The Journal of Systems and Software 85 (2012) 511–524

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A reusable structural design for mobile collaborative applications

Andrés Neyem[a,*], Sergio F. Ochoa[b], José A. Pino[b], Rubén Darío Franco[c]

[a] Department of Computer Science, Pontificia Universidad Católica de Chile, Av. Vicuña Mackena 4860, Macul, Santiago, Chile
[b] Department of Computer Science, Universidad de Chile, Av. Blanco Encalada 2120, Santiago, Chile
[c] CIGIP Group, Universidad Politécnica de Valencia, Spain

ARTICLE INFO

Article history:
Received 28 May 2010
Received in revised form 20 May 2011
Accepted 24 May 2011
Available online 7 June 2011

Keywords:
Reusable architecture
Coordination services design
Communication services design
Mobile collaborative applications

ABSTRACT

Architecting mobile collaborative applications has always been a challenge for designers. However, counting on a structural design as a reference can help developers to reduce risks and efforts involved in system design. This article presents a reusable architecture which helps modeling the communication and coordination services required by mobile collaborative applications to support collaboration among users. This architecture has been used as a basis for the design of several mobile systems. Two of them are presented in this article to show the applicability of the proposal to real world collaborative systems.

1. Introduction

The complexity of designing and communicating designs has been recognized by many designers and researchers since the medieval age. This socio-technical activity, although expensive, is the key to conceive suitable, usable and implementable solutions (Hubka and Eder, 1987; Bucciarelli, 1994; Eckert et al., 2005). Given the complexity of such process, the reuse of design knowledge is valued as a way to reduce the costs and difficulties of creating new design solutions. Well-known examples of reusable designs are software patterns (Gamma et al., 1994; Buschmann et al., 2007). These abstract structures capture the essence of a design solution and allow inexperienced developers to design as expert ones.

The definition of reusable designs is more challenging in novel application scenarios, such as computer supported mobile collaborative work, because the knowledge for such application domains is being built every day. Herskovic et al. (2011) have recognized and discussed the difficulties involved in the design of mobile collaborative applications, since many groupware services are not visible for designers. These applications involve communication and coordination services that are intended to be run on several devices and integrate with various back-end systems. Building a mobile collaborative solution can often be daunting given the many technology choices and implementation approaches. Thus, the software architecture becomes the key element for the development of mobile collaborative systems.

This paper presents a reusable architecture which can be advantageously applied in the development of several mobile collaborative systems. The architecture presents particular design solutions to address the challenge of modeling coordination and communication services required to support mobile collaboration. The article also introduces an autonomous software infrastructure, named SOMU (Service-Oriented Mobile Unit) (Neyem et al., 2008), which implements the proposed designs for the communication and coordination services. SOMU contains a set of generic components that supports development, deployment and execution of mobile collaborative applications. Examples of functionality provided by these components include provision of services to coordinate the operations on shared resources and support interactions among mobile users. The reuse of services is the most emphasized benefit of using this type of supporting platforms. Thus it is possible to enhance the efficiency of applications development.

Next section presents the main challenges to be met when designing a solution to support mobile collaborative systems. Section 3 presents and discusses related work. Section 4 describes the proposed reusable architecture. Section 5 introduces the SOMU platform and shows how it adheres to the proposed architecture. Section 6 presents two mobile collaborative applications that also adhere to the reusable structural design. Section 7 presents the conclusions and future work.

* Corresponding author. Tel.: +56 2 3547550; fax: ++56 2 354 4444.
E-mail addresses: aneyem@ing.puc.cl (A. Neyem), sochoa@dcc.uchile.cl (S.F. Ochoa), jpino@dcc.uchile.cl (J.A. Pino), dfranco@cigip.upv.es (R.D. Franco).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.05.046

2. Requirements for mobile collaborative systems

"Essentially a system's utility is determined by both its functionality and its non-functional characteristics, such as usability, flexibility, performance, interoperability and security" (Chung and Leite, 2009). In that sense mobile collaborative systems are not an exception. The authors have specified a list of transversal requirements that are usually satisfied when developing mobile collaborative systems elsewhere (Herskovic et al., 2011). These requirements are classified into seven categories: users' interaction flexibility, users' interaction protection, communication, heterogeneity and interoperability, autonomous interaction-support services, user awareness, and data consistency and availability. This section describes these requirements to show the complexity behind the design of mobile collaborative applications. It also discusses the relationships among these requirements.

- *Users' interaction flexibility* (we will refer to this requirement as *flexibility*): The work context is dynamic in mobile activities. Therefore mobile collaborative applications must react to changes in the environment, such as changes in group size and structure, or in the availability of shared resources due to users' mobility (Neyem et al., 2008). 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.
- *Users' interaction protection* (we will refer to this requirement as *protection*). This requirement considers the capability of the mobile application to protect the work and resources of a mobile user from possible unauthorized attempts of other users (Kortuem et al., 2001). Some mechanisms used to provide protection are ad hoc sessions, explicit user privacy and users' identity verification. The first one provides mobile users an interaction space that can be protected on-demand depending on users' requirements; e.g. sessions can be public, private or by invitation. The second mechanism (i.e. explicit user privacy) allows users to indicate the information they want to share with their collaborators. It is done by copying such information in the public space each mobile user must have. The third mechanism considers not only to verify the users' identity when they login to an ad hoc session, but also to do it in each access to the shared resources. The level of users' identity verification can be set on-demand.
- *Communication*. Communication is the basis that supports coordination and collaboration (Ellis et al., 1991). Typically the interaction among mobile users involves exchanging several resources, such as documents, messages or alarms (Caporuscio and Invernardi, 2003). Since it is not possible to guarantee the availability of a communication channel in mobile collaboration when a user decides to interact with other people, several communication services must be considered to support mobile work. Some of these services are synchronous/asynchronous messaging, file transfer and pushing notifications.
- *Heterogeneity and interoperability* (we will refer to this requirement as *heterogeneity*). The type of device utilized by mobile users cannot be a limitation to perform on-demand interactions among them. Therefore mobile collaborative applications must interoperate in terms of data and services (Neyem et al., 2008). Two well-known strategies must be used to address this challenge: using standard technologies and contextual information. The first one supports the interoperability requirements and the second one allows dealing with the devices heterogeneity. Neyem et al. (2008) propose to use a mobile units' profile to help collaborative applications to address the heterogeneity in the work scenario.

Table 1
Relating general requirements (Herskovic et al., 2011).

	Protection	Communication	Heterogeneity	Networking	Awareness	Information Support
Flexibility		-		-	-	-
Protection		-			-	
Communication			-			
Heterogeneity						
Networking						
Awareness						+

+ : x_i positively impacts x_j
- : x_i negatively impacts x_j

- *Autonomous interaction support services* (we will refer to this requirement as *networking*). Users' mobility generates frequent disconnections/reconnections, which must be transparent for the end-user because of usability reasons (Rodríguez-Covili et al., 2011b). Typically when the work scenario does not provide wireless communication support, collaborative applications must create a Mobile Ad hoc Network (MANET) to support users' interactions (Neyem et al., 2008). Some mechanisms that can be used to help provide connectivity in unstable communication scenarios are automatic MANET formation and topology management, service and device discovery, message routing, gossip delivery and automatic user connection (Herskovic et al., 2011).
- *Users awareness* (we will refer to this requirement as *awareness*). Since the interaction among users in mobile collaboration is performed on-demand, identifying the availability of potential collaborators is mandatory in this type of systems (Pinelle and Gutwin, 2005). A well-known strategy to address this requirement involves the use of awareness mechanisms (Papadopoulos, 2006). Some of the awareness mechanisms that can be used by mobile collaborative applications to ease on-demand users interactions are the following ones: users' reachability (i.e. connected/disconnected), users' availability (i.e. available/busy) and notification of users' presence/availability (Herskovic et al., 2009).
- *Data consistency and availability* (we will refer to this requirement as *information support*). The offline work and the frequent disconnections of mobile users typically generate inconsistency and unavailability of the shared data (Neyem et al., 2008). Therefore mobile collaborative applications must provide mechanisms to address such situation. Some mechanisms that can be used are the following ones: explicit data replication, caching and conflict resolutions (Herskovic et al., 2011).

Table 1 shows a correspondence matrix among these requirements. The impact of a requirement may be positive (if it contributes to the accomplishment of the other), negative (the opposite case) or neutral (both requirements are independent).

Typically solutions providing flexibility to mobile collaborative applications impact negatively the capabilities for communication, networking, awareness and information support. The main cause is that flexibility helps increase the autonomous offline work, which decreases the cohesion of the work sessions' members and the interaction capability among them.

Solutions addressing protection requirements affect negatively communication and awareness capabilities. This occurs because

such solutions typically reduce the visibility that other users have about the user being protected. The protection services also reduce the reachability of protected users.

The communication services are negatively affected by the heterogeneity. This happens because the more heterogeneous a group is in terms of devices and software, the more difficult it is to build interoperable communications services.

Finally, information supporting services positively impact awareness mechanisms embedded in the application, since shared information about users can be transferred or synchronized among the mobile devices both automatically or on-demand.

The architecture of these systems must be fully distributed considering that mobile collaborative applications must support flexibility of the users' interactions and collaboration on-demand in an unstable communication scenario. This is a consequence of the fact the accessibility to data and services – which is required by the mobile workers to perform their activities – should not depend on centralized components. Centralized components usually reduce the availability of the system in mobile work scenarios (Neyem et al., 2008; Rodríguez-Covili et al., 2011a). Moreover, the rapid change and heterogeneity of the work scenario force the application to be context-aware and interoperable. Typically architectures including fine-grain components are easier to self-adapt when the work context changes, than those embedding coarse-grain components. Next section presents and discusses the related work.

3. Related work

Most research works in this area are more focused on presenting the use and impact of mobile collaborative applications than describing the design of these solutions. Many applications have been reported to support mobile collaboration in several areas such as education (Ochoa et al., 2007), healthcare (Morán et al., 2007), emergency support (Monares et al., 2011), m-commerce (Tarasewich, 2003), and productive activities (Ochoa et al., 2008). Some of these applications are fully distributed, which are appropriate to support mobile collaboration. However, there are no explanations about the strategies used to deal with the typical coordination (e.g. distributed sessions and shared information support) and communication services (e.g. communication channels and routing) required for mobile collaboration. Thus, the potential design solutions behind these implementations cannot be reused.

Several approaches have been proposed trying to reuse design knowledge in the computer supported collaborative work (CSCW) domain. Some of them are design patterns and pattern languages, particular architectures, frameworks and toolkits.

A groupware pattern is a structured description of a solution to a recurrent problem in a particular CSCW context (Avgeriou and Tandler, 2006; Dearden and Finlay, 2006). These patterns use specific examples, state the groupware problem that they address, and deliberately scope their context of application. However, taken in isolation, patterns only represent unrelated good ideas; thus, a pattern language is required to provide coherent support for design generation. A pattern language provides a taxonomy to enable designers to find patterns, to find related or proximal patterns, to evaluate the problem from various perspectives, and to develop new solutions (Dearden and Finlay, 2006). Schümmer and Lukosch (2007) argue the reuse should focus on design reuse rather than code reuse. These researchers also propose a patterns language for collaborative stationary scenarios, therefore they do not consider the users' mobility.

Groupware architectures determine the nature of the CSCW application components and their location on the devices of the various participants in a collaborative effort. According to Phillips (1999), there are three distinct architectural views in synchronous groupware: (i) reference models, which partition complete systems into named functional elements and specify how data flows between those elements; (ii) architectural styles, which prescribe the type of components and connectors and their allowed interaction patterns; and (iii) distribution architectures, which represent the distribution of system functionality across connected computing devices. The relation among these views can be stated as "A system may be best understood (as a whole) in terms of a particular reference model, designed in whole or in part according to a particular architectural style, and implemented using a particular distribution architecture" (Phillips, 1999). The proposed architecture can be considered mostly as a guideline for designers, as it aims to specify the complete structure of mobile collaborative applications, at a high abstraction level, by creating a conceptual structure consisting of communication services, coordination mechanisms and interactions spaces for collaboration. Each group of solutions has well defined interfaces and data flows among these groups.

Several researchers have proposed architectures to deal with particular problems/challenges. For example, Buschmann et al. (2007) presented a catalog of typical patterns to distributed computing. This catalog presents a set of general architectural patterns that could help developers to create sustainable designs for this kind of systems by designing new applications and improving and refactoring existing ones. Although such patterns are conceptually relevant, they do not consider the support required for mobile collaboration.

Medvidovic and Edwards (2010) provide an overview of the intersection between software architecture and mobility. Their work is more focused on mobile software than mobile computing; therefore it is not particularly suitable to support the design of mobile collaborative systems.

Sama et al. (2010) report that a classical architectural style for a mobile context-aware adaptive application is typically layered and it tends to incorporate context-awareness components to support processing context values. Such components are responsible for triggering adaptive changes in the application. The proposal is interesting to support the design of mobile context-aware applications, but not particularly mobile collaborative systems. Typical design issues such as sessions and user management, or shared spaces are not considered in such proposal. Like the pervious case, Fortier et al. (2010) propose mechanisms for handling variability, during the evolution of a single mobile context-aware architecture and across different domains. They also propose design structures and their underlying rationale, in order to deal with mobility requirements, such as location sensing, behavior adaptation, and context variability. This work has the same limitation than the previous one.

Dewan (1999) proposes a reference model that encapsulates architectural properties, common to a wide range of collaborative systems. He identifies a set of design issues (e.g. single-user architecture, collaboration awareness and versioning/replication) that any groupware architecture must deal with. Based on Dewan's work, Laurillau and Nigay (2002) proposed Clover's architecture, which defines the classes of services that must be supported by a groupware application. Duque et al. (2008) propose an architectural model for the development of groupware systems incorporating analysis facilities. It can be used by groupware developers as a guide to the integration of analysis subsystems into groupware applications. Although these three proposals are interesting they do not consider users' mobility because they were conceived with another goal; therefore they are not suitable to guide the design of mobile collaborative applications.

Phillips et al. (2005) and then Rodríguez-Covili et al. (2011a) proposed particular layered architectures to design workspaces. Although both of them consider users' mobility, they are focused on a particular type of mobile groupware application: mobile workspaces.

Groupware frameworks have also been utilized as vehicles to reuse design solutions in the groupware area. These frameworks provide a structure to organize thinking about particular aspects of support, a vocabulary for analyzing activities during collaboration and for comparing solutions on these aspects, and a set of starting points from which initial solutions could be further developed and refined (Pinelle and Gutwin, 2005). High-level frameworks typically describe particular aspects of groupware support at a conceptual level, such as component elements, mechanisms used to provide actual support and their uses for collaboration. The low-level frameworks usually include implementations on specific languages and object hierarchies to facilitate development (e.g. DOORS, Preguica et al., 2005; Manifold Framework, Marsic, 2001), and design constructs such as multi-level architectures that separate design concerns (e.g. presentation layer, domain logic, and collaboration logic).

The reusable architecture presented in this article does not represent a pattern since it has not been discovered through the analysis of different applications by different groups. The architecture was created based on the lessons learned by the authors during more than six years designing mobile collaborative applications. Such architecture describes in abstract form and by means of examples a solution to the recurrent problem of designing mobile collaborative applications. Also, it explicitly establishes the context of use, the problem to address and a proposed solution. The architecture has been tested in practice and it has shown to be useful to guide the design of several mobile collaborative applications. Next section describes the proposed structural design.

4. The proposed reusable architecture

A layered and fully distributed architecture is recommended for mobile collaborative applications since collaboration is based on communication and coordination (Ellis et al., 1991). The advantages of the layered architecture have already been discussed and recognized by the software engineering community (Buschmann et al., 2007). Fig. 1 shows the *Crosslayer* architecture, which structures the basic functionality of a mobile collaborative application.

The collaboration layer provides solutions to support functionalities that must be exposed to end-users by a specific mobile application. These solutions are mainly related to the application front-end and use services provided by the coordination layer. The coordination layer provides solutions to support the typical groupware design aspects (e.g. distributed sessions management and information sharing), 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. The relationship between layers is hierarchical, and the interaction among components belonging to two adjacent layers can be done directly between the components or through a programming interface. On the one hand, the direct interactions help to improve the application performance, which is important if we consider that many of these systems run on handheld devices. Thus, direct interactions jeopardize the maintainability and flexibility of the application. In that sense it is better to implement the interactions between layers through the programming interface. Next section briefly describes this architecture following the nomenclature proposed by Schümmer and Lukosch (2007).

4.1. 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.

Applications supporting mobile workers' activities must allow them to work autonomously and collaborate on-demand. However, the access to centralized resources (e.g. a server acting as coordinator) cannot be ensured in such scenario, because the communication support required to access them may not exist or be highly unstable.

The work context may change frequently due to the users' movement or changes in the features of the physical facilities where the user is located. Therefore, the mobile application must detect changes in the work context and self-adapt its services depending on them. All functionality supporting mobile collaboration must be as automatic and seamless as possible because of usability reasons.

It is well known that collaboration support requires communication and coordination services (Ellis et al., 1991). Thus, mobile collaborative 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 (Ellis et al., 1991; Rodríguez-Covili et al., 2011a): 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 collaborative system are not well structured, then the system will have limitations in terms of scalability, maintainability and adaptability.

4.2. Solution

The services related to different concerns can be grouped in distinct layers (Fig. 1). Therefore, high-level components depend on low-level components to perform their functionality, which further depend on even lower-level components and so on. Decoupling the components in a vertical manner helps designers to separate concerns and increase the system scalability, maintainability and adaptability.

The lowest 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.

At first glance, this layered solution may seem similar to the Model-View-Controller (MVC) architecture (Buschmann et al., 2007); however, they are structurally different since the proposed architecture is linear and the MVC architecture is triangular (i.e. the view sends updates to the controller, the controller updates the model, and the view gets updated directly from the model). Moreover the roles played by the architectural component in both cases are quite different. Next subsections present the solutions used to structure the collaboration, coordination and communication services.

4.2.1. Collaboration solutions

Collaboration solutions concern the provision of collaborative functionalities (or services) that a specific application must expose to end-users. These solutions are mainly related to the application front-end and use services provided by the coordination layer. Next we present a brief description of two particular mobile applications which will be used as examples to help understand the requirements and solutions involved in the collaborative applications layer.

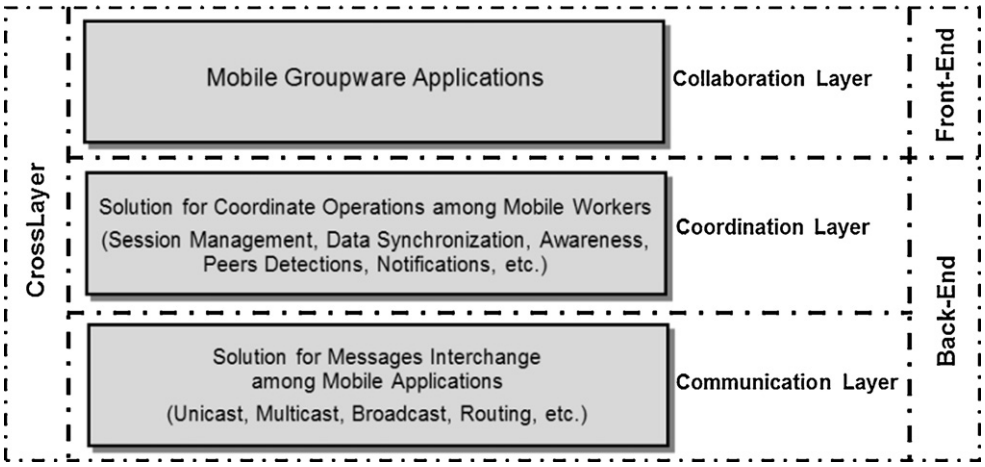

Fig. 1. Layere architecture to support mobile collaboration.

The first application supports firefighters during urban emergency situations (Monares et al., 2011). Firemen attending common emergencies (e.g. a fire or car accident) must make decisions when travelling to the emergency site and also during the emergency response process. Making such decisions requires knowing information about the emergency place, the contingency situation to address and the status of the response process. Therefore, the solutions for this scenario must improve the group decision-making processes and the coordination of efforts when an emergency situation happens. For example, the mobile collaborative application could show a city map where it is possible to identify the emergency site, the fire trucks location, and the location of interest points, such as hospitals or police departments near the emergency place. This information is shared and updated on-demand by firemen in the field and also by firefighters who are travelling to the emergency site.

The second scenario concerns the design of technological support for construction personnel inspecting the physical infrastructure of building projects (Ochoa et al., 2008). Typically each construction site has a main contractor. The main contractor in turn outsources several parts of the construction project, e.g. electrical facilities, gas/water/communication networks, painting and architecture. Some of these sub-contracted companies work concurrently and they have to collaborate in order to know each other's progress to plan the execution of their own pending work. Moreover, all these companies should periodically collaborate with the main contractor in order to report their work progress. The contractor is in charge of coordinating the efforts of the subcontracted companies. The solutions for this scenario could provide a shared workspace allowing users to manage the blueprint of the construction project, to do annotations on the maps, and to synchronize and share the annotations/maps using a network.

4.2.2. Coordination solutions

The coordination solutions concern the provision of services required by mobile workers' applications to coordinate the users' operations 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 main concerns and it could implement this functionality through various components such as the following ones.

Distributed sessions, users and roles management. This component provides services which allow multiple work sessions with users playing several roles (see classes belonging to Group A in Fig. 2). 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 (Pinelle and Gutwin, 2005) 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 types 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 (see classes belonging to Groups B, C and D in Fig. 2). 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 data 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 data which cannot be synchronized. The system typically 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 collaborative applications (Group E in Fig. 2). 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 and network bandwidth.

Fig. 2 shows the class diagram representing the structure that provides the functionality for the three main concerns. A detailed description of this structure can be found in Neyem (2008).

4.2.3. 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 collaborative 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

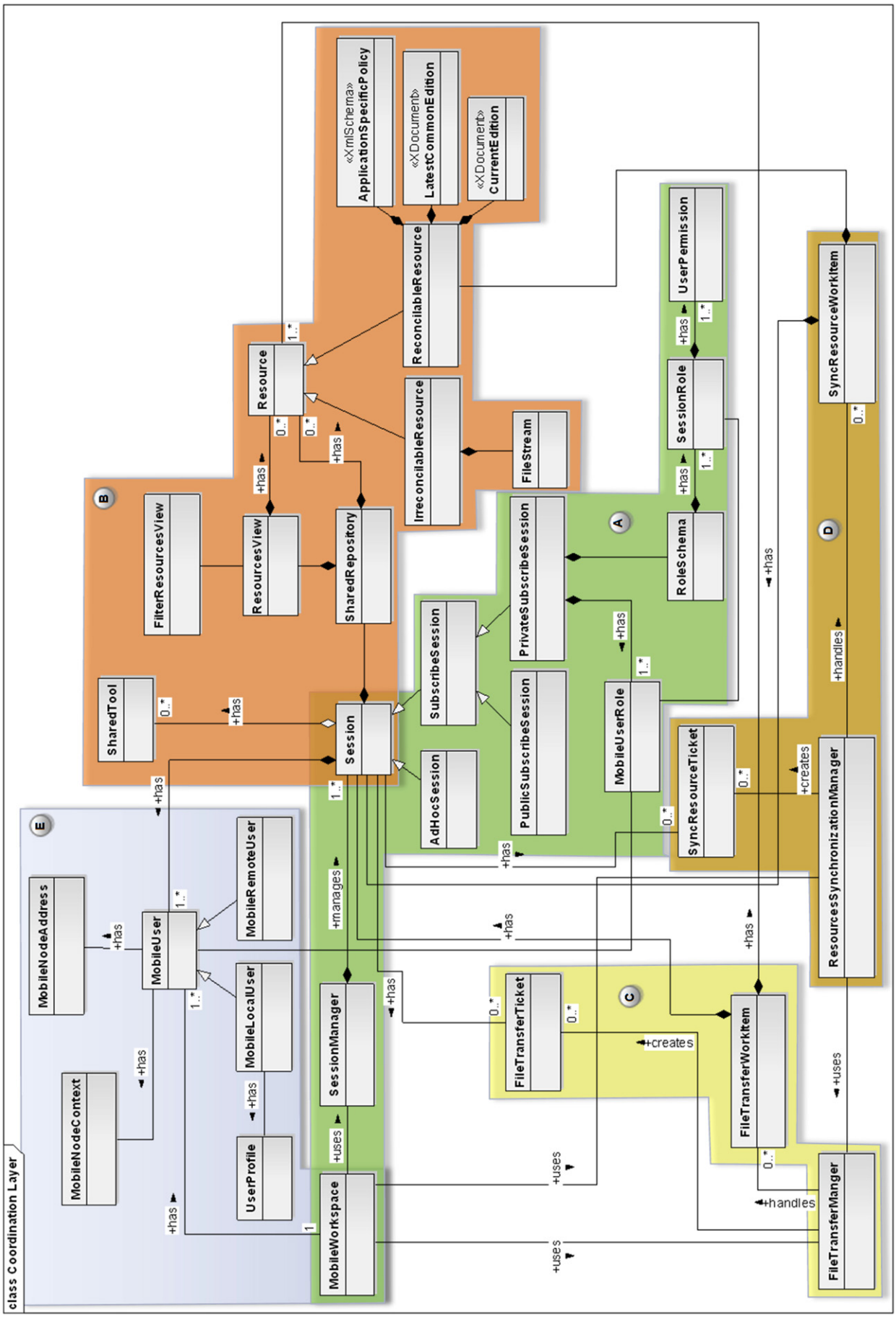

Fig. 2. Class diagram of the coordination layer.

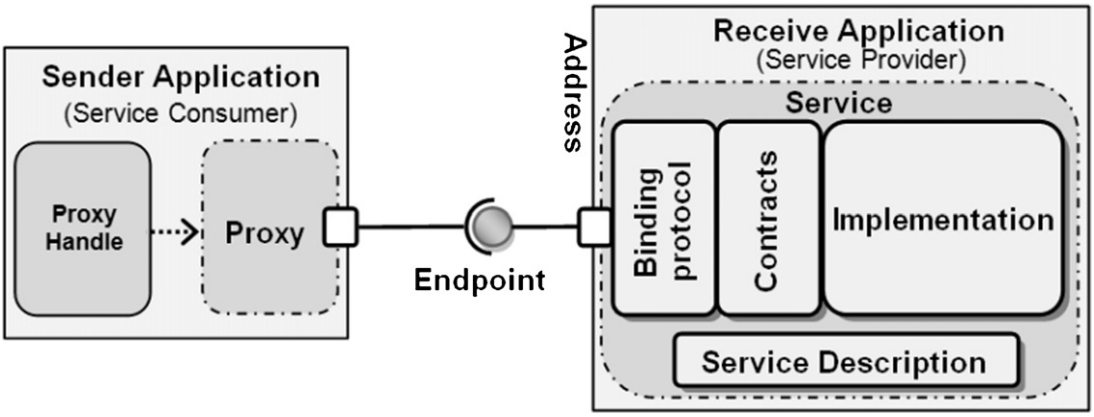

Fig. 3. Relationship between service provider and consumer.

strategies, which can help developers to minimize the complexity of a mobile collaborative application. Although the services provided by this middleware are similar to those provided by the traditional ones (i.e. for stable communication scenarios), the services implementation requirements are quite different because the work scenarios involve other communication conditions. For example, a server (or any centralized component) cannot be used as coordinator because it is not possible to ensure availability of such resource when mobile workers decide to collaborate (Neyem et al., 2008). This implies the communication middleware that support the mobile collaboration must be fully distributed. In addition, the network supporting the communication has a highly changing topology due to the users' movements. It means that, e.g. in a file transfer, the packages start to be sent on a certain network topology and finish using a really different one. It represents a serious challenge to implement various services such as messages routing, mobile users detection, and notifications delivery. These problems are not present in the work scenario used by traditional communication middleware where the communication link is assumed to be available and stable.

The architecture of the proposed communication middleware separates the design concerns in two layers: service and messaging. The service layer provides a service-oriented approach and it lets mobile collaborative systems to be extensible, flexible and interoperable in terms of services discovery, consumption and provision (Neyem et al., 2008). The term *service* refers to a loosely coupled reusable component that encapsulates discrete functionality, which may be distributed and programmatically accessed (Turner et al., 2003). Services exist as physically independent software component with distinct design characteristics that support the attainment of the strategic goals associated with Service-Oriented Computing (SOC) (Erl, 2007). Thus, this layer provides functionality for service design and execution. The service external structure includes a service description and an endpoint (Fig. 3). The service description is specified and communicated using several standards. This service description provides essential information about which particular services (i.e. functions) this component can provide and how they can be accessed by proxies belonging to service consumers. The proxy provides a service interface that allows mobile collaborative applications to treat the remote services as local ones.

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. Thus, contracts define certain aspects of the service, such as the format and structure of the messages, which are sent between the service provider and consumer. Both the service provider and the consumer must agree on the type of operations and structures they will use during the interaction period. 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 implementation contains all the code that will be executed once the service is invoked. The endpoint is in charge of linking a contract and a binding protocol with a service address.

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 (Fig. 4), which represents the pathway over which the messages travel. This channel is an abstraction hiding all the underlying details of the communication process. For example, before two mobile collaborative 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 collaborative applications are running in ad hoc wireless mode. This mode refers to a wireless network without fixed infrastructure (i.e. access points or fixed antennas). When the nodes are assumed to be capable of moving, either on their own or carried by their users, these networks are called MANETs. The network nodes rely on wireless communication to collaborate with each other. The advantage of ad hoc networking is that the absence of a fixed infrastructure reduces the cost, complexity, time required to deploy the network and it allows nodes to be on the move (Neyem et al., 2008).

The solution in this layer involves the provision of a message router which provides routing support in MANETs. Routing capabilities are vital not only to increase the communication threshold but also to reduce the cost of messages delivery. Thus, the message router consumes messages from various message channels, and it splits 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 better performance in terms of reliability and energy consumption (Abbas and Kure, 2010). This intermediate solution involves routing with a settable level of redundancy. Redundancy can be discarded depending on the reliability required in the communication process.

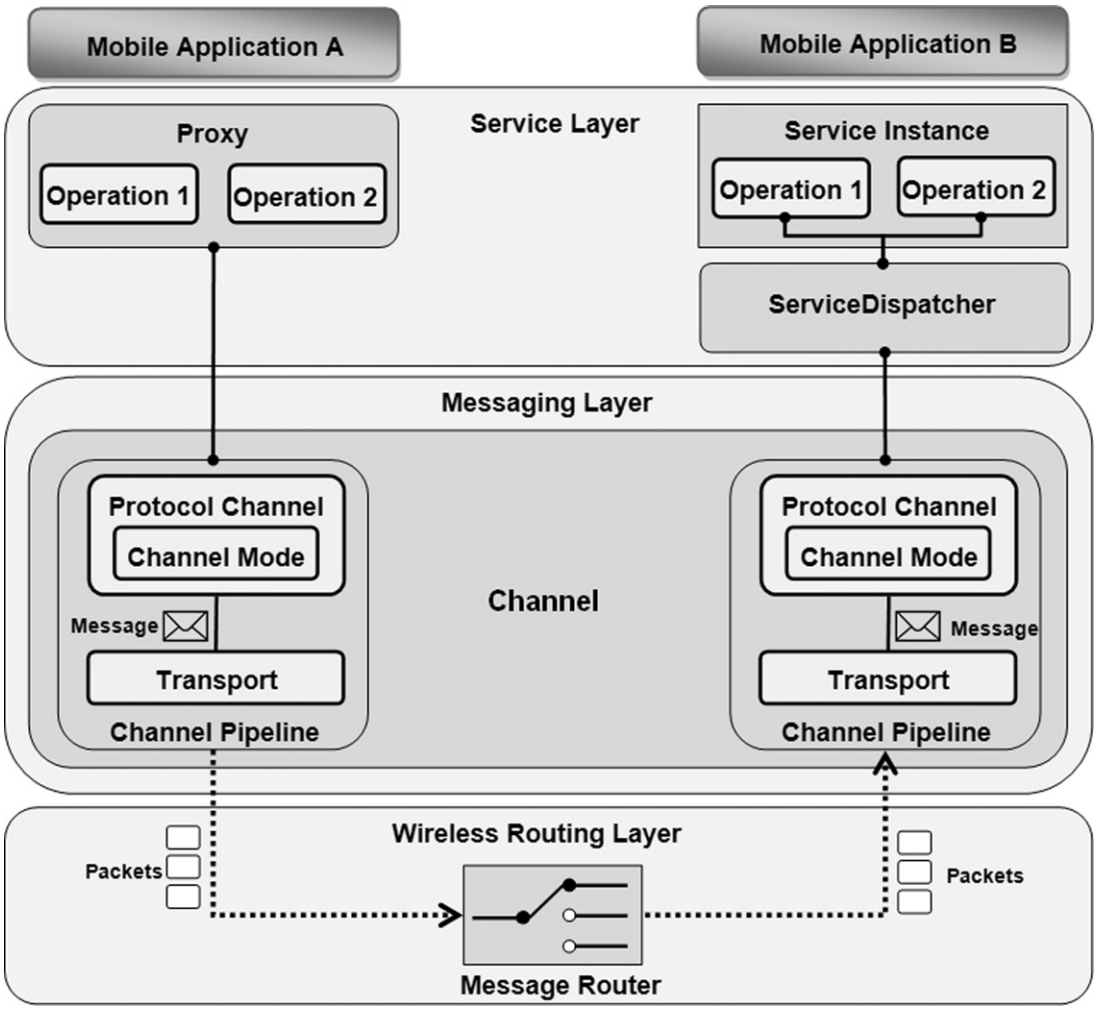

Fig. 4. Conceptual design of the communication layer.

Fig. 4 describes the structure including the main concerns of the communication layer. A detailed description of these functionalities can be found in Neyem et al. (2008). 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.

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. The channel is the last component in the communication pipeline from the service consumer's perspective. By contrast, from the receiver's perspective, it is the first one. Several channel protocols can be implemented over the transport. These protocols are responsible for performing additional functionalities on the message objects, e.g. encryption/decryption, or format transformations.

Finally, the channel mode allows the channel pipeline to change the messaging models. Three messaging models can be used: *One-way, Request-Response* and *Dual.* These delivery strategies are not all necessarily available for any transport protocol. For example, the transport does not support dual communication for Message Queuing. Next section introduces implementation of the proposed architecture, which was done on the SOMU platform (Neyem et al., 2008).

5. SOMU platform

The Service-Oriented Mobile Unit (SOMU) is a lightweight collaboration platform, able to run over wireless networks with or without infrastructure. By default the platform works in ad hoc wireless mode (i.e. without infrastructure). However if mobile users know that an infrastructure-based wireless network is available in the work scenario (e.g. a corporative wireless network) they can set the platform on-demand, and indicate they want to use such network. This situation typically occurs when the mobile work is performed at a specific place, such as a hospital, a school or an offices building. Using an infrastructure-based wireless network frequently helps to increase the communication threshold and bandwidth during mobile collaborative activities. However, it reduces the flexibility of the solution because the collaboration

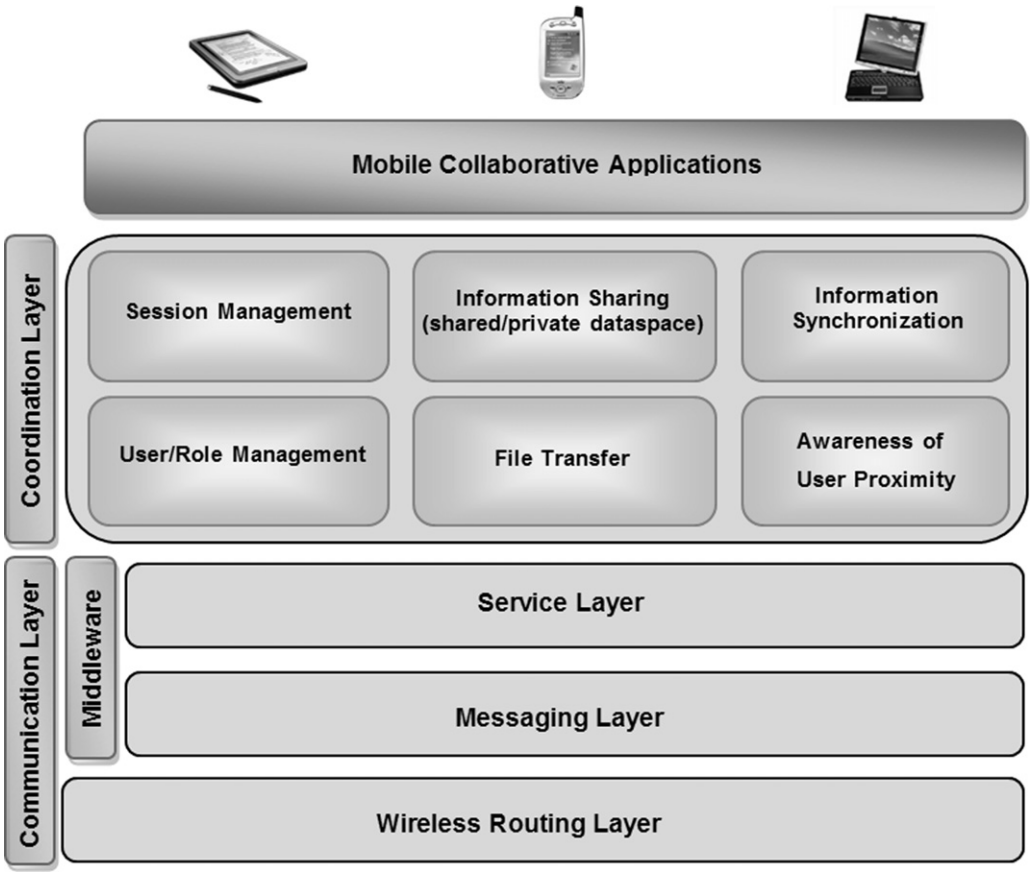

Fig. 5. SOMU basic architecture.

activities are supported just at locations within the network scope. The platform is fully distributed and it enables each mobile computing device to produce and consume services from other peers connected to the MANET. It is described in detail in Neyem et al. (2008).

The SOMU architecture consists of components organized in two layers (Fig. 5): coordination and communication. These layers manage the corresponding collaborative services and a shared data space. Mobile collaborative applications developed on SOMU can use the general communication and coordination solutions encapsulated in the platform. Thus, developers can reuse the implemented designs. These solutions include the management of sessions, users, messages delivery, shared objects and repositories; and it partially allows management of the work context. Mobile collaborative systems inherit the capabilities for interacting with applications running on other 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 services).

6. Applying the proposed architecture

This section shows how the design solutions embedded in the proposed architecture were reused for the design of two particular mobile collaborative applications. These applications, which were briefly introduced in Section 4.2.1, were developed by graduate students as part of their MSc. theses. These students did not participate in the definition process of the reusable architecture, and they voluntarily used it as support for their applications design.

The process of reusing the architecture in order to develop new mobile collaborative systems involves four stages: (1) modeling the interaction scenario, (2) identifying the requirements involved in the application, (3) identifying the services that will allow the system to provide such functionality, and (4) reusing the components providing services from SOMU. Modeling the interaction scenario was done with the MCM (Mobile Collaboration Modeling) language (Herskovic et al., 2009) to represent all interaction among mobile devices that would be required in the work scenario. Moreover, the language also allows designers to identify which user roles would be involved in each interaction type. The result of this activity is an interaction graph, which can be input to a software module. The output of running this module is a list of requirements that must be included in the application in order to support the previously described interactions (Herskovic, 2010). During the third stage, the designer has to identify which components of the architecture provide services needed to address the requirements of the list. Typically they are components of the coordination layer, because all components of the communication layer are usually required. Finally, the designer has a basic architecture that will allow users to perform the interactions described by the MCM graph. The components of such basic architecture can be reused from SOMU or from another framework providing them.

6.1. MobileMap

MobileMap is a fully distributed mobile collaborative application that allows firefighters to share information during an urban emergency (e.g. a fire or car accident), using a peer-to-peer or an ad hoc interaction strategy (Monares et al., 2011). This application

Fig. 6. MobileMap application.

is routinely used on laptops and smartphones (Fig. 6a and b) depending on the user's role. MobileMap is set to utilize the 3G network as the default communication support. In that case the application uses a peer-to-peer communication strategy. All information shared by the peers can be locally stored by any other node connected to the same work session (i.e. an emergency response process). When the 3G network is unavailable, the application automatically switches to the ad hoc mode.

Devices running MobileMap have locally pre-stored much emergency support data, including city maps, location of hydrants, hospitals and schools. Therefore just information concerning the particular emergency relief operation is exchanged among the network members. This reduces the network traffic and increases the shared information availability.

The response to urban emergencies is typically in charge of an incident commander, who is in the field to manage the emergency response process. Several other roles are also played by firefighters in the field; for example, communication officer, rescuers, and response and logistics personnel. All of them make decisions in a distributed way, based on the available shared information. Such decisions must be coordinated to ensure the response process success. Fig. 6c shows the shared information firemen get with MobileMap. The application runs on PDAs, smartphones and laptop/tablet PC.

6.1.1. MobileMap architecture

The architecture of MobileMap is layered (Fig. 7) and it adheres to the cross layer architecture. The collaboration layer implements the front-end and uses services provided by the coordination layer. The *UsersView* (in coordination layer) shows a collection of all mobile users currently signed on to the MobileMap application. The *MapView* shows a map where it is possible to identify the emergency place, the fire trucks locations, and the location of interest points, such as hospitals or police departments near the emergency site. Such information can be stored locally in the mobile device or consumed from any other device member of the MANET. Fig. 6b shows its representation on the application's user interface. The *ResourcesView* shows a set of files (e.g. pictures of the current emergency or maps of the affected area) that are shared among firemen both in the field and going to the emergency place. Fig. 6c shows this component on the MobileMap user interface.

The coordination layer indicates the use of a *MobileMap Environment* which is an engine allowing the creation of workgroups. It allows the workgroup owners to populate the shared environment with the tools and contents required to solve the problem. The component is used to manage and provide general services to support the shared environment. The *Session Manager* carries out work with a session involving users playing roles. This manager records information about users, sessions, roles and shared

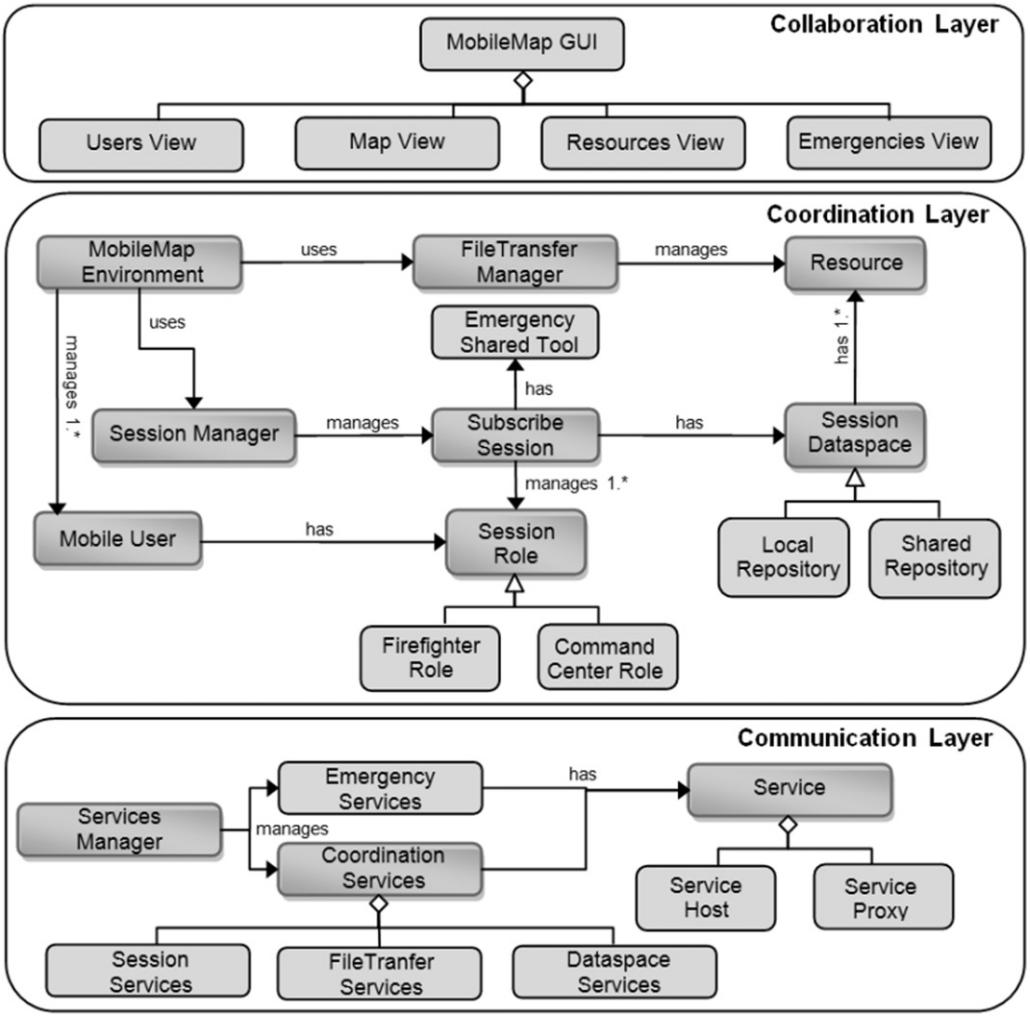

Fig. 7. Architecture of MobileMap.

resources. The *Session Role* is just informative data, since it is not possible to carry out a distributed control of it. Since this component records users and roles data, it is possible to provide users/roles awareness information to other users in the session.

The *Session Dataspace* is a component which records information about private and shared resources and it also lets users interact with the objects shared by other users logged into the session. This information is stored in files and the *FileTransfer Manager* provides a mechanism to manage all the file transfer processes in a transparent way. Finally, the emergency tool manages detailed information about emergencies and it is used by the Emergencies View.

Concerning the communication layer, it provides support for message interchange among users of the MobileMap application. The services are grouped as *Coordination Services* and *Emergency Services*. The first group includes the typical services that implement functionalities for managing sessions, shared resources and transfer processes. The second group includes specific application services implementing the communication aspect of the emergency tool. All these services are managed by a *Service Manager* component which is responsible for processing messages between host and proxy services.

6.1.2. MobileMap design evaluation

The design embedded in the application was evaluated through (1) focus groups and (2) the empirical use of the application in real emergencies. Three focus groups were done with firefighters who usually make decisions in the field, and some of them act as incident commanders. Each focus group had 5–7 persons belonging to various firefighting companies. The MobileMap functionality was explained in those meetings. Thereafter they were able to use the application to make decisions on a hypothetical response process to a fire that was happening in the Company quarter. The quarter has three floors and each floor has around 300 m^2 and the goal of the response process was to rescue three simulated victims.

The fireman playing the role of incident commander used a laptop (with MS Windows XP) and the rest of the team used several smartphones with MS Windows Mobile. A MANET was used as communication support. The simulated victims were located in the facilities and several areas were marked as “on fire” so that firemen must locate each victim and then find a path to rescue them. Three observers participated in the process by recording the exercise.

The goal of the evaluation process was to determine that MobileMap is functionally able to deal with mobile work during an emergency. Most requirements described in Section 2 were observed, particularly flexibility (i.e. automatic users’ detection and management of users’ connection/disconnection), protection (particularly the use of ad hoc sessions), communication (i.e. sharing resources), networking (i.e. communication on the MANET), heterogeneity (particularly interoperability between devices, and awareness of users’ presence and availability). All these requirements were considered important to support the response process of that emergency.

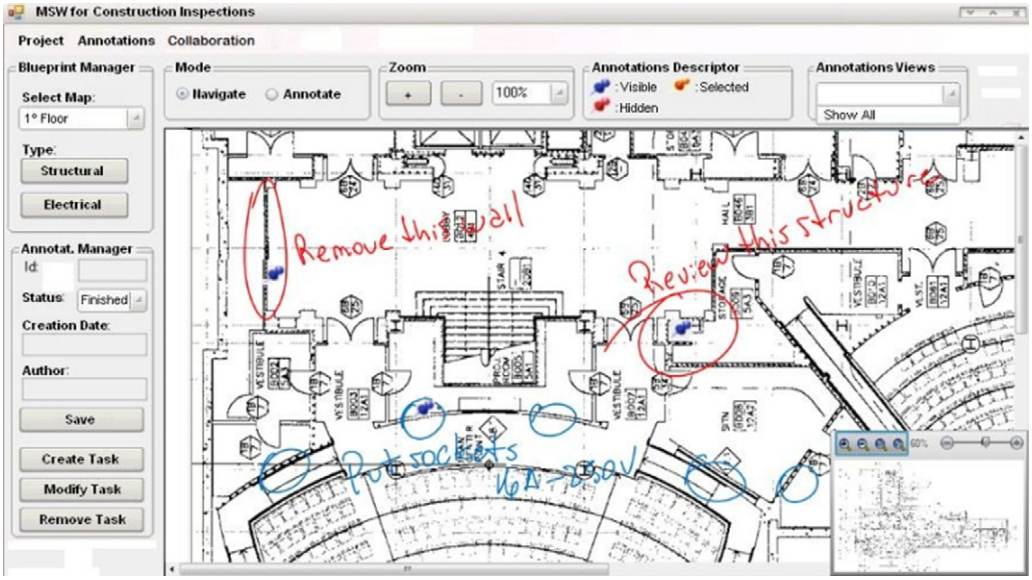

Fig. 8. COIN main user interface.

All participants were able to enter and leave the work session and also share information with the teammates. The communication support was able to manage the users' disconnection/reconnection to the MANET and also indicate (through the awareness mechanisms) when a mobile user is reachable and available. The heterogeneity of the devices was not a problem, but we must consider that all the devices used the same operating system family.

The interview performed after the exercise indicated the firemen felt comfortable using the application and they were motivated for trying to use the application afterwards in a real emergency scenario. The changes in the communication scenarios were automatically managed by the application and the users were not conscious of such changes. These preliminary results show the components included in the architecture played the roles which were designed for.

The Nunoa command center and the 2nd Fire Company (both from Santiago, Chile) used the tool in five typical urban emergencies. It was used to retrieve and share information about the emergency. In these cases the 3G network was used as communication support.

The obtained results were similar to those obtained with the focus groups and the details have been reported in Monares et al. (2011). These preliminary results indicate the architecture of MobileMap helps to coordinate firemen during urban emergencies.

Finally, the student in charge of designing this application found the proposed architecture intuitive and easy to use. Although he did not have experience designing coordination and communication services for collaborative systems, he felt the use of this solution helped him to find a sound design option. After this experience, the designer thinks he is able to apply this structural design to other application scenarios.

6.2. COIN (COnstruction INspector)

COIN is a mobile collaborative application which supports the work of inspectors in construction projects (Ochoa et al., 2008). Fig. 8 shows the main user interface of COIN. The inspection process typically involves three activities: *registration*, *validation* and *reporting*. Inspectors review part of the physical infrastructure (e.g. electrical facilities, water network or the painting) during the first activity and record the project advances through annotations on blueprints. When COIN is used to support such activity, the annotations can be done on digital blueprints using tablet PCs. During the validation activity, the inspectors physically meet to synchronize annotations and resolve contradictory annotations. If inspectors identify inconsistencies between annotations from two inspectors, then a new review is performed at the interest locations. Finally, the chief inspector reports the results to the contractor during the reporting activity.

6.2.1. COIN architecture

The COIN architecture (Fig. 9) is layered and adheres to the cross-layer architecture. The collaboration layer provides the functionality for handling several projects through the *Project View*. Each project is represented by an XML file storing all related information. The associated manipulation menu provides facilities for creating, opening, storing, saving, re-naming, importing/exporting and synchronizing projects. The *Blueprints View* is available once the inspector has selected a project to work on. A particular floor of the building can be selected to be inspected using the blueprints manager, and then a particular map can be loaded on the shared panel. It allows the inspector to start the inspection process. Through this view, inspectors are able to handle all information related to an annotation. It shows all the data related to the annotation being selected on the shared panel, e.g. author of the annotation, current state and creation date. The panel also allows changing the state of an annotation, and therefore the state of all the tasks related to it. Finally, the users have a *Resources View* to share resources that are important for the inspection process.

The coordination layer involves the use of *COIN Environment* which is an engine providing the same functionality of *MobileMap Environment* for managing multiple work sessions (e.g. for multiple inspections). Just private work sessions (i.e. a type of subscribe session) are supported because the information used and recorded during an inspection is private and it cannot be shared with persons outside the team. Each session has its own shared workspace through the use of the *Session Dataspace* component, list of mobile users and awareness mechanisms. Two users' roles are supported in COIN: inspector and chief inspector, and also two data replication mechanisms: file transfer and reconciliation through a data synchronization process. All these components adhere to the solution defined in the coordination layer.

The communication layer provides the typical services for message interchange among mobile applications (e.g. allowing a user

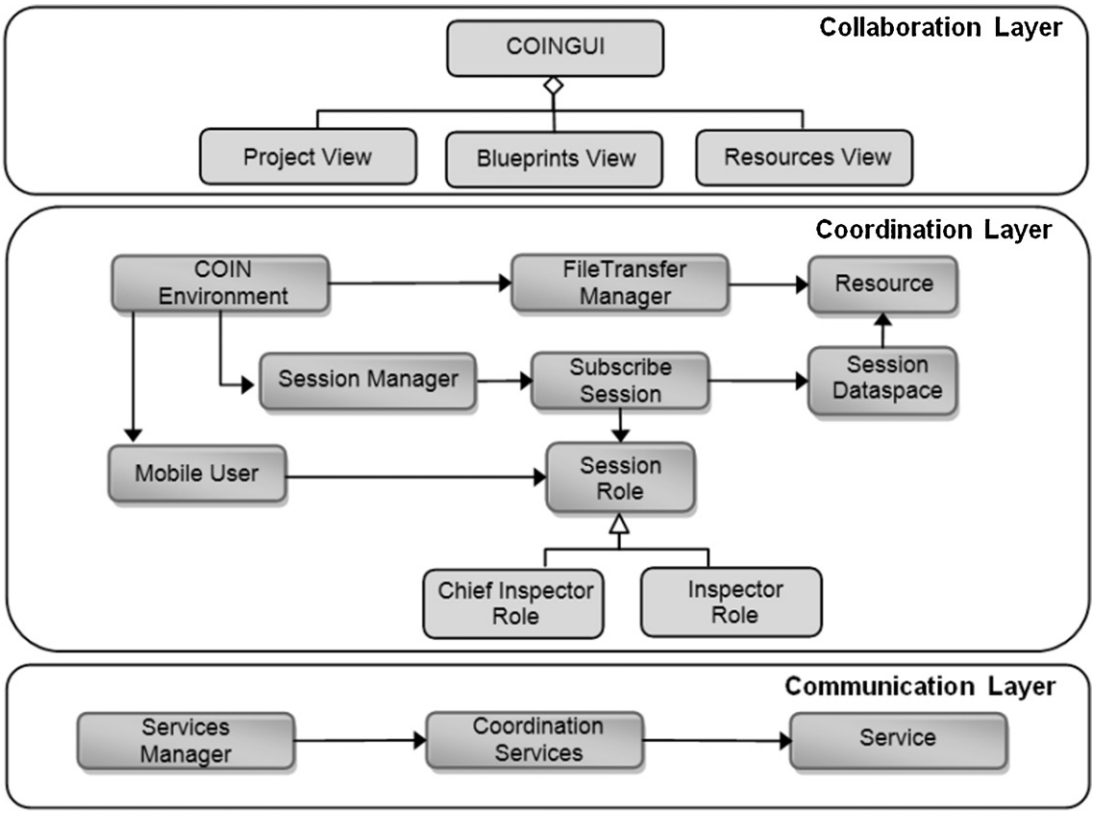

Fig. 9. Architecture of COIN.

to send a message to other users connected to a session, to a specified group of users, or to a single user). These services are used by the coordination layer for coordinating the operations of mobile workers in order to provide a consistent view of the group tasks.

6.2.2. COIN design evaluation

The design solutions embedded in COIN were evaluated and reported in Ochoa et al. (2008). The experimentation scenario involved a simulated construction project. The reviewing process considered two inspectors, who recorded contingency issues of the physical facilities. The results showed that inspectors using COIN were able to perform the three activities involved in this process. Moreover both felt comfortable using the solution. During this experiment we compared the inspection process using paper-based blueprints with a process using COIN with digital blueprints. The obtained results show the *registration* activity was a little bit favorable to the COIN usage; however, inspectors using the application in the *validation* and *reporting* stages completed their work several orders of magnitude faster than without it. Although the time involved in the registration activity was similar, inspectors preferred to use COIN instead of the paper-based blueprints because of the simplicity to handle these resources. Inspectors also found the functionality supporting mobile collaboration is appropriate to support this activity. Although the results are still preliminary, they indicate the proposed reusable architecture could be suitable to support coordination and communication in construction inspection scenarios.

7. Conclusions and further work

Developing mobile collaborative applications is a challenging task, since this is a recent area and software must overcome challenges in technical implementation, collaboration support and users' adoption. Designers of these applications have to invent particular solutions to deal with the systems modeling, because there are no reference models. Design solutions for stationary groupware systems are not suitable to support mobile collaboration, because most of them use centralized components that jeopardize the availability of communication or coordination services for mobile users.

This paper presents a reusable architecture to support the design of coordination and communication services required by mobile collaborative systems. This architecture deals with most of the requirements stated in Section 2, and it also serves as educational and communicative media for developers, students or researchers on how to design coordination and communication mechanisms for mobile collaborative applications. It also fosters the reuse of proven solutions.

At the moment, this architecture has shown to be useful to design both mobile collaborative applications (Ochoa et al., 2008; Monares et al., 2011) and a middleware to support the development of mobile collaborative systems (Neyem, 2008). The reuse of these designs and the implementation of the proposed solutions have been quite simple. However, the authors have been involved in most of the evaluation experiences. Therefore, future work is required to carry out evaluations activities with external collaborative developers in order to determine the real contribution of this proposal. Moreover, the solutions proposed by the architecture should be extended to support an ample variety of mobile computing devices, involving various hardware resources, computing power and user interaction paradigms.

Acknowledgements

This work was partially supported by Fondecyt (Chile), grant Nos: 11090224, 11060467 and 1080352 and LACCIR grant No: R0308LAC005.

References

Abbas, A.M., Kure, O., 2010. Quality of service in mobile ad hoc networks: a survey. International Journal of Ad hoc and Ubiquitous Computing 6 (2), 75–98.

Avgeriou, P., Tandler, P., 2006. Architectural patterns for collaborative applications. International Journal of Computer Applications in Technology 25 (2/3), 86–101.

Bucciarelli, L., 1994. Designing Engineers. MIT Press, Cambridge.

Buschmann, F., Henney, K., Schmidt, D.C., 2007. Pattern-Oriented Software Architecture. A Pattern Language for Distributed Computing, vol. 4. John Wiley & Sons.

Caporuscio, M., Invernardi, P., 2003. Yet another framework for supporting mobile and collaborative work. In: Proceedings of the International Workshop on Enabling Technologies ,. IEEE Press, Washington, DC, pp. 81–86.

Chung, L., Leite, J., 2009. On non-functional requirements in software engineering. LNCS 5600, 363–369.

Dearden, A., Finlay, J., 2006. Pattern languages in HCI: a critical review. Human Computer Interaction 21 (1), 49–102.

Dewan, P., 1999. Architectures for collaborative applications. In: Beaudouin-Lafon, M. (Ed.), Computer Supported Cooperative Work. John Wiley & Sons Ltd..

Duque, R., Rodríguez, M.L., Hurtado, M.V., Noguera, M., Bravo, C., 2008. An architecture to integrate automatic observation mechanisms for collaboration analysis in groupware. In: Proceedings of the OTM Workshops 2008. LNCS, vol. 5333 , pp. 354–363.

Eckert, C.M., Maier, A.M., McMahon, C., 2005. Communication in design. In: Clarkson, P.J., Eckert, C.M. (Eds.), Design Process Improvement – A Review of Current Practice. Springer, London.

Ellis, C.A., Gibbs, S., Rein, G.L., 1991. Groupware: some issues and experiences. Communications of the ACM 43 (1), 38–58.

Erl, T., 2007. SOA: Principles of Service Design. Prentice Hall.

Fortier, A., Rossi, G., Gordillo, S.E., Challiol, C., 2010. Dealing with variability in context-aware mobile software. Journal of Systems and Software 83 (6), 915–936.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1994. Design Patterns: Elements of Reusable Object Oriented Software. Addison-Wesley.

Herskovic, V., Ochoa, S.F., Pino, J.A., 2009. Modeling groupware for mobile collaborative work. In: Proceedings of the International Conference on Computer Supported Cooperative Work in Design ,. IEEE Press, Santiago, Chile, pp. 384–389.

Herskovic, V., 2010. Evaluation of Mobile Shared Workspaces to Improve their Support for Collaboration. Ph.D. Thesis, Universidad de Chile.

Herskovic, V., Ochoa, S.F., Pino, J.A., Neyem, A., 2011. The iceberg effect: behind the user interface of mobile collaborative systems. Journal of Universal Computer Science 17 (2), 183–202.

Hubka, V., Eder, E.W., 1987. A scientific approach to engineering design. Design Studies 8, 123–137.

Kortuem, G., Schneider, J., Preuitt, D., Thompson, T.G., Fickas, S., Segall, Z., 2001. When peer-to-peer comes face-to-face: collaborative peer-to-peer computing in mobile ad hoc networks. In: Proceedings of the International Conference on Peer-to-Peer Computing ,. IEEE Press, Washington, DC, pp. 75–91.

Laurillau, Y., Nigay, L., 2002. Clover architecture for groupware. In: Proceedings of the Conference on Computer-Supported Cooperative Work ,. ACM Press, USA, pp. 236–245.

Marsic, I., 2001. An architecture for heterogeneous groupware applications. In: Proceedings of the International Conference on Software Engineering ,. IEEE Press, Canada, pp. 475–484.

Medvidovic, N., Edwards, G., 2010. Software architecture and mobility: a roadmap. Journal of Systems and Software 83 (6), 885–898.

Monares, A., Ochoa, S.F., Pino, J.A., Herskovic, V., Rodriguez-Covili, J., Neyem, A., 2011. Mobile computing in urban emergency situations: improving the support to firefighters in the field. Expert Systems with Applications 38 (2), 1255–1267.

Morán, E.B., Tentori, M., Gonzalez, V.M., Favela, J., Martínez-Garcia, A.I., 2007. Mobility in hospital work: towards a pervasive computing hospital environment. International Journal of Electronic Healthcare 3, 72–89.

Neyem, A., Ochoa, S., Pino, J., 2008. Integrating service-oriented mobile units to support collaboration in ad-hoc scenarios. Journal of Universal Computer Science 14 (1), 88–122.

Neyem, A., 2008. A Framework for Supporting Development of Collaborative Systems for Mobile Communication Infrastructures. Ph.D. Thesis, Universidad de Chile.

Ochoa, S.F., Neyem, A., Bravo, G., Ormeño, E., 2007. MOCET: a MObile Collaborative Examination Tool. In: Proceedings of the 12th International Conference on Human-Computer Interaction (HCI). LNCS, vol. 4558 , pp. 440–449.

Ochoa, S.F., Pino, J.A., Bravo, G., Dujovne, N., Neyem, A., 2008. Mobile shared workspaces to support construction inspection activities. In: Proceedings of the IFIP International Conference on Collaborative Decision Making (CDM) , Toulouse, France, pp. 270–280.

Papadopoulos, C., 2006. Improving awareness in mobile CSCW. IEEE Transactions on Mobile Computing 5 (10), 1331–1346.

Phillips, W.G., 1999. Architectures for synchronous groupware. Technical Report 1999-425, Queen's University.

Phillips, W.G., Graham, T.C.N., Wolfe, C., 2005. A calculus for the refinement and evolution of multi-user mobile applications. In: Gilroy, S., Harrison, M. (Eds.), DSV-IS 2005, LNCS, vol. 3941. , pp. 137–148.

Pinelle, D., Gutwin, C., 2005. A collaborative design framework for loosely coupled workgroups. In: Proceedings of the 9th European Conference on CSCW , pp. 65–82.

Preguica, N., Martins, J.L., Domingos, H.J.L., Duarte, S., 2005. Integrating synchronous and asynchronous interactions in groupware applications. In: Proceedings of the 11th International Workshop on Groupware (CRIWG). LNCS, vol. 3706 , pp. 89–104.

Rodríguez-Covili, J., Ochoa, S.F., Pino, J.A., Herskovic, V., Favela, J., Mejía, D., Morán, A.L., 2011a. Towards a reference architecture for the design of mobile shared workspaces. Future Generation Computer Systems 27 (1), 109–118.

Rodríguez-Covili, J.F., Ochoa, S.F., Pino, J.A., Messeguer, R., Medina, E., Royo, D., 2011b. A communication infrastructure to ease the development of mobile collaborative applications. Journal of Network and Computer Applications.

Sama, M., Elbaum, S., Raimondi, F., Rosenblum, D.S., Wang, Z., 2010. Context-aware adaptive applications: fault patterns and their automated identification. IEEE Transactions on Software Engineering 36 (5), 644–661.

Schümmer, T., Lukosch, S., 2007. Patterns for Computer-Mediated Interaction. John Wiley & Sons, West Sussex, England.

Tarasewich, P., 2003. Designing mobile commerce applications. Communications of the ACM 46 (12), 57–60.

Turner, M., Budgen, D., Brereton, P., 2003. Turning software into a service. IEEE Computer 36 (10), 38–44.

Andrés Neyem is an Assistant Professor in the Computer Science Department at the Pontificia Universidad Católica de Chile. He received his Ph.D. in Computer Science from the Universidad de Chile. His research interests include mobile computing, software engineering and computer supported collaborative work. He has published several papers in conferences proceedings and journals in these research areas.

Sergio F. Ochoa is an Assistant Professor of Computer Science at the University of Chile. He received his Ph.D. in Computer Science from the Catholic University of Chile. His research interests include computer supported collaborative work, educational technology and software engineering. Dr. Ochoa is a member of IEEE, ACM and the Chilean Computer Society and sits on the Steering Committee of the LACCIR (Latin American and Caribbean Collaborative ITC Research Initiative). He currently serves as an IT consultant for a number of public and private organizations.

José A. Pino is a Full Professor of computer science at the Universidad de Chile. His research interests include computer supported collaborative work, human–computer interaction, and software industry studies. He has served as President of the Chilean Computer Science Society (SCCC) and President of CLEI (the Latin American Association of Universities Concerning Information Technology). He has co-authored six books and published research papers in international conferences and journals, including Journal of the ACM, Communications of the ACM, Decision Support Systems, Interacting with Computers, and Information Technology and People.

Rubén Darío Franco is an Associate Professor in the Business Administration Department and researcher of the Research Center on Production, Management and Engineering at the Technical University of Valencia. His research topics are in the area of software architecture with focus on digital business ecosystems and service engineering for Operations Management improvement. He has published several papers on the topic and has also co-authored and co-edited some books from international publishers.