Reusing Groupware Applications

Sergio F. Ochoa[1], Luis A. Guerrero[1], José A. Pino[1], and César A. Collazos[2]

[1] Department of Computer Science
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
{nbaloian,pgaldame,luguerre}@dcc.uchile.cl
[2] Department of Systems
Universidad del Cauca
FIET-Sector Tulcan, Popayán, Colombia
{ccollazo}@unicauca.edu.co

Abstract. Many groupware applications have been developed and continue being developed over white-box groupware platforms. These platforms have brought important contributions to the development of groupware systems. However, the lack of compatibility among these platforms is limiting the portability of such solutions. This paper presents a middleware, which allows to improve the portability of new and legacy groupware applications supported by white-box platforms. The middleware translates a set of functionalities provided by the groupware platforms to a set of common groupware services used by the applications. These services provide groupware support and allow to improve the portability of groupware systems. A prototype of the proposed middleware has been tested and the interim results are encouraging.

1 Introduction

Concepts like portability and reuse of components and designs have already been accepted and promoted by the software industry [1, 6]. The main advantage is the reduction in the development time and cost. In addition, an improvement in quality of the final product can be expected because of the reuse of mature designs and software components. These results have motivated the software industry to move towards the reuse of adaptable solutions that use standard interfaces, since they are easy to scale, update, and replace. Although reuse of solutions brings important advantages for the software industry, it also requires a high level of product standardization. Portability is poor in the case of groupware systems, specifically, applications supported by white-box groupware platforms. White-box groupware platforms, such as Coast [12], Groupkit [7] and TWiki [5], provide through the server, a set of classes or services implementing typical concepts of the groupware area, such as floor control, shared repository and session management. These services are not compatible among different platforms, because of the lack of standards in the groupware area. Therefore, applications using these services are not easy to reuse.

Many organizations have spent important efforts on developing and inserting groupware applications to support vital parts of their functionality (e.g. workflows, discussion forums or messaging systems). Lack of portability of their groupware applications forces organizations to avoid or delay the change of such supporting

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 262–270, 2004.

platforms. This would mean the re-development of the current applications, the re-training of the employees and the migration of the information.

This paper presents a middleware component, called GPC (Groupware Platform Compatibility), which acts as an intermediary between groupware applications and white-box groupware server platforms. GPC homogenizes the main groupware services required by applications and hides the differences among white-box groupware platforms.

Next section describes the groupware concepts that could be used by groupware applications in order to provide collaborative functionalities. Section 3 presents related work about portability of groupware solutions. Section 4 presents the main components of the GPC middleware. This section also shows how to work GPC and the restrictions required for using it. Sections 5 and 6 present the interim results, the conclusions and future work.

2 Groupware Services

Guerrero et al. have defined nine basic concepts (or patterns) which are eventually present in a groupware system to be supported by server platforms [8]. These concepts are: *sessions*, *users*, *roles*, *messages*, *objects*, *repositories*, *views, environments* and *floor control.* The *sessions* (also called work sessions, groups, or conferences) maintain information about the users who interact in an asynchronous or synchronous way through a groupware application. The *users* (or collaborators) are the members of the work group. Every user can have specific access rights according to his/her *role*. Users send *Messages* (notifications or events) to communicate. Application modules also send messages for the same purpose. *Objects* (meta-objects or information objects) are information about object instances produced by the users during group work. Most of the time, it is needed to store object attributes, which is information about the work object (or meta-information), such as the owner of the object, the creation date and time, and previous versions. The objects are stored in *repositories* and it is possible to see a portion of a repository by using various views. The *environments* organize and coordinate multiple working sessions, or multiple user groups working on the same groupware applications. They also allow sharing groupware applications among many user groups. The *floor control* policies define the strategies used to manage the shared resources.

Table 1. Common features in white-box groupware platforms.

Features	Coast	GroupKit	Habanero	JSDT	MetaWeb	TOP	TWiki
Sessions	X	X	X	X	X	X	X
Users	X	X	X	X	X	X	X
Roles		X				X	
Messages	X	X	X	X	X	X	X
Objects	X		X	X	X	X	X
Repositories	X			X	X	X	X
Views	X				X	X	
Floor Control	X	X	X	X	X	X	X
Environments	X		X	X		X	X

White-box groupware platforms, such as: *Coast* [12], *GroupKit* [7], *Habanero* [4], *JSDT* [3], *MetaWeb* [14], *TOP* [8], and *TWiki* [5], provide services to support most of these concepts (see Table 1). However, the services are not compatible among the different platforms. This generates the main problems of portability for groupware applications.

3 Related Work

Although there are no specific solutions to address the problem of portability in applications supported by white-box groupware platforms, some options designed with other goals could be used to address it. Software components [13, 6], such as: EJB, DCOM and .Net, could be used as intermediary to increase the portability of such groupware applications. These technologies have demonstrated being useful to reduce the dependencies among client and server applications. However, they involve the use of heavy-weight server platforms, and some of them are tied to a specific family of operating systems. In contrast, groupware applications and white-box platforms tend to be light-weight and independent of the operating system; therefore, the use of these component technologies could be a high cost to pay in order to improve the portability of such groupware solutions.

An alternative would be to use Web services [2], which propose an architecture based on XML messages allowing applications to talk to each other. Although the Web services paradigm is the ideal for developing portable applications, it uses connection-less interactions between client and server applications. Awareness, replication of operations, and users and sessions management are some of the groupware services that require connection-oriented interactions. In other scenarios, such as databases, operating systems and communication protocols, interesting solutions have been created to solve similar problems. The most related works are the ODBC and JDBC frameworks [10]. They reduce the dependency that an application has respect to the database it is using. This solution has demonstrated to be useful to reduce the incompatibilities among the services that databases provide to client applications. In addition, it is light-weight and independent of the operating system of both, the client and the server application. By reusing this idea, GPC middleware was created to help overcome the limitations of portability of such groupware solutions. Because most of white-box server platforms use event-based client-server architectures [9], GPC acts as an intermediary homogenizing the interaction between groupware applications and white-box server platforms. This homogenization gives portability to the applications.

4 Groupware Platform Compatibility (GPC)

The GPC middleware was implemented using the J2EE platform. The access to the functionality of the middleware was implemented through a DLL (Dynamic Link Library) that runs on the client side, similarly to an ODBC driver. Even though the rest of components of the GPC also run on the client side, they are not available to be used by external applications.

The main components of the GPC middleware are three processes and three configuration files (see Fig. 1). The processes are *Service Deliverer*, *Service Manager*

and *Auxiliary Service Provider*; and the configuration files are *Communication Status File* and a *Groupware Platform Specification File* and *Current Client-Server Settings*. The integrated work of these elements allows GPC to provide groupware services to the server and client applications.

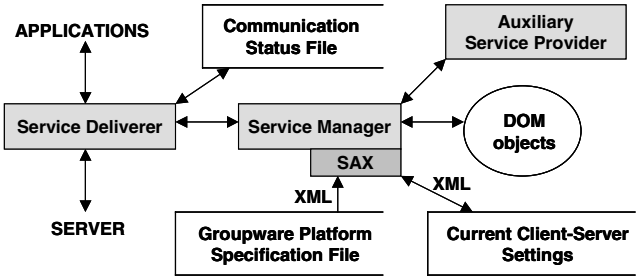

Fig. 1. Architecture of GPC.

Service Deliverer is a software agent which acts as intermediary among clients and servers, managing the communications and maintaining their status. Typically it uses the *Communication Status File* to store and retrieve information about the current communication which is being managed by GPC. This information concerns IP numbers and identifiers of client and server applications, communication protocols and ports, status of request between a server and a client, and related data. On the other hand, *Service Manager* is also a software agent, which translates the requests and responses from the original format to one understandable by the receiver. Also, it is responsible to emulate the message interchange protocol that is available in each supported white-box groupware platform. This emulation capability is needed when the granularity of the service requested by a client is different to the offered by the server. In this case, the Service Manager also acts as temporal buffer emulating the server or client behavior, and keeping track of the interactions among them.

The Service Manager carries out these tasks supported by the Communication Status File (CSF), Groupware Platform Specification File (GPSF), Auxiliary Service Provider (ASP) and Current Client-Server Settings (CCSS). CSF provides updated information to generate the communication instances (commands) between the Service Deliverer and the receivers, in order to carry out the interactions required by collaboration processes. GPSF provides general information to understand such communication instances (request or results) and assigns them a right meaning for each white-box platform. GPSF is an XML document, which also stores data about how to translate native requests or responses from one supported platform to one or more groupware services provided by GPC middleware. In addition, this configuration file provides the inverse function; this means to translate each GPC groupware service to one or more native services provided by a specific white-box server platform.

Because most of white-box server platforms do not provide the nine basic concepts which could be required by groupware applications, an *Auxiliary Service Provider* (ASP) is included in the middleware. This helps Service Manager to carry out the proposed service translation. ASP is a small server process based on the TOP white-box platform [8]. This process provides some TOP services not offered by the server platform but possibly requested from a groupware application.

Finally, the CCSS is an XML file - similar to the GPSF - storing information about the relationship between each client application and its corresponding server platform. This file is modified every time a client application changes from a server platform to another one. Using this file, the GPC middleware can manage multiple interactions among clients and servers. However, the GPC middleware may become itself a bottleneck, reducing the performance of groupware applications.

4.1 Functionality of GPC

Typically, when a groupware application sends a request (in a native format) to a server through GPC, the middleware Service Deliverer receives the message. Then, it sends it to the Service Manager, which checks if client and server share the same groupware platform. If they do, the Service Manager returns the original request to the Service Deliverer, which sends it to the server. Otherwise, the Service Manager translates the original request into a set of equivalent GPC service requests. The Service Manager uses the CCSS file to identify the specific server platform that provides support to each client application. Moreover, Service Manager uses GPSF to identify which requests will be sent to the server through the Service Deliverer and which ones will be sent to the Auxiliary Service Provider (ASP). Then, the response results provided by the server will be received by the middleware Service Deliverer and processed by the Service Manager. The results provided by the ASP are directly received and processed by the Service Manager. This software agent also translates the results into a native format for the client (understandable by the client) when it is needed. Thus, the processed results are delivered to the clients in the same way the native server would do it. The Service Manager uses SAX [11] to parse the GPSF and CCSS files and then it generates a DOM tree. This tree stays in memory in order to reduce the overhead produced by the translation process.

4.2 Implementing the Groupware Platform Specification File

One important challenge when implementing GPC middleware was to define an appropriate structure for the Groupware Platform Specification File (GPSF). This structure must be general enough in order to exclude limitations to the translation processes. A map of the resulting GPSF structure is presented in Fig. 2. It is currently implemented as an XML file.

GPSF has two entries, one through the native server platform and the other through the standard service (common groupware service). By using some of these identifiers, it is possible to retrieve the information about how to translate a service from the native server platform to GS (Groupware Services) and vice versa. For example, when the Service Manager needs to understand a service request from a client application, it should specify the platform of the client and the requested service. Being a request of service, the *request* section of GPSF provides information to validate the format of the received command and to assign it the right meaning. Thereafter, such request is translated to one or more GS provided by GPC middleware. Later, these GS are translated to a set of one or more services provided by the current server platform and/or the Auxiliary Service Provider. Such translation is done using the second entry of GPSF, which uses a GS as primary key. Finally, the translated services are delivered to the corresponding server platform through the Service Deliverer.

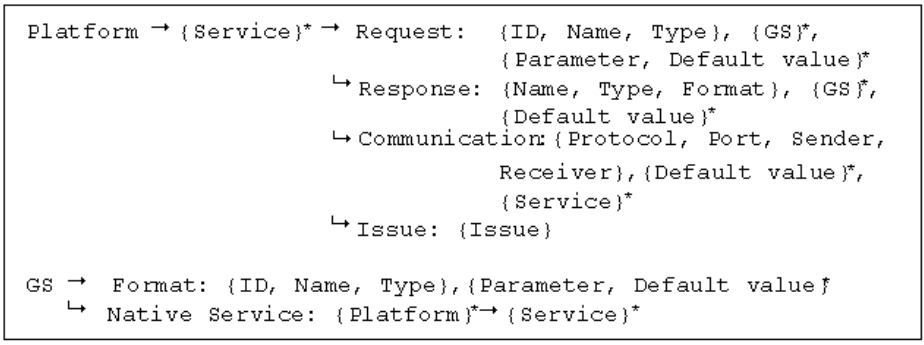

Fig. 2. Structural Map of the Groupware Platform Specification File.

Similarly, Service Manager uses the information specified in the *response* section of GPSF to translate the results from the server format to the format understood by the client. It should also emulate the interaction protocol between the client and server, by using the information specified in the *communication* section of GPSF, Communication Status File and Current Client-Server Settings.

4.3 Using the GPC Middleware

The first time GPC is included in a groupware solution, the connection to the server should be changed in the source code of the client application. This change seeks to address GPC as its new server, to indicate the white-box server platform the client application believes to be addressing, and to assign a unique identifier for such application. This allows GPC to identify each client, to assign the right semantics to receive requests and to translate the data to be delivered as result of such requests. In other words, the client application will continue believing the original white-box server platform is providing such services, and GPC is responsible to induce such beliefs. The application must change its connection function call, including its own identifier as an additional parameter in each server request. These modifications to the source code are required the first time such application is configured to work with the server using GPC as intermediary. Then, every time the server platform needs to be changed, only a CCSS file is modified.

On the other hand, server platforms should recognize GPC as a client application which is using the groupware services provided by them. It involves updating of the same configuration file included in GPC, which allow client applications change the server platform without the need to update their source code.

Initially, GPC works as a listener of the interactions between client server, and it assumes more important roles when the server platform is changed. During these interactions the middleware is in charge of encapsulating a set of typical groupware services by providing interfaces allowing client applications to talk with a server platform different than the one it was designed. Client applications can reach portability by using only these groupware services. Otherwise, if a client application uses extra groupware functions provided by a specific server platform, it must modify its source code to fit this restriction.

The groupware services provide support for collaborative work to groupware applications. These services are never accessed directly by client or server applications, but by GPC components. Usually, each request from a client application or server platform is translated to one o more groupware services. These services seek to establish a common intermediate language to translate the functionality provided by each white-box server platform.

5 Results

The current implementation of the GPC middleware supports TOP and TOP II white-box platforms. These platforms are partially compatible because many functions available in TOP were redefined in TOP II.

Three groupware applications supported by TOP were used to test the GPC middleware. The applications were: an asynchronous discussion forum, a shared drawing environment, and a notification system. The migrated applications were selected keeping in mind the incompatibilities among the services provided by these platforms.

The migration of the first application, the asynchronous discussion forum, was slow and problematic because of incomplete implementations of the Groupware Platform Specification File (GPSF) and the Service Manager. The implementation and appropriate use of these two components were challenging tasks. Once the first application was translated, the rest of the applications were easy to migrate. The effort spent to adjust the source code of the client application was within the estimated limits. However, small adjustments were required to the GPC middleware in order to better fit the migration process. The authors' feeling is that most of the problems were because it was the first test for the GPC middleware; therefore, many problems were made clear in such instance. In addition, the lack of expertise in the setting of the GPC middleware was another cause for problems encountered during the first migration process. Currently, most of the services provided by the JSDT platform [3] have been included in the current version of GPC middleware. They have been partially tested and successful results have been obtained. The process to include support for JSDT in the middleware was, at the moment, faster and easier than for the previous platforms.

The authors believe an important effort will be required to include support for every white-box groupware platform. However, once the support is well implemented, the groupware applications migration will be fast and with low cost.

6 Conclusions and Further Work

Many groupware applications developed on top of platforms supporting collaborative work are limited in their portability due to the lack of compatibility among the services they provide. Currently there is not a specific solution to help improve the portability of groupware applications supported by white-box server platforms. This paper presents the GPC (Groupware Platform Compatibility) middleware, which offers an option to overcome some of these limitations using a set of common groupware services. GPC works as a bridge between the groupware applications (clients) and the white-box groupware platforms (servers). This hides incompatibilities among the

requested and the given services. GPC can be seen like an ODBC/JDBC driver, which translates requests and responses from clients and servers to a compliant format and meaning. This strategy has strengths and weaknesses. An important strength is the small changes in the source code of groupware applications, and a small set up effort required by the new solution. In addition, this solution is light-weight and independent of the operating systems used by groupware application and server platforms. On the other hand, a limitation of GPC is that it does not take maximum advantage of the server platform full functionality, since the collaboration should be based on the groupware services provided by the middleware. In addition, there would be a performance reduction for groupware applications using this solution. Despite the limitations, GPC provides a way to improve the portability of groupware applications. The current supported white-box platforms are TOP and TOP II. Three applications have been ported from TOP to TOP II platform. The preliminary results have shown a high portability of these applications. Although it is early to obtain conclusions, important advantages on reuse of groupware applications could be obtained if the initial results endure. Currently, the authors are working to include support for JSDT [3] soon and other platforms later, in order to get well supported conclusions on the portability features provided by GPC.

Acknowledgments

This work was partially supported by Fondecyt (Chile) grants No. 1040952 and 1030959 and by MECESUP (Chile) project No. UCH0109.

References

1. Brown, A., Large-scale component-based development. Object and Component Technology Series, Prentice Hall., (2002)
2. Burner, M., The deliberate revolution: creating connectedness with XML Web services. ACM QUEUE. 1 (1), (2003), 28-37
3. Burridge, R., Java Shared Data Toolkit: user guide. Sun Microsystems, Inc., (1998)
4. Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., Seguin, C., Java object-sharing in habanero. Comm. of the ACM 41, 6, (1998), 69-76
5. Fabre,Y., Pitel, G., Soubrevilla, L., Marchand, E., Géraud, T., Demaille, A., Asynchronous architecture to manage communication, display, and user interaction in distributed virtual environments. Proc. of EGVE'2000, J.D. Mulder and R. van Liere (Eds.). Computer Science / Eurographics Series, Springer-Verlag, (2000), 105-113
6. Gokhale, A., Natarajan, B., Schmidt, D., Wang, N., Modeling and synthesis of middleware components. Communications of the ACM, Special Issue on Enterprise Components, Services and Business Rules, edited by Ali Arsanjani, (2002)
7. Greenberg, S., Roseman, M., Groupware toolkits for synchronous work. Beaudouin-Lafon, ed., Computer-Supported Cooperative Work, Chapt. 6, John Wiley & Sons, (1999), 135-168
8. Guerrero, L., Fuller, D., A pattern system for the development of collaborative applications. Information and Software Technology 43, 7, (2001), 457-467
9. Ochoa, S., Guerrero, L., Fuller, D., Herrera, O., Designing the communications infrastructure of groupware systems. In J. Haake, J.A. Pino (eds.): Groupware: Design, Implementation and Use. Lecture Notes in Computer Science 2440, (2002), 114-123

10. Peterson, R., Database development with Jdbc, Odbc and SQL/Sqlj. Sams Publishing, (2001)
11. Devsphere., SAX + DOM Mix = SAXDOMIX. URL: www.devsphere.com/xml/saxdomix/
12. Schuckmann, C., Schümmer, J., Seitz, P., Modeling collaboration using shared objects. In: S. C. Hayne (ed.): Proc. of ACM SIGGROUP Conf. on Supporting Group Work (GROUP'99). Phoenix, Arizona, USA, 189-198, (1999)
13. Szyperski, C., Component software. Addison-Wesley, (2002)
14. Trevor, J., Koch, T., Woetzel, G., MetaWeb: bringing synchronous groupware to the World Wide Web. Proc. of ECSCW'97, Lancaster, (1997)