteboard.

4 Conclusions and Future Work

In this paper, we have presented an SOA-based generic architectural proposal for the development of CSCW systems. Since it is based on SOA, the resulting application is modular, reusable, interoperable, scalable, flexible, adaptable, robust, ubiquitous and cost effective. It also provides group awareness and mechanisms to support dynamical groups such as those for changing the group size, the role played by the user and the interaction policies between users. The architecture provides a good potential for reusability for three reasons: firstly, it is possible to reuse the session management policy because there are similar interactions in many work contexts; secondly, the shared access policies can be reused to coordinate the interaction of different resources or sessions; and thirdly, separating the group awareness mechanism allows us to implement it in different types of sessions.

We have studied and analyzed the main existing collaborative applications, and we have ascertained that most have been developed for specific applications. This implies that in order to create a new application, we must start from scratch, something which entails a great deal of effort for developers. There are other applications that support the development of these applications, but none is perfect, e.g. some require extra effort in order to customize the application, others are not platform independent or do not provide consistency mechanisms.

Our architectural proposal not only allows third-party components to be added, but it also provides open, flexible and robust sessions with group awareness mechanisms for supporting collaborative work consistently. The proposed architecture can be extended with new web services when required, without the need to modify existing services; we need only consider the description details of the service to be linked. By way of future work, we intend to increase the number of session and shared data access policies in order to provide sessions which are more flexible and suitable for the diversity of existing ways to organize the groupwork. We also want to establish notification mechanisms for the exchange of documents by means of messages in order to reinforce group awareness, and also to define security mechanisms.

References

1. Burridge, R.: Java Shared Data Toolkit User Guide version 2.0. Sun Microsystems, JavaSoft Division (1999).
2. Calvary, G., Coutaz, J., and Nigay, L.: From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCW. Proc. of CHI'97 (1997) 242-249.
3. Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., and Seguin, C.: Java Object Sharing in Habanero. Communications of the ACM, Vol. 41, 6 (1998) 69-76.
4. Coutaz, J.: PAC-ing the Architecture of your User Interface. Proc. of the Fourth Eurographics Workshop on DSVIS'97. Sringer-Verlag (1997) 15-32.

5. Dewan, P.: Multiuser Architectures. Proc. of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction (1995) 15-32.
6. Ellis, C.A., Gibas, S.J., and Rein, G.L.: Groupware: some Issues and Experiences. Communications of the ACM Vol. 34, 1 (1991) 39-58.
7. Fuentes, L., Pinto, M., Amor, M., and Jimenez, D.: CoopTEL: A Component-Aspect Middleware Platform. ACM/IFIP/USENIX Int. Middleware Conference (2003).
8. García, P., and Gómez, A.: ANTS Framework for Cooperative Work Environments. IEEE Computer Society Press, Vol. 36, 3 Los Alamitos, CA, USA (2003) 56-62.
9. Garrido, J.L., Paderewski-Rodríguez, P., Rodríguez, M.L., Hornos, M.J., and Noguera, M.: A Software Architecture Intended to Design High Quality Groupware Applications. Proc. of the 2005 International Conference on Software Engineering Research and Practice (2005) 59-65.
10. Graham, T.C.N. and Urnes, T.: Integrating Support for Temporal Media in to an Architecture for Graphical User Interfaces. Proc. of the International Conference on Software Engineering (ICSE'97), ACM Press, Boston, USA (1997) 172-182.
11. Goldberg A.: Smalltalk-80: The Interactive Programming Environment. Addison Wesley, (1984).
12. Guerrero, L.A., and Fuller, D.: CLASS: A Computer Platform for the Development of Education's Collaborative Applications. Proc. of CRIWG'97 (1997) 51-60.
13. Harrison, M., Thimbleby, H. (eds.): Formal Methods in Human-Computer Interaction, Cambridge University Press (1990).
14. Laurillau, Y., and Nigay, L.: Clover Architecture for Groupware. Proc. of the ACM Conference on CSCW. New Orleans, Louisiana, USA (2002) 236-245.
15. Li, D., and Muntz, R.: COCA: Collaborative Objects Coordination Architecture. Proc. of CSCW'98. ACM Press (1998) 179-188.
16. Paderewski-1Rodríguez, P., Rodríguez-Fórtiz, M.J., and Parets-Llorca J.: An Architecture for Dynamic and Evolving Cooperative Software Agents. Computer Standards & Interfaces, Vol. 25, Elsevier Science (2003) 261-269.
17. Paderewski-Rodríguez, P., Torres-Carbonell, J.J., Rodríguez-Fórtiz, M.J., Medina-Medina, N., and Molina-Ortiz, F.: A Software System Evolutionary and Adaptive Framework: Application to Agent-Based Systems. Journal of Systems Architecture, Vol. 50, Elsevier Science BV (2004) 407-416.
18. Patterson, J.F.: Taxonomy of Architectures for Synchronous Groupware Applications. Proc. of the CSCW'94 Workshop on Software Architectures for Cooperative Systems, Vol. 15, 3 ACM SIGOIS Chapel Hill, North Carolina (1994) 27-29.
19. Prinz, W.: NESSIE: an Awareness Environment for Cooperative Settings. Proc. of the European Conference of CSCW (1999) 391-410.
20. Roseman, M., and Greenberg, S.: Building Realtime Groupware with GroupKit, A Groupware ToolKit. ACM Trans. Computer-Human-Interaction, Vol. 3, (1996) 66-106.
21. Schuckmann, C., Kirchner, L., Schümmer, J., and Haake, J.M.: Designing Object-Oriented Synchronous Groupware with COAST. Proc. of CSCW'96, (1996) 30-38.