An Architectural Pattern for Mobile Groupware Platforms

Andrés Neyem[1], Sergio F. Ochoa[2], José A. Pino[2], and Dario Franco[3]

[1] Department of Computer Science, Pontificia Universidad Católica de Chile
aneyem@ing.puc.cl
[2] Department of Computer Science, Universidad de Chile
{sochoa, jpino}@dcc.uchile.cl
[3] CIGIP Group, Universidad Politécnica de Valencia, Spain
dfranco@cigip.upv.es

Abstract. Architectural patterns have been proposed in many domains to capture recurring design problems that arise in specific design situations. This article presents an architectural pattern to help modeling the coordination and communication services required by mobile groupware platforms. This pattern serves as educational media for developers, students or researchers on how to design coordination and communication services for mobile groupware systems. The pattern also fosters the reuse of proven solutions.

Keywords: Coordination patterns, Communication patterns, Mobile Groupware, Mobile collaboration.

1 Introduction

Mobile groupware platforms can be built in various ways, used on several devices, operate over many different networks, and integrate with various back-end systems. The task of building a mobile groupware solution can often be daunting given the many technology choices and implementation approaches. Thus, the software architecture becomes the key for the development of these systems.

This paper presents an architectural pattern which can be advantageously applied in the development of mobile groupware platforms. The architectural pattern presents solutions for recurring problems associated with coordination and communication services required to support mobile collaboration.

In addition, the article presents an autonomous software infrastructure as an implementation of the proposed architectural pattern. This infrastructure contains generic components to support development, deployment and execution of mobile groupware applications. Examples of functionality provided by its components include provision of services to coordinate the operations on shared resources and services. The reuse of services is the most emphasized benefit of using services platforms support. Embedding common complex tasks into the platform and making them uniformly available for reuse can greatly enhance the efficiency of application development.

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 401–410, 2009.

Next section presents the main challenges to be met when designing a solution to support mobile groupware systems. Section 3 describes the architectural pattern and its implementation to deal with the design challenges of coordination and communication services. Section 4 discusses related work. Section 5 presents the conclusions and future work.

2 Requirements for Mobile Groupware Systems

This section describes general requirements that are usually present in the development of mobile groupware systems. Some requirements influencing mobile collaboration, in terms of communication and coordination support, are the following ones:

- *Flexibility*: Mobile groupware systems must support frequent changes in group size and structure, as mobility may cause group participants to connect or disconnect from the group [12]. Two mechanisms to provide flexibility are the following ones: automatic user detection and user connection / disconnection. The first one provides contextual information to implement awareness mechanisms. The second one allows applications to work offline and switch to online use on-demand.
- *Shared Information Support*: Frequent disconnections and autonomous work usually cause inconsistencies and unavailability of the resources which are being shared by the group members [12]. Some requirements supporting consistency and availability of data are explicit data replication, caching and conflict resolution.
- *Mobile Awareness*: Since a mobile groupware system supports collaborative work, it must provide awareness mechanisms to improve the users' understanding of the group's work [16]. The system should offer offline and online awareness, both of which should be updated as information becomes available.
- *Safety*: Mobile groupware systems must incorporate measures ensuring the work of each user is protected [13]. Some of these measures are: work sessions, user privacy and security.
- *Integration*: Regardless of the mobile device used to access the mobile application, a person should be able to interact with any collaborator using the same mobile groupware application [12]. Differences among devices are a burden to the users who should be eased by the collaborative system. Heterogeneity and interoperability are requirements which must be considered.
- *Communication*: Mobile groupware system users need to communicate with each other by exchanging messages [16]. Since the users are on the move to carry out their activities, some of them could be unreachable during some time period. Therefore, the system should provide mechanisms for synchronous/asynchronous communication and attended/unattended message delivery.
- *Networking*: Many work scenarios do not provide wireless communication support; in those cases the mobile groupware application should work based on a Mobile Ad-hoc Network (MANET) [12]. Networking issues should be transparent for the end-user. Otherwise, the collaboration capability is at risk.

3 An Architectural Pattern

A layered and fully-distributed architecture is recommended for mobile groupware applications since collaboration is based on communication and coordination [6]. The advantages of the layered architecture have already been discussed and recognized by the software engineering community [4]. Fig. 1 shows the *Crosslayer* pattern, which structures the basic functionality of a mobile groupware system.

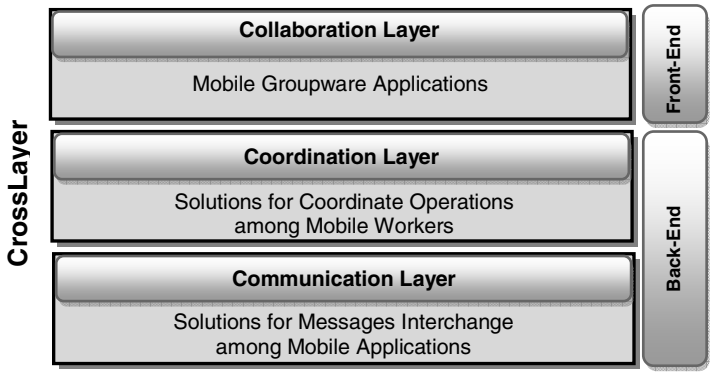

Fig. 1. Layered architecture to support mobile collaboration

The coordination layer provides solutions to support the typical groupware design aspects, but considering work contexts that involve unstable communication services. The communication layer focuses on the provision of services for message interchange among mobile workers' applications. Next section briefly describes this pattern following the nomenclature proposed in [18].

3.1 Context and Problem

A mobile groupware environment consists of various independent applications, distributed over several mobile nodes connected via a wireless network. These applications need to interact with each other, exchanging data or accessing each other's services in order to coordinate the work performed by a group of collaborators who pursue a common goal. It is clear that collaboration support requires communication and coordination services [6]. Thus, mobile groupware systems require separate functionality in three basic concerns: communication, coordination and collaboration. Each layer provides services and records data related to such services. These services are different in terms of concerns and granularity. The interaction between services related to two concerns is hierarchical: communication <-> coordination and coordination <-> collaboration. Integration of these services is required to support mobile collaboration, because frequently the service provider and the consumer run on different computing devices. Therefore, if the services provided by the mobile groupware system are not well structured, the system will be limited in terms of scalability, maintainability and adaptability.

3.2 Solution

The services related to different concerns can be grouped in distinct layers (Fig. 1). Designers can thus separate concerns and increase the system scalability, maintainability and adaptability. The lower layer provides services for message interchange. The coordination layer focuses on coordinating the operations of mobile workers in order to provide a consistent view of the group tasks. Finally, the collaboration layer provides support for managing interactions among mobile workers trying to reach the group goals. Next subsections present the solutions used to structure the coordination and communication services.

3.2.1 Coordination Solutions

The coordination solutions concern the provision of services required by mobile workers' applications to coordinate the operations of mobile workers on the shared resources (e.g. files, sessions and services). These solutions use services provided by the communication layer. This layer must separate functionality in the three basic concerns and implement it in components such as the following ones:

Distributed Sessions, Users and Roles Management. This component provides services which allow multiple work sessions involving users playing several roles. The rights are related to the role each user has for each session s/he is working on. Sessions, users and roles management should be fully-distributed since the workers have to keep their autonomy. Moreover, the loosely coupled work requires on-demand collaboration, information sharing and data synchronization; thus, explicit session management [16] should be provided by this component. In explicit sessions, participants must intentionally connect with other clients in order to interchange information or carry out opportunistic collaboration. The type of explicit work sessions matching loosely coupled work are the following ones: ad-hoc, public-subscribe and private-subscribe.

Distributed Management of Shared and Private Resources. This component should provide several services for every mobile user. First, a local (private) repository to store the private resources and second, a shared (public) repository to store resources the users want to share. The shared repository contains two types of information resources: reconcilable and irreconcilable. A reconcilable resource is a piece of information which can be synchronized with other copies of such resource in order to obtain a consistent representation of it. On the other hand, the irreconcilable resources are pieces of information which cannot be synchronized. Typically, the system has no information about the internal structure of these files. These resources are shared through file transfer mechanisms.

Context Management. This component must provide functionality for everything that can influence the behavior of mobile groupware applications. It includes hardware resources of computing devices and also external factors (e.g. bandwidth or quality of the network connection). This component has to be fully distributed and it must store, update and monitor current status of the context. This context manager has also to be carefully engineered in order to reduce the use of limited resources, such as battery, CPU, memory or network bandwidth.

Figure 2 shows the class diagram representing the structure that provides the functionality for the three main concerns. Group A are classes for services which allow multiple work sessions involving users playing several roles. Group B, C and D are classes for managing shared and private resources. Finally, group E are classes for context management. A detailed description of this structure can be found in [13].

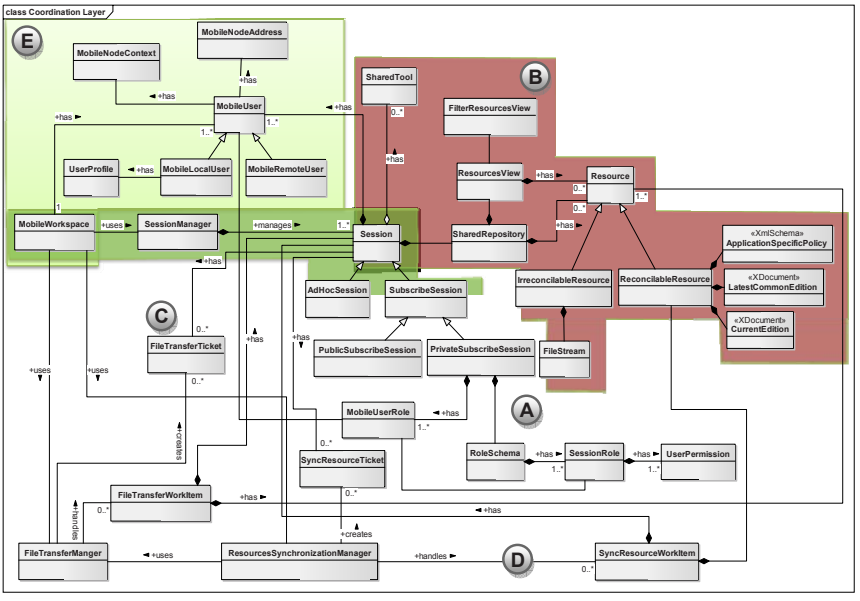

Fig. 2. Class diagram of the coordination layer

3.2.2 Communication Solutions

The communication layer is typically in charge of providing the support for messages interchange among mobile units. This component allows a user to send a message to other users in the wireless network in several ways: to those connected to a session, to a specified group of users, or to a single user. Based on that infrastructure, a mobile groupware application can send messages to other users. This layer must separate functionality in two main concerns: *communication middleware* and *wireless routing*.

Communication Middleware. This component provides a common programming model able to support several communication strategies, which can help developers to minimize the complexity of a mobile groupware application. The architecture of this component separates the design concerns in two layers: *service* and *messaging layer*.

The service layer provides a service-oriented approach and it lets mobile groupware systems to be extensible, flexible and interoperable in terms of services discovery, consumption and provision [12]. Thus, this layer provides functionality for service design and execution. The service external structure includes a service description and a collection of endpoints (Fig. 3). The service description provides essential information about what particular services (i.e. functions) this component

can provide and how they can be accessed. These service descriptions are specified and communicated using several standards. The internal structure of a service contains the binding protocol, contracts and implementation code (Fig. 3). The binding protocol describes how a service should be accessed. The contracts describe contextual information related to such service, e.g. its behavior, structure, or understandable message format. The implementation contains all the code that will be executed once the service is invoked.

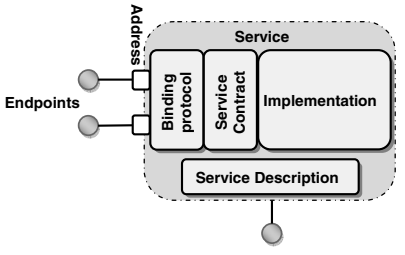

Fig. 3. Internal view of a service

A fundamental part of the service design is the definition of contracts. There are two kinds of contracts: a) *Service contracts*, which describe the operations a service can perform, and b) *Data contracts*, which define information structures passed to service operations. The endpoints are in charge of linking a contract and a binding protocol with a service address.

On the other hand, the messaging layer provides the general model allowing the message communication. In this layer, the messages are serialized and transmitted using the selected transport, protocol rules (like reliable messaging) and security policies (like encryption). A key component of this layer is the channel, which represents the highway over which the messages travel. This channel is an abstraction hiding all the underlying details of the communication process. For example, before two mobile groupware applications can exchange messages, a channel must be established between them. The first step involves a client trying to create a channel towards the endpoint of the service provider. If the service is available and reachable at that destination address, a channel can be established and the message exchange can take place. Once communication is completed, the channel can be turned down.

Wireless Routing Layer. This layer provides a transparent solution when mobile groupware applications are running in ad hoc wireless mode. The solution involves the provision of a message router which consumes messages from different message channels, and it split them in packets which are routed to the receiver. Finally, this router rebuilds the message based on the received packets. The packet delivery service should offer an intermediate solution between the routing and flooding techniques in order to achieve high reliability with moderate degradation of performance.

Figure 4 describes the structure including the main concerns of the communication layer. A detailed description of these functionalities can be found in [12]. Services are represented as a series of objects containing the service description, program code and runtime information. The service description includes the service contracts, behaviors and endpoints. The program code for the service is represented through the service type. The runtime information includes channels, instances of the service type, and extensions to the service. For each service there exists a master class, representing a service at runtime, called *ServiceDispatcher*. The ServiceDispatcher object contains the service type. When a ServiceDispatcher object is created, the service type is specified. From the client side, this application is responsible for triggering the

processing in a service-oriented solution by initiating the communication. Since services can call other services, client code can appear in both client and service programs. Any program initiating messages is acting as a client, even if that program is also a service. Clients talk to services via proxies. A client accesses service operations by calling proxy methods.

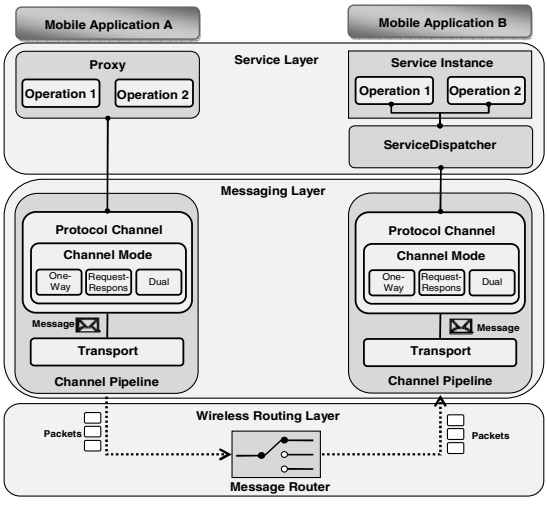

Fig. 4. Conceptual overview of the communication layer

The transport indicates the protocol the channel is going to use. If the transport is TCP/IP then the applications will interact with other mobile applications using sockets. The channel transport is always the lowest level of the channel component. From the service consumer's perspective, the channel is the last component in the communication pipeline. However, from the receiver's perspective, it is the first one. Over the transport, several channel protocols can be implemented. These protocols are responsible for performing additional functionalities on the message objects, e.g. encryption/des-encryption, or format transformations.

Finally, the channel mode allows the channel pipeline to change the messaging patterns. Three messaging patterns can be used: *One-way, Request-Response* and *Dual.* These patterns are not all necessarily available for any transport protocol. For example, the transport does not support dual communication for Message Queuing.

3.3 Pattern Implementation

A pattern provides a generic solution for a recurring problem: a solution which can be implemented in many ways without necessarily being 'twice the same' [4]. This section briefly presents the implementation of the architectural pattern defined in the previous section. A detailed description of these functionalities can be found in [13].

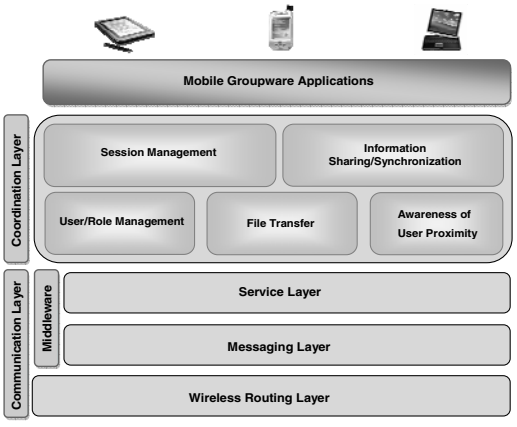

Fig. 5. SOMU basic architecture

The Service-Oriented Mobile Unit (SOMU) is a lightweight platform running over wireless networks (i.e. infrastructure and ad hoc mode), where the mobile nodes compose a mesh. The platform enables each mobile computing device to produce and consume services from other peers connected to the mesh [13]. The SOMU architecture consists of components organized in two layers (see Fig. 5): coordination and communication. The layers manage the corresponding groupware services and a shared data space. These components communicate with the adjacent one through an API. Mobile groupware applications developed on SOMU can use the general solutions implemented in that platform. These solutions include the management of sessions, users, messages delivery, shared objects and repositories; and it partially allows management of the work context. Mobile groupware applications inherit the capabilities for interacting with applications running on others mobile units. It helps developers to focus on the application's main goal, freeing them to deal with low-level interaction processes (i.e. communication and coordination).

Developed applications over this platform include mobile collaborative software to support disaster relief operations [10], to conduct inspections on construction sites [15] and to manage exams in computer science courses [14].

4 Related Work

There are several experiences reporting the use of mobile groupware applications [3], [7]. Although some of these applications are fully-distributed, they do not describe or evaluate the strategies used to deal with the typical coordination services (e.g. distributed sessions, shared information support and context management) and communication services in mobile groupware (e.g. communication channels, protocols and routing). Thus, the potential design solutions cannot be evaluated when they are formalized through design patterns or reused in future applications. Schümmer and Lukosch argue groupware reuse should focus on design reuse rather than code reuse [18]. These researchers also propose a patterns system for groupware for stationary scenarios, therefore they do not consider the users mobility.

Jørstad et al. have proposed and described a set of generic coordination services for distributed (but stable) work scenarios [8]. These services include locking, presentation control, user presence management and communication control. Other researchers have also proposed similar solutions to support coordination and communication on fixed networks [2]. However, the contextual variables influencing the collaboration scenario and the mobile work make such solutions unsuitable to support mobile collaboration.

On the other hand, there are several middleware and platforms that provide different mechanisms and techniques to enable communication between distributed components [4]. Examples of these middleware are RPC-based middleware (e.g. JINI [1]), message-oriented middleware (e.g. Sun's Java Message Queue [11]), publish-subscribe middleware (e.g. JEDI [5]), tuple-based middleware system derived from LINDA (e.g. FT-LINDA, PLinda, TSpaces, Lime, JavaSpaces, and TOTA [17]) and data sharing-oriented middleware (e.g. XMIDDLE [9]). However, a middleware able to combine more than one approach is required in order to cover a wide range of mobile work scenarios. For example, the asynchronous model of interaction between hosts, such as the message-oriented style, would offer potential solutions and drastically enhance middleware possibilities in ad-hoc environments. In addition, keeping the middleware lightweight and taking the context awareness as a functional requirement could help dealing with the instability of the work contexts. Furthermore, efficient adaptable resource discovery mechanisms, specifically intended for mobile ad hoc environments, could provide robustness and flexibility to the solutions, mainly in terms of handling the frequent changes in network components and topology.

5 Conclusions and Further Work

Developing mobile groupware platforms to support collaborative work is a challenging task, since this is a recent area and software must overcome challenges in technical implementation, collaboration support and users' adoption.

This paper presents an architectural pattern to support the design of coordination and communication services required by mobile groupware systems. This pattern deals with most of the stated requirements and serves as educational and communicative media for developers, students or researchers on how to design coordination and communication mechanisms for mobile groupware applications. It also fosters the reuse of proven solutions.

At the moment, the architectural pattern has shown to be useful to design both, mobile groupware applications and a middleware to support collaborative systems [13]. The reuse of these designs and the implementation of the proposed solutions have been quite simple. However, the authors have been involved in each one of these testing experiences. Therefore, future work is required to carry out evaluations with external groupware developers in order to determine the real contribution of this proposal. Moreover, the solutions proposed by the architectural pattern should be extended to support devices using mobile telephone systems (i.e. 3.5G).

Acknowledgements. This work was partially supported by Fondecyt (Chile), grants Nº: 11060467 and 1080352 and LACCIR grant Nº R0308LAC005.

References

1. Arnold, K., O'Sullivan, B., Scheifler, R.W., Waldo, J., Wollrath, A.: The Jini Specification. Addison-Wesley, Reading (1999)
2. Avgeriou, P., Tandler, P.: Architectural Patterns for Collaborative Applications. Int. J. of Computer Applications in Technology 25(2/3), 86–101 (2006)

3. Baloian, N., Zurita, G., Antunes, P., Baytelman, F.: A Flexible, Lightweight Middleware Supporting the Development of Distributed Applications across Platforms. In: 11th Int. Conf. on Comp. Supported Coop. Work in Design, Australia, pp. 92–97 (2007)
4. Buschmann, F., Henney, K., Schmidt, D.C.: Pattern-Oriented Software Architecture. A Pattern Language for Distributed Computing, vol. 4. John Wiley & Sons, Chichester (2007)
5. Cugola, G., Di Nitto, E., Fuggetta, A.: The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. IEEE Trans. on Software Engineering 27(9), 827–850 (2001)
6. Ellis, C.A., Gibbs, S., Rein, G.L.: Groupware: some issues and experiences. Communications of the ACM 43(1), 38–58 (1991)
7. Herskovic, V., Ochoa, S.F., Pino, J.A.: Modeling Groupware for Mobile Collaborative Work. In: 13th Int. Conf. on Comp. Supported Coop. Work in Design, pp. 384–389. IEEE CS Press, Chile (2009)
8. Jørstad, I., Dustdar, S., Van Thanh, D.: Service Oriented Architecture Framework for collaborative services. In: 14th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pp. 121–125. IEEE Press, New York (2005)
9. Mascolo, C., Capra, L., Zachariadis, S., Emmerich, W.: XMIDDLE: A Data-Sharing Middleware for Mobile Computing. J. on Pers. and Wireless Comm. 21(1), 77–103 (2002)
10. Monares, A., Ochoa, S.F., Pino, J.A., Herskovic, V., Neyem, A.: MobileMap: A collaborative application to support emergency situations in urban areas. In: 13th Int. Conf. on Comp. Supported Coop. Work in Design, Chile, pp. 432–437 (2009)
11. Monson-Haefel, R., Chappell, D.A., Loukides, M.: Java Message Service. O'Reilly & Associates, Sebastopol (2000)
12. Neyem, A., Ochoa, S., Pino, J.: Integrating Service-Oriented Mobile Units to Support Collaboration in Ad-hoc Scenarios. J. of Universal Comp. Science 14(1), 88–122 (2008)
13. Neyem, A.: A Framework for Supporting Development of Groupware Systems for Mobile Communication Infrastructures. Ph.D. Thesis, Universidad de Chile (November 2008)
14. Ochoa, S.F., Neyem, A., Bravo, G., Ormeño, E.: MOCET: a MObile Collaborative Examination Tool. In: Smith, M.J., Salvendy, G. (eds.) HCII 2007. LNCS, vol. 4558, pp. 440–449. Springer, Heidelberg (2007)
15. Ochoa, S.F., Pino, J.A., Bravo, G., Dujovne, N., Neyem, A.: Mobile Shared Workspaces to Support Construction Inspection Activities. In: IFIP Int. Conf. on Collaborative Decision Making (CDM), France, pp. 270–280 (2008)
16. Pinelle, D., Gutwin, C.: A Groupware Design Framework for Loosely Coupled Workgroups. In: 9th Europ. Conf. on CSCW, pp. 65–82 (2005)
17. Russello, G., Chaudron, M., van Steen, M.: An experimental evaluation of self-managing availability in shared data spaces. Science of Comp. Programming 64(2), 246–262 (2007)
18. Schümmer, T., Lukosch, S.: Patterns for Computer-Mediated Interaction. John Wiley & Sons, West Sussex (2007)