Security and Adaptability to Groupware Applications using a Set of SOA-based Services

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

[1] Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, 14 sur y avenida San Claudio. Ciudad Universitaria, San Manuel, 72570 Puebla, Mexico.
{anzures, luzsg}@correo.ugr.es
[2] Departamento de Lenguajes y Sistemas Informáticos, E.T.S.I. Informática y de Telecomunicación, Universidad de Granada, C/ Periodista Saucedo Aranda, s/n, 18071 Granada, Spain.
{mhornos, patricia}@ugr.es

Abstract. Two fundamental aspects of groupware are security and adaptation. The former protects the information and the resources being shared, as well as allowing only those authorized users to make use of them. The latter allows us to adapt the application to the inherent dynamic of group work. This paper proposes a security model and an adaptability process, both of which use services based on SOA. As each service can be independently modified, both are adjusted according to the requirements of each group. Security model makes up of set services and focuses on controlling the user's access to shared resources (avoiding inconsistencies in groupware application) and the groupware application itself (preventing unauthorized users joining into or making use of it). The adaptability process focuses on the group organizational structure, using a set of services to adjust the Groupware application. In order to facilitate the inherent dynamic of the group organization structure is used an ontology, which to model the organizational structure, determining its behaviour through the concepts its relationship and the axioms defined.

1 Introduction

Groupware concentrates on support group work to achieve a common goal; therefore, it is necessary to facilitate group flexibility to respond to different collaborative scenarios that upsurge from the inherent dynamism of group work, as well as group interaction to allow their members to share resources and information, which is a main aspect to facilitate group work. Group flexibility depends on the adaptation of the group organization structure to the inherent dynamism of the group work; i. e. when users join or quit a session, when users change his/her role, when organization style is changed, or when a user plays several roles, etc. In sum, this dynamism is related to group size and organization, so that the application can be flexible enough, to adapt itself to the group requirements. When information and resources are shared,

Advances in Computer Science and Engineering.
Research Computing Science 45, 2010, pp. 279-290
Received 30/04/10
Final version 19/05/10
Accepted 14/05/10

the security is an important aspect to be considered. This happens in Groupware applications. Whereas in most systems, security is achieved through mechanisms such as: authentication, access control, data encryption, digital signature, etc. In the groupware field, special attention has been paid to authentication and access control mechanisms. The former is a mechanism that allows us to identify and verify the user's identity, trying to protect the system from unauthorized access. The latter is a mechanism that permits us to protect information according to security policies, by permitting access to shared resources only to authorized users.

This paper focuses on set services to provide a security model (which supports authentication and access control mechanisms to control interaction among users) and an adaptability process (that facilitates the adjustment to the group dynamic nature and to the changing needs of the same). Both avoid inconsistencies in the application on account of cooperative and competitive activities. In addition, SOA is the potential solution to the problems arisen from adaptation and reuse; therefore, it facilitates reuse and adaptation of each service and module here proposed. The paper is organized as follows: Section 2 describes the adaptability process. Section 3 explains the security model. Finally, Section 4 outlines the conclusions and future work.

2 Services-based Adaptability

Software adaptation [7] is based on the necessity of adjusting the system functionality in accordance with the new requirements that will appear in the future (changes in the environment, users' needs, different devices, etc.), in such a way that the system can continue working correctly. There are two forms of adaptability [6], [9]:

1. *Adaptive*, the adaptability is automatically performed and it is based on certain mechanisms previously defined by the designer and/or the developer.
2. *Adaptable*, the adaptability is carried out by the user's direct intervention in related to a set of constrains that avoid inconsistencies in the application.

The Groupware adaptability is focused, mainly, on the following aspects: access control [11]; concurrency control [4]; coupling of views [3]; and extensible architectures [8]. In the work presented in [1] these aspects have been addressed. In this paper, the adaptability focuses on the group organizational structure.

The adaptability controls in what way the components of the groupware application will be adapted when a change (or event) requiring modification takes place, so that the application functionality can be preserved. The adaptability of the group organizational structure comprises two phases: Pre-Adaptation and Adaptation.

2.1. Pre-Adaptation Module

This module (see Figure 1) provides a set of services which allow the Groupware application to determine whether it will carry out the application adaptability. Therefore, the Pre-Adaptation Module executes the following steps:

1. It monitors, at all times, (using *Detection Service*) events triggered in the execution environment (in this case, the *Collaborative Application Service*).

2. Each event detected by the *Detection Service* is compared with those contained in the *Adaptation Event Repository*.
    2.1. If a detected event is found into the *Adaptation Event Repository*, an adaptability process is required (go to step 3).
    2.2. Otherwise, an adaptability process is not required (go to step 1).
3. It determines the kind of adaptation to be carried out.
    3.1. Adaptable, go to step 4.
    3.2. Adaptive, take the following steps:
        3.2.1. A consensus must exist among all the group members (by means of the *Agreement Service*), to decide on (through the *Voting Tool Service*) whether an adaptation process should be performed or not.
        3.2.2. If the group decides to adapt the application, go to step 4.
        3.2.3. Otherwise, go to step 1.
4. Groupware application adaptability (go to Section 2.2).

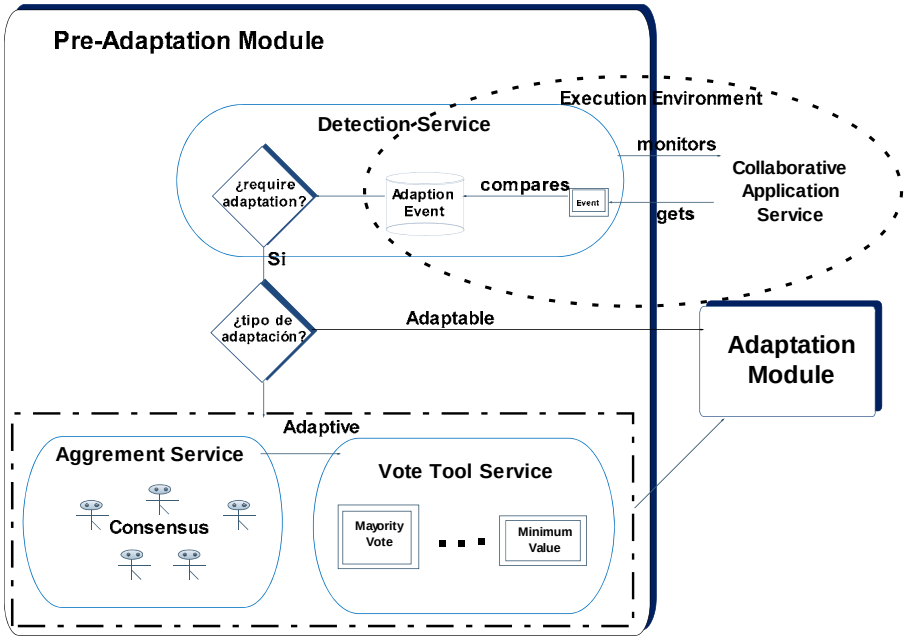

Fig. 1. Pre-Adaptation Module based on services.

2.1.1. Detection Service

This service (see Figure 1) monitors each event carried out at Groupware application through the *Notification Service*, which triggers itself whenever an event takes place. The *Detection Service* compares this event with those events registered at the *Adaptation Event Repository*, which contains only those events that involve an adaptability process. In addition, each event, in this repository, is associated to the kind of adaptability to be carried out, i.e., adaptive or adaptable process. The former

requires knowing the established organization structure in the group, namely that if it is a non-hierarchical organizational structure, the *Agreement Service* which allows users to reach a consensus using the *Vote tool Service*, will be executed. But, if it is a hierarchical organizational structure, an adaptable process is required.

2.1.2. Agreement Service

This service (see figure 1), is used by group members in an adaptive process whose organizational style is non-hierarchical, to decide whether the Groupware application adaptability should be performed or not. Therefore, the *Agreement Service* supports the consensus that all members have to reach.

2.1.3. Vote Tool Service

This service (see figure 1), provides a set of voting tools to be used by group members in order to reach an agreement. The *Vote Tool Service*, allows group members to choose from different kinds of agreements, which are based on a majority vote or on maximum/minimum value, etc. The group then will make a choice according to the established requirements to accomplish the given group task.

2.2. Adaptation Module

This module (see figure 2) performs the Groupware application adaptability; therefore, it executes a set of operations (which are stored in the *Operation Repository*) to determines that actions must be carried out in each service has be adapted. In order to avoid possible inconsistencies in the Groupware application, this phase is carried out by the *Adaptation Flow* Service.

2.2.1. Adaptation Flow Service

This service (see Figure 2) controls and manages the actions, which should be carried out in each service that will be adapted. These actions are determined by set of associated operations to an event. If some action can not be performed a reparation process is executed using the *Reparation Service*. The *Adaptation Flow Service* performs the following steps:

a) It requires and gets to the *Adapted Components Service* the services list related to the adaptability process, as well as, the associated operations list to each service that show in the services list. The adaptability process stars with the first service and stop with the last service in the list, storing its name in the *Adaptation flow Repository* together with the operations that will allow us its adaptation. This adaptability process begins in the step “b)”.
b) It determines if the service meets the necessary pre-conditions (which establish the conditions that Groupware application must fulfil to be adapted)

to be modified. If these pre-conditions are not satisfied the *Adaptation Flow Service* is stopped, a message that indicates the reason by which stopped is sent, and sometimes it is necessary perform a reparation process.

c) It executes the adaptation operations associated with service to adapt it.

d) It decides if the service adapted meets the necessary post-conditions (which are the conditions that must be achieved later that the component has been modified). In case that the post-conditions are not satisfied, a reparation process is carry out, following the adaptation flow established by the adaptability process but in opposite direction. Thus, each component returns to their previous state and notifies users that adaptation can not take place.

e) The steps "b" to "d" are repeated until that all the services in the list got in the step a) have been adapted, thereby the adaptation process finish.

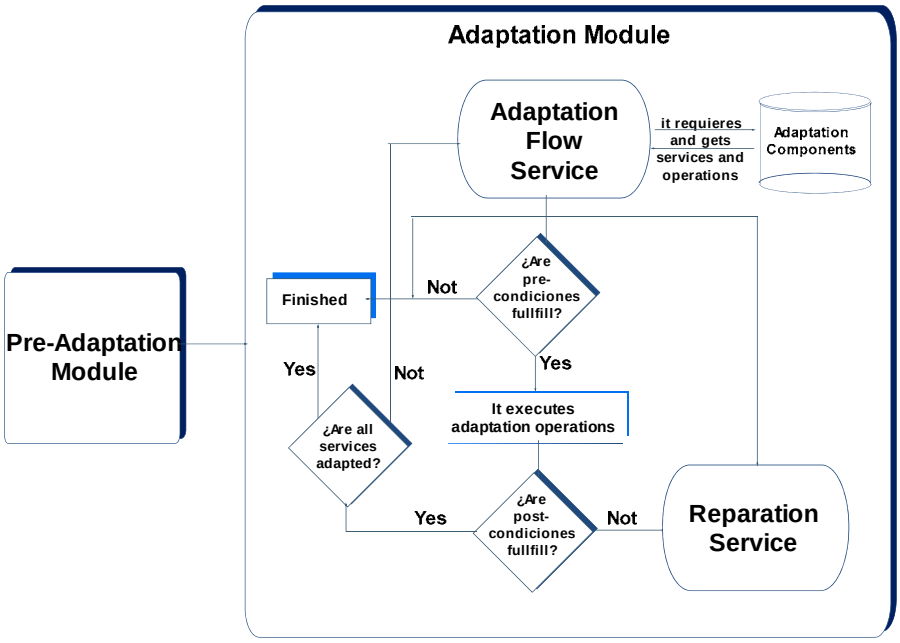

Fig. 2. Adaptation Module based on services.

2.1.2. Reparation Service

This service (see Figure 2) provides a mechanism in order to the application functionality can be preserved when the adaptability process can not be finished because not fulfill the post-conditions. The *Reparation Service* queries the *Adaptation Flow Repository* to get the last operation carried out in the last service adapted. In this way, each adapted service (starting with the last service and finishing with the first service of that repository) is returned to their previous state and notifies users that adaptability can not take place; preserving the appropriate application operability.

3 Services-based Security Model

In order to propose a service-based security model, on the one hand, the access control management and the authentication have been analyzed and modeled from the beginning of the Groupware application development. On the other hand, the main existent access control models have been studied; as a result we have detected their main problems, which are: they are not able to dynamically change permissions, they do not provide the global context of the collaborative application, and they do not specify neither restriction nor fine granularity control. The service-based security model is used to control how the shared resources are managed in the Groupware application in order to avoid inconsistencies. This model is based on RBAC model [10], the permissions are assigned to roles instead of users; in this way, when users modify their roles in the organization, the access control policy does not change.

The service-based model improves the existent access control models, for it considers: the static and dynamic aspects of the security, sophisticated and appropriate security policies, the general context related to collaborative activity, and the dynamic nature of the group, in such a way that permissions, roles and constrains are part of the model and these can be changed in runtime. In order to achieve these benefits, this model is based on four modules (see Figure 3): *Authorization*, *Session, Interaction Control,* and *Context.* Each of them abstracts a concern related to the establishing of sophisticated and appropriate access policies for group work, as well as, dynamic adaptation of its functionality in runtime. The changes can be:

- *Evolutionary:* Changes in the model which are dynamically updated in the collaborative application.
- *Adaptive:* Changes in the group organizational structure, which are predetermined in the access policy defined in the collaborative application.

3.1. Authorization Module

In the existent access control models, the authorization of users and the modification of permissions in runtime are core aspects, because access to resources in a session is controlled by the permissions or authorization that a user has at a given moment. This module (see figure 3) establishes the authorized roles to access a session, and defines how the users interact with shared resources and among themselves. It also allows the organizational structure to be dynamic. The *Authorization Module* facilitates an appropriate, dynamic and simple authorization of users in the session by four services: *Registration, Authentication, Stage,* and *Group Organizational Structure* (see Figure 3).

3.1.1. Registration Service

This service allows to the users to participate in a session, because the first thing that a user must do is to register her/himself on this. A user can join a group under own

petition or via an invitation. In the first case, the user must send a request via email indicating that she/he is keen to take part in the session. If the user meets the use conditions, the login and the password are sent to her/his email address. In the second case, the user receives in her/his email client an invitation with a login and a password to access the session. She/he will use the information received when she/he wants to join this session. When a user is registering in this, it is necessary that she/he fills in a registration form with her/his personal data and with useful information for carrying out the group work. The role that the user will play in this session is also stored and is provided in accordance with the valid roles in the *Group Organizational Structure Service* according to current stage (which is established in the *Stage Service).*

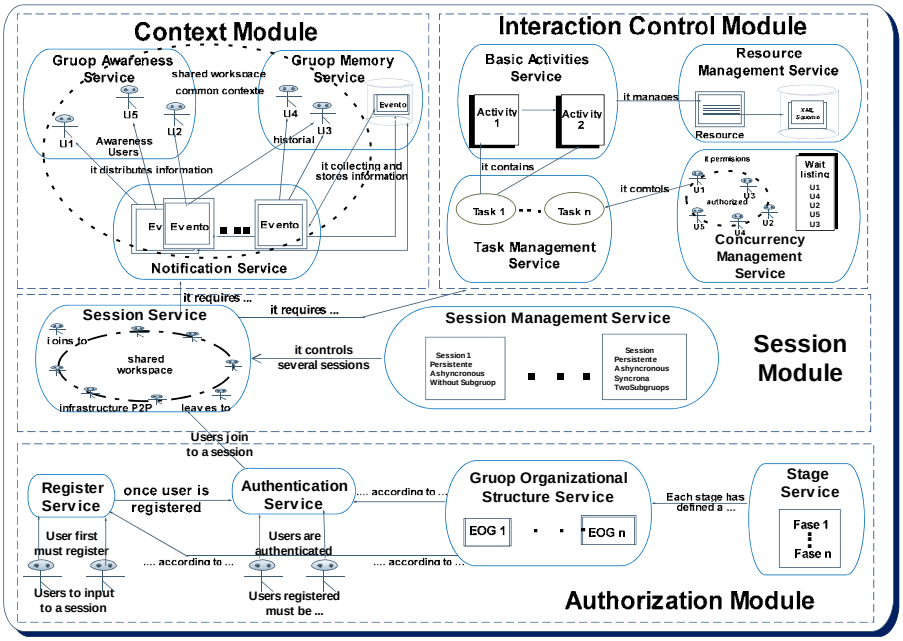

Fig. 3. Service-based Access Control Model

3.1.2. Authentication Service

This service controls the user access to the Groupware application, is used once the user is registered in the Groupware application. When the user inputs a login and a password, this service authenticates her/his access to the session, and stores these data, which are used by the system to corroborate that she/he is an authorized user. The authentication is dependent on the defined organizational style (*Group Organizational Structure Service)* and the currently executed stage (*Stage Service*).

3.1.3. Group Organizational Structure Service

This service presents an ontological model [2] that specifies the group organizational style taking into account all its static and dynamic aspects. Moreover, it allows controlling the access to the shared workspace and resources, and facilitates user authentication in this space. The group can change its organizational style in runtime. A style determines the roles that users can play and each role represents the set of access rights that users have on shared resources and the actions that they can perform. We have described the elements and the functional dependencies between them by means of this ontology, which allows specifying and controlling the changes on: the roles that a user can perform in a session, the access rights of each role, and the tasks to carry out in this. Moreover, in accordance with this model, we can create templates of policies that can be used to facilitate their reuse in runtime. For these reason, the security model facilitates the group work adaptability.

3.1.4. Stage Service

We consider long-term Groupware applications, where sharing information takes place at various stages. A stage [5] in a coordination model is defined as each of the collaboration moments. This service defines at least two stages: the collaboration and the configuration. The former, where the work group is carried out, can be in turn divided into several stages; for example, we distinguish four stages for it in the CMS: submission, assignment, review, and acceptance of papers. Each stage controls the roles that can participate in it, which facilitates the user authentication to the shared workspace. This service manages the collaboration moments, by establishing when a stage begins and ends, when a stage ends, it is possible to change the current organizational style (see section 3.1.3) to another more appropriate.

3.2. Session Module

This module supplies a shared workspace to carry out the interaction process in the Groupware applications, by means of two services (see Figure 3): *Session Management Service*, and *Session Service.*

3.2.1. Session Service

This service orchestrates the session, which includes its establishment, initiation, suspension, resumption and stopping, as well as the provision of information about the session state. A session denotes a set of geographically distributed individuals, who share a common interest to perform common tasks, so only a group can work in a session. This service provides a mechanism, which is supported by means of a peer-to-peer infrastructure, which allows connected users to work in a collaborative environment. In this way, it allows the users to join, to leave, to invite someone to,

and to exclude someone from a session. Once a session is established, this service manages and controls the session and the connections of users to it, storing information about each user and her/his work session; with this, it is possible to identify the users that are connected to each session. A user can participate in more than one session.

3.2.2. Session Management Service

This service supports the execution of several sessions at the same time. Here, each session is persistent, can be asynchronous or synchronous, and can be executed into another session. We consider session as persistent, which means they keep the session state until the next connection. This helps overcoming the well-known latecomer problem, and managing the adaptation process. Thanks to that, the session state can be retrieved from the server and it is possible to present the current state of the session to a new user. A Groupware application must be able to support both asynchronous and synchronous communication, so that they can be used when necessary. This service allows a group working in a session to make up a subgroup, which can start a new session. Both (group and subgroup) sessions can have different communication types, i.e. asynchronous or synchronous. This is important, because it allows the application to be adapted to the requirements of the group work.

3.3. Interaction Control Module

Once the group organization has been defined (i.e. user authorization to carry out the tasks and to use the shared resources) by means of the above mentioned module (see section 3.1.1), it is important to control the communication and coordination logic of the collaborative application to provide a good performance of the same at all times. This way, the mutually exclusive resource usage during the interactions among users must be guaranteed. This is achieved by means of this module, which contains four services, called: *Concurrency Management, Task Management, Basic Activities, and Shared Resource Management* (see Figure 3).

3.3.1. Concurrency Management Service

This service facilitates the manipulation of user permissions relevant to ensure exclusive access, which are granted in accordance with a default policy and a lock mechanism. The default policy for accessing the shared resources is “free for all”, where the conflicts are resolved by a serialization of the access requests to shared resources on a first-come-first-served basis. This policy can be modified by the organizational style (established at the current stage). The access request to a resource is made to the *Shared Resources Repository*, which verifies whether the user has the required access rights to use that resource. In affirmative case, and if the resource is free, the user is allowed to use it to carry out the corresponding task. If the resource is

busy, the user requesting this resource is informed about the resource state and her/his request is put on the waiting list. If a user does not want to wait for the resource, she/he can remove her/his request from the waiting list at any moment. Once the resource is free (because the user who was using it finished their task or left the session), it is assigned to the next user on the list. This process is repeated until either all the users on the waiting list have used the resource or the session has finished. If the user does not have the rights to access to the resource, she/he receives a notification indicating that she/he can not use this resource.

3.3.2. Shared Resource Management Service

This service defines a flexible mechanism for storing the resources used which does not establish any restriction about the data logical model used in its description. The resources storing is performed through *XML* documents, so that the elements of the *Shared Resources Repository* are serialized in *XML* format. The *XML Schema* representation of each type of shared resource that the system uses needs to be stored in the meta-level repository. In this way, each invocation to the primitive contexts associated takes two parameters: one identifies the resource requested in the operation, and the other is a reference to its structural schema, so that each operation can be uniformly carried out regardless of the shared resource type on which it operates. Moreover, this service supplies a set of operations for managing resources in the repository. These operations allow us to insert, update, delete and query resources in the repository.

3.3.3. Basic Activity Service

This service defines a component that provides a set of activities to manage the resources stored in the *Shared Resources repository*. These activities are independent of the specific semantic associated with the different types of shared resources used, although their implementation should be carried out according to the data logical model used in the repository. This service uses the services of *Concurrency* and *Group Organizational Structure*.

3.3.4. Task Management Service

This service manages the tasks to be performed in the Groupware application. Each task is made up of a set of basic activities, which must be carried out to complete the corresponding task. The information about the tasks to be carried out is gotten from the *Group Organizational Structure Service*, which facilitates the tasks management. Due to the tasks are carried out in the shared workspace, this service uses the *Concurrency Management Service* that guarantees a coordinated management of the shared tasks used in the cooperative processes.

3.4. Context Module

The context is a very important element in the collaborative work, since the *Security Model* must take into account the current context of each user to establish the access permissions. Context information is necessary to manage the users and the resources access control, as well as users' authentication. This module (see Figure 3) contains three services, called: *Group Awareness*, *Group Memory*, and *Notification*, *w*hich provide a set of elements in relation to the collaborative context. Thanks to them, it is possible to control the state of each user, using the information stored in a session.

3.4.1. Group Awareness Service

This service provides to the users with the necessary information to support the group awareness. In this way, users are aware of other members presence in the session, as well as the actions that each of them has carried out and is carrying out, which facilitates user cooperation in collaborative tasks.

3.4.2. Notification Service

It provides users with operations for registering the interest in receiving collaborative events and for removing that interest, as well as for notifying the events produced during a session. For each user joining a session, an instance of the *Notification Service* is created. This instance is registered in the *Group Awareness* and/or *Group Memory Service* as an event consumer. Each cooperative event triggers the invocation of the corresponding notification operation, which delivers it to the *Group Awareness* and/or *Group Memory Service*, which in turn will propagate it to all registered consumers. These notifications are asynchronous, to avoid waiting and connections problems in the application. All events are distributed as strings, making the system suitable for any kind of event produced by any kind of Groupware application.

3.4.2. Group Memory Service

This service supplies a common context in the Groupware application which is called the group memory. This is a common space on which the collaborative activities of the group are carried out, and where the information about the shared resources is stored. This memory is created to provide understanding and reasoning about the collaborative process, and to do an exact tracking of this process. Therefore, this service collecting, storing and distributing information about the shared resources used and the activities undertaken with these resources, with the aim of supporting the dynamics of the group knowledge representation. In addition, the *Group Memory Service* provides persistence to the Groupware application, which gives the necessary information to perform several processes, such as the latecomer, adaptation or reparation ones.

4 Conclusions and Future Work

In this paper, it has been presented a set of SOA-based services for supporting the adaptability and the security in the Groupware applications. These are two quality attributes very important. The former is controlled and managed by five services, which allow us to adjust the organizational structure to the group necessities, preserving in all moment the appropriate application functionality. The latter is supported by *Security Model* using thirty services in order to protect the Groupware application, as well as the shared information and resources of unauthorized users. In this way, the services developed can be used in and/or modified to other applications to provide security and adaptability, thanks to that SOA provides reuse and adaptability. The future work will propose an ontology to facilitate the integration of these services in architectural model for develop Groupware applications.

References

1. Anzures-García, M., Hornos, M.J., Paderewski, P. Development of Extensible and Flexible Collaborative Applications using a Web Service-based Architecture, Springer-Verlag, LNCS Vol. 4401, 2007, pp. 66-80.
2. Anzures-García, M., Sánchez-Gálvez, L.A., Hornos, M.J., Paderewski-Rodríguez, P. Ontology-Based Modelling of Session Management Policies for Groupware Applications, Springer-Verlag, LNCS, Vol. 4739, 2007, pp. 57-64
3. Dewan, P., Choudhary, R. Coupling the User Interfaces of a Multiuser Program, ACM Transactions on Computer Human Interaction, Vol. 2-1, 1995, pp. 1-39.
4. Ellis, C.A., Gibbs, S.J. Concurrency Control in Groupware Systems, In Proceedings ACM SIGMOD, ACM Press, 1989.
5. Ellis, C., Wainer, J. A Conceptual Model of Groupware, In Proceedings of the ACM Conference on CSCW, 1994, pp. 79-88.
6. Fink, J., Kobsa, A., Nill, A. Adaptable and Adaptive Information for all Users, Including Disabled and Elderly People. The New Review of Hypermedia and Multimedia. Vol. 1-4, 1998, pp. 163-188.
7. Hiltunen, M.A., Schlichting, R.D. Adaptive Distributed and Fault-Tolerant Systems, Journal of Computer Systems and Engineering. Springer Verlag, Vol. 11-5, 1995.
8. Lee, J.H., Prakash, A., Jaeger, T., Wu, G. Supporting MultiUser, MultiApplet Workspaces in CBE, In Proceedings of 1996 the ACM Conference on CSCW, 1996, pp. 344-353.
9. Medina, N., García, L., Torres, J.J., Parets, J. Evolution in Adaptive Hypermedia Systems, In Proceedings of Conference on Principles of Software Evolution, 2002, pp. 34-38.
10. Sandhu, R. S., Coyne, E. J., Feinstein, H. L., Youman, C. E. Role-based Access Control Models, IEEE Computer, Vol. 29-2, 1996, pp. 38-47.
11. Shen, A.H., Dewan, A.P. Access Control in Collaborative Environments, In Proceedings of the ACM Conference on CSCW, 1992, pp. 51-58.