Design and Implementation of IP Multimedia Subsystem Applications: An Enabler-Oriented Approach

Alberto Hernández, Pedro Capelastegui, Enrique Vázquez, and Francisco González,
Universidad Politécnica de Madrid
Antoine de Poorter, Ericsson España

ABSTRACT

Multiparty networked applications, which are gaining interest in areas as diverse as online games, e-learning, e-health, and collaborative work, tend to depend on a set of common functions. In conventional IP networks, these shared functions require application-specific servers, leading to scale diseconomies and complicating interoperability. The IP Multimedia Subsystem addresses this by incorporating basic services called service enablers, which can be used as building blocks to facilitate application development. This article discusses our experience creating innovative enablers for IMS networks, and combining them into new multimedia applications. An enabler-centric process is described, and the configuration of a development IMS network is explained, with suggestions on how to replicate this architecture with freely available components. Following these principles, we designed and implemented a series of novel group-centric enablers exploiting the advantages of group management, which we describe in detail. A use case made possible by these enablers is shown, consisting of an e-health scenario for remote medical visits. We reflect on the advantages and challenges introduced by this development methodology, and provide insight on how a project of this nature should be approached.

INTRODUCTION

Applications such as collaborative work, multiplayer networked games, e-learning, and e-health encourage communication among users in a variety of professional and entertainment spheres, both in fixed and mobile environments. These applications usually require a set of common functions such as user management (creation of records, storage of the user profile, creation of groups, etc.), information services (presence, willingness to participate in activities, etc.), and communications management (establishment of multiparty sessions, event notification, support for multiple devices, state management, quality of service in the network, etc.).

Should the network provide only IP connectivity (like the Internet), all the previous functions must be provided by application-specific servers. This situation usually leads to custom developments of the required functions, where different providers duplicate the same functionality. It also generates scale diseconomies and makes interoperability among applications difficult and, thus, among users.

The IP Multimedia Subsystem (IMS) [1] is defined by the Third Generation Partnership Project (3GPP) as a standard architecture for the provision of all-IP services in mobile networks and has been adopted by other standardization bodies for fixed telephony, cable, fiber optic, and wireless networks. The IMS architecture is based on the Session Initiation Protocol (SIP) to create and maintain communication sessions. It also includes a set of basic services (service enablers), including group management, presence, messaging, and so on.

Applications geared toward the IMS are deployed in the application server and usually developed as Java SIP Servlets [2]. Since the IMS is more feature-rich than a conventional SIP architecture, the application design and implementation cycle for this particular environment has to take into account these features to really make the most of IMS's possibilities.

This article of tutorial nature addresses the key topics specific to the design and implementation of IMS applications, introducing a set of reusable service enablers that follow a novel group-centric approach. This approach eases the development of multiparty applications by delegating certain functions to the IMS group manager and making the most of its notification system.

Since these best practices and approaches are the result of the experience designing and implementing the communication aspects of the novel e-health AmIVital platform [3], use cases are provided in the context of e-health scenarios (remote consultation, remote rehabilitation classes, etc.). However, there are no e-health

0163-6804/11/$25.00 © 2011 IEEE

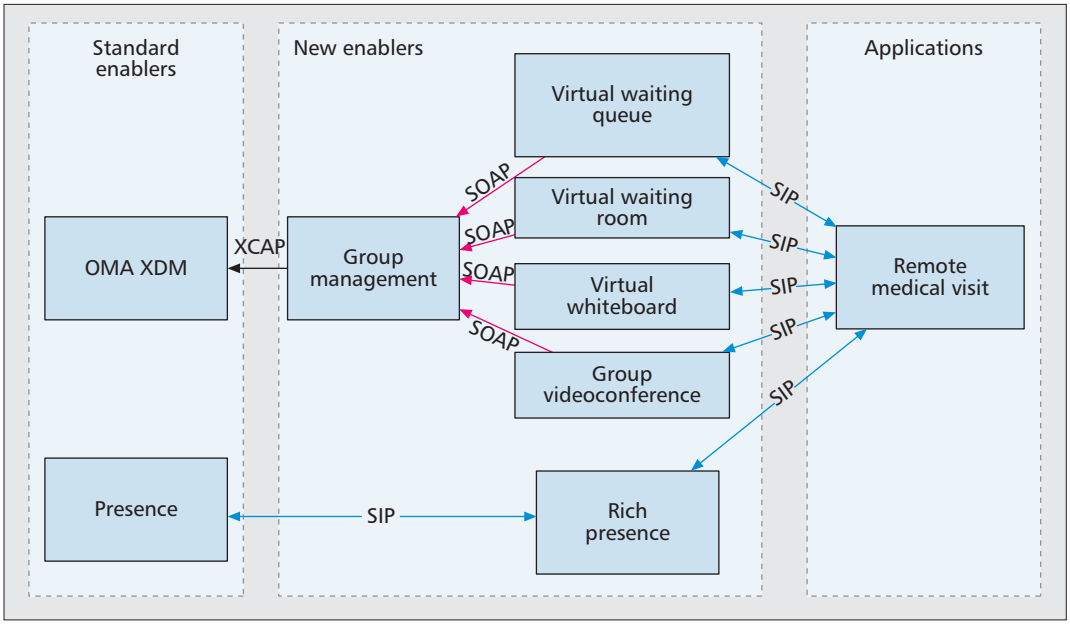

Figure 1. *Enabler set and dependencies.*

We followed an iterative development process, paying special attention to the requirement capture phase, and following with successive design and implementation steps, parallelizing work on different modules whenever possible.

specific enablers, so the ideas and decisions made are applicable to other fields.

The rest of this article is structured as follows. First, we introduce an enabler-centric development cycle we used to design and implement reusable network components. Then we discuss how to set up an IMS network for developing and testing applications, using freely available tools and without dedicated hardware. The next section describes a series of enablers we created following these principles, emphasizing the design choices behind them and the challenges they presented. The use case for an e-health application showcasing these enablers is also explained. Finally, we present a summary of the lessons learned and conclude.

Enabler-Oriented Development

In IMS, a service enabler is a reusable component that performs a specific network function. Enablers can be used as building blocks to build applications, and are typically defined as standards by institutions like the Open Mobile Alliance (OMA) and adopted by 3GPP as IMS standards. Common examples of service enablers include presence, location, and push-to-talk.

One of our research goals for the AmIVital project was to identify network functions that could be used in a wide enough variety of scenarios to merit the creation of new service enablers. We would then carry out the design and implementation of these enablers, and use them to compose a series of applications that would serve as proof of concept and testbed. Since AmIVital was developing a tele-assistance platform as its main prototype, our applications would be oriented toward e-health so that they could be integrated in this platform.

We followed an iterative development process, paying special attention to the requirement capture phase, and following with successive design and implementation steps, parallelizing work on different modules whenever possible.

Requirement capture started with brainstorming for innovative e-health related network applications. We discussed over a dozen potential applications that would fit in e-health scenarios. Then we analyzed the use cases for the applications in this list in order to extract common processes that could become enablers. These tentative enablers were evaluated according to applicability to different scenarios, relevance of their associated functionality, and cost of implementation. Based on these criteria, we came up with the set of enablers we wanted to develop. To conclude, we reexamined the brainstormed use cases and selected the ones that would showcase our new enablers.

At the end of this process, we had decided to design and implement the service enablers and network applications showed in Fig. 1 that will be explained throughout the article in more detail.

The next step was to provide functional description for all system components. Although each new enabler was considered a separate module, they could interact with each other, and a hierarchy of dependencies was established. It should be noted that some of these dependencies pointed to already existing, standardized components, as some of our new elements extended previous functionality, rather than introducing brand new features.

When defining interfaces for the different modules, we used a combination of SIP, web services, and web access. In many cases, the main functionalities were carried out through SIP-based interfaces, as could be expected in an IMS application, but some of the enablers also included web pages as a means to provide complementary information. In addition, web services interfaces were introduced for enabler-to-enabler communication, management, and testing.

In our case, we decided that our applications and enablers would be completely contained on the application servers, so we could use any standard-compliant SIP UA supporting audio and video sessions, instant messaging, and presence.

Regarding deployment, all modules were designed to be able to operate in separate application servers if needed, although in our environment they were collocated in a single machine. Either way, a point that should be given special attention is the definition of IMS service triggers and the potential interactions between enablers. It is not enough to consider signaling sequences for each module in isolation, since the probability of unintended interference grows with the number of enablers.

Testing was performed at different levels, and with a variety of tools. For unit tests we chose the JUnit (http://www.junit.org) framework, which we had positive previous experiences with. However, we had not used it in combination with Java servlets before, so we did not anticipate the challenges it would bring. Testing servlet-related classes with JUnit is complicated, as the tests run outside of the servlet container, so the creation of actual servlet instances is not possible. There is the alternative of configuring stubs for the servlet classes, but it is costly, and we found it was not worth the effort. In the end, we decided to decouple application logic as much as possible from network logic, so we could leave classes inheriting SipServlet and HttpServlet without automated unit tests, while still having test cases that covered a good share of the total code.

For the functional tests of each enabler, we needed tools that matched its multiple interfaces. For SIP signaling we went with a script-based approach, using software like SIPP (http://sipp.sourceforge.net) or the automated testing framework in Ericsson SDS. This was quick but restrictive, and rarely covered more than a basic success scenario and a few of the most common errors, so we complemented it with manual test runs against real user agents whenever possible. Web services interfaces did not take as much effort, since we used soapUI (http://www.soapui.org), which is well integrated in the IDE and even allows including web services calls in JUnit test cases.

As the test scenarios integrated more components, the growth in complex interactions over multiple interfaces led to a decrease in test automation. By the time the tests covered the whole system, most of the task was manual, and the examination of software traces and network captures from programs like Wireshark (http://www.wireshark.com) proved invaluable. This was a cumbersome process, but an automated alternative would have probably involved implementing a dedicated user agent for testing purposes, taking up even more time.

Network Environment

A crucial element in IMS application development is the network environment where the software will be tested. Too often a developer will not have access to a full featured commercial IMS network during the first stages of the application life cycle. Nevertheless, many IMS nodes are available through free implementations, so a developer can end up with a network architecture suitable for testing at no economic cost — although with significant configuration and integration effort. For our network environment, we prioritized the use of components that were available for free or could be provided by project partners. In this section, we describe the elements that composed our architecture, and suggest alternate free solutions that can be used on a development IMS network.

Figure 2 shows the IMS network we used for development, including the different IMS functions and the technologies that implemented them. The core IMS functionality (i.e., HSS, P-CSCF, and S-CSCF) is provided by Ericsson Service Development Studio (SDS), which is implemented as an extension for the Eclipse IDE and includes several developer-oriented features. SDS also provides integration with Sailfin, the open server we used as our IMS application server (AS). For these reasons, added to our previous experience with the software, we chose it over the open source alternative, Open IMS Core. Unfortunately, SDS support has since been discontinued by Ericsson, and the tool is no longer available to download. For some elements in our network, we were given permission to use proprietary solutions developed by Ericsson, which was one of the partners in AmIVital. These included an OMA XDM server [4], or XDMS, for group management purposes, and a media resource function (MRF) to handle multiparty conferences. Although the XDMS could have been replaced with an open source alternative such as Mobicents SIP Presence Service, we were unable to find a viable free implementation for the MRF. Since the absence of a free version of this server is a significant obstacle to the development of conferencing services over IMS, we believe that an open source project implementing an MRF could be of great benefit to the developer community.

The selection of a user agent (UA) implementation presented considerably more options than any other component of the network. An important factor to take into account here is the type of application that is going to be developed, and specifically whether or not it will require new client-side logic. In our case, we decided that our applications and enablers would be completely contained on the application servers, so we could use any standard-compliant SIP UA supporting audio and video sessions, instant messaging, and presence, such as Counterpath X-Lite. If we had developed an application requiring UA extensions, an open source implementation would have been preferable. We have had good experiences with Java-based SIP stack MjSIP, but it has not received any updates in the last few years; a more up-to-date alternative, also written in Java, is Minisip.

Table 1 summarizes the network elements described in this section.

Designing with the IMS in Mind, A Group-Centric Approach

This section introduces the e-health scenario we implemented for the AmIVital project using our enabler-oriented methodology. We describe the basic use case, the application enablers that were derived from it, and the integration of these enablers into a functional application.

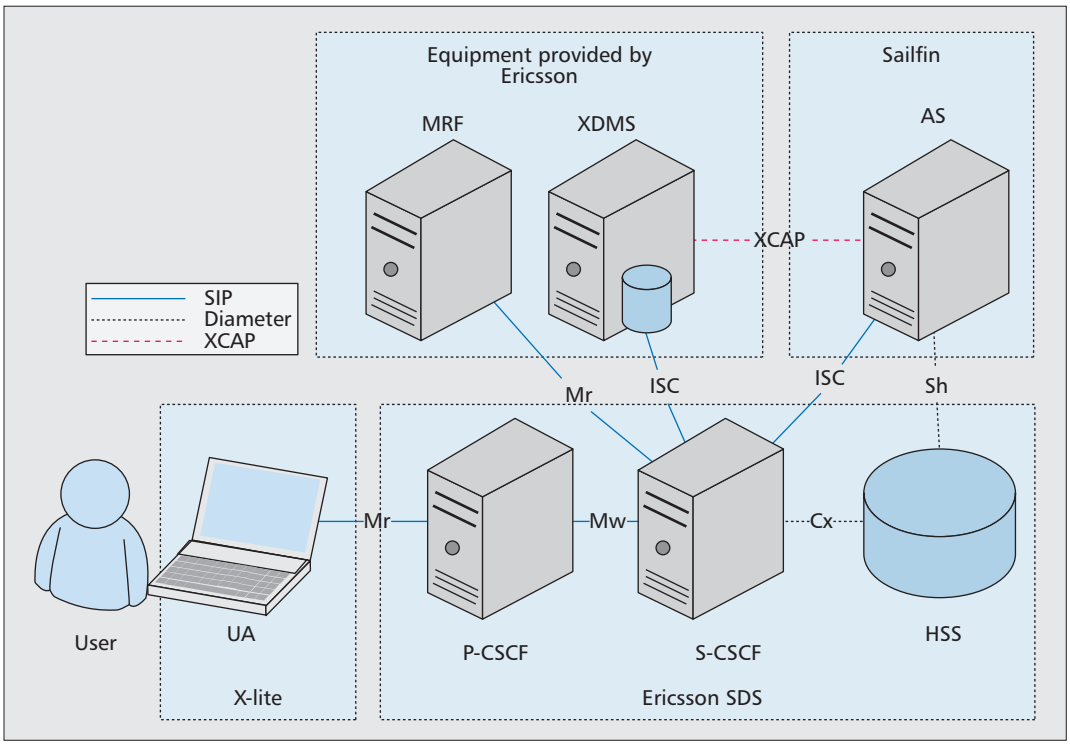

Figure 2. *IMS development network.*

We defined a use case for an e-health system that could be easily decomposed in a set of common functions which could then be implemented as reusable enablers. It consists of a remote medical visit, where patients can be examined without need of physically going to the doctor's premises.

E-Health Use Case

We defined a use case for an e-health system that could easily be decomposed in a set of common functions which could then be implemented as reusable enablers. It consists of a remote medical visit, where patients can be examined without need of physically going to the doctor's premises.

The application scenario is as follows. A patient has scheduled a virtual consultation with a given doctor. At the appointed time, the patient is queued until the doctor is ready. The state of the system queue can be queried at any time, and a content delivery service can provide waiting users with multimedia resources for entertainment or medical information as if they were in a virtual waiting room.

When the doctor is ready to see a patient, a videoconferencing session is started between both parties in order to perform the examination. Further participants such as medical specialists can be incorporated into the session later if needed. As part of the virtual visit, the doctor can invoke a virtual whiteboard service to share medical images such as radiographs with the patient, adding annotations to them in real time.

Once the visit is over, all associated sessions are terminated, freeing network resources, and the queue is notified to call the following user.

Figure 1 shows the common network functions that can be extracted from this use case. In addition to the enablers that add direct value to end users, we have identified some functions that are only accessed by other modules of the system, like the OMA XDM server, which stores group data used across different applications, and the group management enabler, which hides calls to that server under a web services interface. Although most of these enablers are new developments, a few are defined as standards, like OMA XDM and the presence enabler, which we extend with a rich presence enabler providing additional user information. A remote medical visit application aggregates this set of enablers to implement the use case.

Service Enablers

One of the key decisions made during the design phase is following a group-centric approach and exploiting the features of the notification system of IMS to allow communication among enablers. Traditionally, the IMS group manager is merely seen as an enhanced contact book to store phone numbers and SIP URIs of the contacts (resource lists [5]). However, our novel approach turns the standard group manager into a key element to ease the development and deployment of service enablers that deal with multiparty applications. The main idea behind this is that service enablers will offer the functionality to a group of users, identified by its unique group URI, and will use the group as a shared space to exchange information and allow management via standard IMS group management interfaces based on the XCAP protocol [6].

The group-centric concept is illustrated in Fig. 3 with the videoconference and whiteboard enablers as an example, although the same design philosophy is applicable to other multiparty enablers. Videoconferences are provided in the IMS by means of the MRF node [7]. This node acts as a central SIP signaling point for multiparty conferences and provides media functions to compose video mosaics, transcode video, and generate audio/video streams to each user in the conference. To create a videoconference, it is necessary to configure a conference URI in

Element	Solution	Developer	License	URL
IMS core	SDS*	Ericsson	Free for noncommercial use	—
	Open IMS	FOKUS	Open source	http://www.open-ims.org
AS	Sailfin*	Sun, Ericsson	Open source	https://sailfin.java.net
OMA XDM	Mobicents SIP Presence Service	Mobicents	Open source	http://www.mobicents.org/sipp-resence/intro.html
UA	X-Lite*	Counterpath	Free for noncommercial use	http://www.counterpath.com/
	MjSIP	University of Parma	Open source	http://www.mjsip.org/
	Minisip	KTH	Open source	http://www.minisip.org/

* Used in our development IMS architecture

Table 1. *IMS components available for developers.*

the MRF, and the participants must establish a session with that URI to receive the mixed audio/video data stream, just like in the current telephone service when we set up an audio meeting.

The group videoconference enabler automatizes this process and completely hides SIP signaling and MRF interfaces from the conference administrator. In fact, except for a web service (SOAP) call to the enabler to launch the videoconference, adding and removing users can be done with just a standard XCAP client, easing the introduction of videoconferences in other group-oriented enablers and applications like the virtual whiteboard or the virtual waiting room. Thanks to the notification system of the IMS group manager, the videoconference enabler subscribes to the group associated to the videoconference and updates the videoconference status according to group composition. This way, when setting up a group and launching the videoconference via the enabler's web services *startVideoconference* call — specifying the group identifier as a parameter — the enabler will query the group manager to obtain all the group's members and automatically refer them to the MRF conference URI previously set by the enabler. If a new user is added to the group, the enabler is notified by the group manager and referred to the MRF so he/she can take part in the conference. Note that SIP UAs tested in our use cases lack floor control capabilities as they mimic an actual medical visit. However, MRF's own floor control and video mixing control mechanisms may be used as needed.

The virtual whiteboard is a service enabler that allows applications to provide a shared space to the members of a group where they can write, annotate, paint and share pictures. Its main use in the remote consultation scenario is the display of radiographies, where the doctor is able to point and annotate certain parts of the picture. This enabler envisions different interaction methods depending on the capabilities of the user agent, taking advantage of the SIP session negotiation mechanism. When requesting a virtual whiteboard session by sending a SIP INVITE, inspection of the message's headers will determine the type of interaction available. More specifically, the supported interaction methods are the following:

Download and setup of custom application installed on user device. It requires specific support in the device but may provide the best integration with the device's user interface and input methods. The interaction is done in the context of SIP sessions, as there is a negotiated TCP data flow between each user and the enabler.

AJAX/applet interaction. When adding a user to the whiteboard, the enabler will send an RFC 4483 conformant SIP MESSAGE specifying a dynamic URL for the virtual whiteboard enabler instance. The user agent should automatically open that URL in the device's browser, where available.

Videoconference establishment. The enabler supports the interaction via standard video sessions, in a view-only fashion, as long as there is a component able to generate a video stream and update it when changes are made to the virtual whiteboard content. This eases the integration of the virtual whiteboard into videoconferences as it would take the advantage of the MRF video handling, composing, and encoding functions. However, this feature was not implemented due to limits in the Java Media Framework (http://java.sun.com/products/java-media/jmf/), such as software encoding and lack of H.264 support.

All the enablers benefit from the inherent advantages of IMS, like user authentication, media handling in the MRF node, group management, and messaging, among others. Moreover, as all of these enablers follow the same group-centric approach, it is possible to combine them and control their behavior through the standard XCAP-based group manager interface. For instance, if members of a group videoconference decide to start the virtual whiteboard service, the virtual whiteboard enabler will establish a session with each member of the group or send a SIP MESSAGE with a link to open with the browser, depending on the capabilities of the UA. An external controller (not necessarily

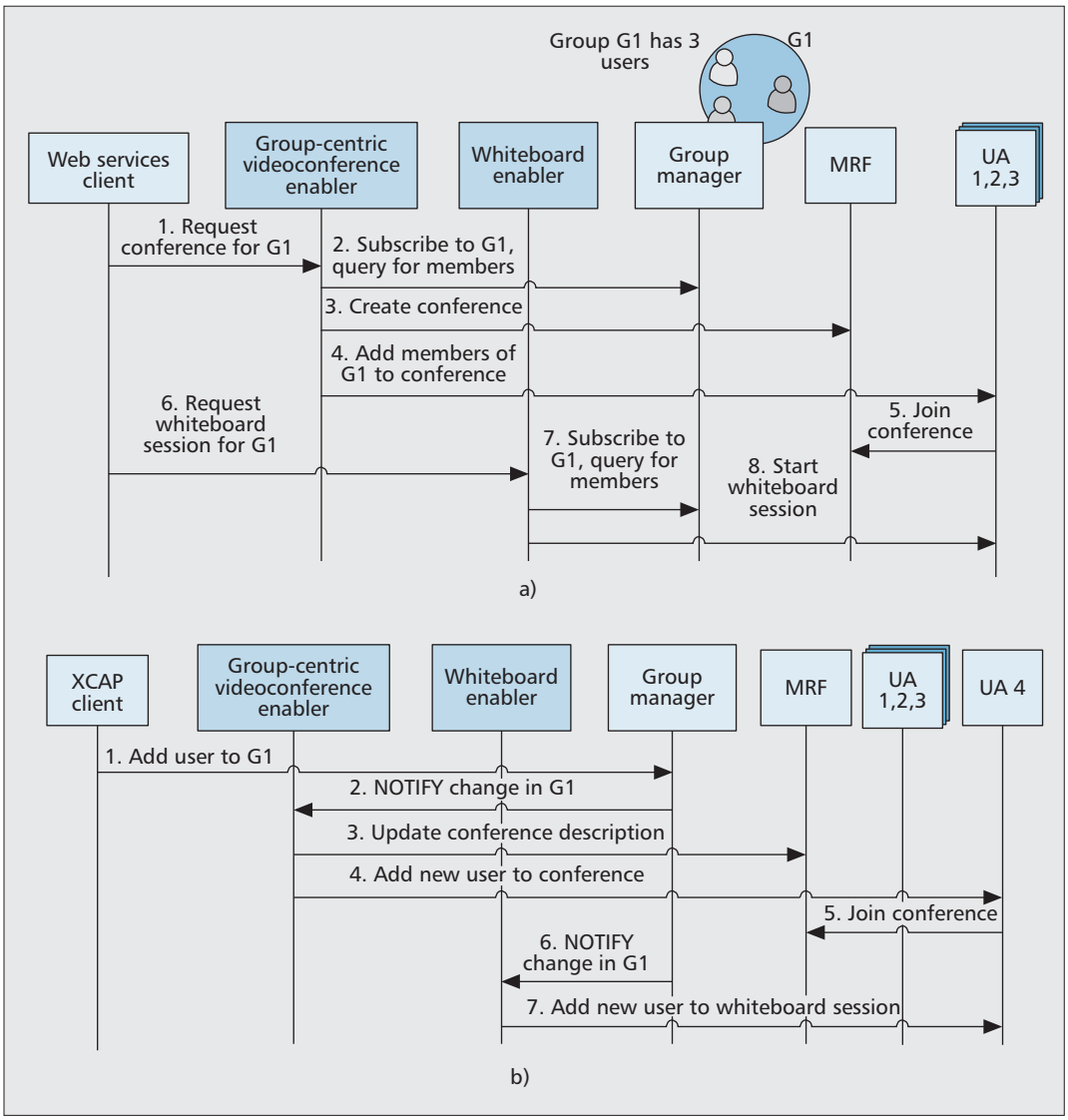

Figure 3. *Using groups across enablers for multiparty applications: a) establishing enabler sessions — videoconference and whiteboard enablers are activated for a common group; group data can be shared across enablers; b) when a new user is added to the group, all enablers used by the group are updated; the user is automatically incorporated into videoconference and whiteboard sessions.*

The virtual whiteboard is a service enabler that allows applications to provide a shared space to the members of a group where they can write, annotate, paint and share pictures. Its main use in the remote consultation scenario is the display of radiographies.

aware of the involved services) may add a member to the group; and automatically, the new member will take part in the videoconference and virtual whiteboard sharing as each enabler has received notification of a new member and has started the sessions needed to add the new participant. In a similar way, removing a user from the group will automatically tear down the sessions related to both videoconference and virtual whiteboard.[1]

Another advantage of the group-centric approach is scalability and reliability. The common technique to scale an application in the IMS is to increase or decrease the number of copies of the application server running each time. This way, each received SIP request is sent by the IMS core to one of those instances in a load-balancing fashion. In enablers where each enabler instance is tied to a user there is no problem following this approach, as each instance stores the data related to the user and the required functionality. In case there are more users using the enabler, creating more instances is enough as they are not related to each other. However, given that the IMS core is not aware of multiparty applications, it is possible that each SIP request of a group of users participating in an application is routed to different application server instances. For instance, the group-based videoconference enabler eases the establishment of multiparty conferences just by specifying a group identifier and calling a SOAP method called *startVideoconference*. The enabler may act as a third party call controller and hence needs to exchange SIP signaling with all the users in the group. Optionally, the enabler can also be configured to support dial-in schemes where users join the videoconference by means of an INVITE to a conference URI, and are automatically included in the associated group.

Instead of deploying custom databases to store data for each enabler, we have opted to use the same group XML document describing each group in the group manager. By storing

[1] As there are no SIP sessions involved if the interaction mode with the virtual whiteboard is set to AJAX/applet, it will just show a message to the users indicating that they are disconnected from the whiteboard share.

Given that XCAP communication with OMA XDM servers is a common feature shared by several of the enablers, we simplified their implementation by introducing a new abstraction layer that handles all XCAP signaling, and provides simple web services and Java interfaces.

key-value parameters on a group, we can model the state of an enabler regarding that specific group of users. For instance, in the videoconference service a parameter can indicate whether user Alice has been successfully added to the videoconferencing session. This group-centric approach improves scalability by avoiding the need to route all the messages belonging to a certain group of users to the same server instance. This way, server instances can be created and destroyed at any time, and users may be connected to one server or another seamlessly as the enabler global state is not stored in server instances.

Note that any group oriented enabler must download the group information and subscribe to its changes, so actually there is not much extra stress put on the XDM servers as long as the state information is reasonably small (a few key-values associated to each user in the group that will typically change when joining, starting or ending a multiparty application).

To ease the interoperability with other members of the project consortium, all the designed enablers comprise a set of IMS and web services interfaces. In addition, to keep compatibility with standard SIP/IMS user agents while providing enhanced interaction with the user, the enablers may expose a web (HTML) user interface that complement the user agent standard user interface. This is the case of the virtual whiteboard.

Given that XCAP communication with OMA XDM servers is a common feature shared by several of the enablers, we simplified their implementation by introducing a new abstraction layer that handles all XCAP signaling, and provides simple Web Services and Java interfaces. This new layer, which we call the Group Management enabler, also inserts additional metadata to the groups such as time of joining for each user, which is useful for other enablers.

There are other scenarios where IMS has proven to be very useful to build a platform of services, even relying on standard enablers. We can find an example in a self-monitoring scenario, where the home device (a set-top box connected to the user's television) reads vital metrics from a Bluetooth sensor. This process takes a few minutes and, if interrupted, has to be started again. In this case, a presence agent in the device will automatically set the state of the user to not available, and all calls except emergency ones will be rejected so the monitoring results are not altered.

Summary, Related Work, and Lessons Learned

This article introduces the advantages and challenges of developing innovative applications in an IMS environment. It complements earlier published work, offering a more tutorial approach based on the experience developing the IMS communication modules of the AmIVital e-health platform. More specifically, Moreno *et al.* [8] envision the application of the IMS to ambient assisted living environments, describing the possibilities and fields of interest from an e-health perspective, whereas Capelastegui *et al.* [9] is focused only on group based applications.

We have discussed an enabler-centric development cycle, and used it in the design and implementation of novel reusable network functions. We have described an IMS testbed scenario partially based on freely available elements. The key decisions behind the creation of a series of innovative enablers have been presented along with a use case for an e-health application showcasing them.

The first of the lessons learned is that we cannot think of IMS enablers as final services (group manager as contact list, presence service to show availability to contacts, or videoconference for a face-to-face meeting) as they can reduce development effort if integrated correctly in the applications running in the IMS. We have shown how a group-centric approach helps sharing information across different applications.

The second lesson we learned is that the range of free and open source tools available to IMS developers is enough to deploy and test applications for point-to-point scenarios, or multiparty applications lacking video. However, the absence of a free MRF means that rich multimedia multiuser applications require investing in commercial solutions, which may be difficult for small developers.

One of the challenges we came across was the integration of IMS applications in non-IMS environments. Web services interfaces were crucial here, exposing enabler functionality to third parties while hiding the inner protocols of the service platform.

The main advantage of an enabler-oriented design lies in the potential for module reuse, and in this regard our experiences have been mostly positive. For new applications in IMS environments, the adoption of an existing enabler can be relatively straightforward. As an example, we have worked on a different project dealing with immersive conferencing systems where we could benefit from the group-based data model, and the videoconference and whiteboard enablers from AmIVital, and the effort of migrating and configuring these modules was low. However, this reuse can reveal shortcomings in the initial requirement and feature definitions, and in our case we found that some enablers that had been designed to meet the needs of AmIVital needed some adaptation to fit in different scenarios.

This brings up a significant challenge associated with the use of enablers: the need for fixed interfaces limits the changes you can introduce in a module, yet the wide variety of scenarios that each enabler is expected to cover means that sometimes, vital requirements are missed in the original definition. We have had to deal with changes in interfaces halfway through development, which were costly and would have become prohibitive at later stages of the process. For this reason, we would recommend defining, for each enabler, sample usage scenarios in combination with at least two or three different applications, even if only one of them is to be implemented in the short term.

To conclude, we believe that designing innovative applications and proving the advantages of using IMS as a service platform are the main obstacles to market adoption of this technology. The principles discussed in this article can help promote the development of new advanced multi-

media applications for groups of users, which is the key to success in areas like e-health, e-learning, social networks, and online gaming, to name a few.

ACKNOWLEDGMENTS

This work is partially funded by the Spanish Industrial and Technological Development Centre (CDTI) under the CENIT Program (AmIVital Project).

REFERENCES

[1] G. Camarillo and M.A. Garcia-Martin, *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*, Wiley, Aug. 2004.
[2] Y. Cosmadopoulos and M. Kulkarni, "JSR 289: SIP Servlet v1.1," Aug. 2008; http://www.jcp.org/en/jsr/detail?id=289.
[3] CENIT AmIVital Project web site; http://amivital.ugr.es/index.php?lang=en.
[4] Open Mobile Alliance, "XML Document Management Architecture, V2.0," 2007.
[5] Open Mobile Alliance, "Resource List Server (RLS) XDM Specification V1.1," Nov. 2006.
[6] J. Rosenberg, "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)," May 2007.
[7] 3GPP, "Conferencing Using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3," TS 24.147.
[8] P. A. Moreno *et al.*, "Teleconsulting: A Medical Application based on IP Multimedia Subsystem Technology for Ambient Assisted Living," *Proc. 2nd Int'l. Symp. Applied Sciences in Biomedical and Communication Technologies*, ISABEL 2009, Bratislava, Slovak Republic, 24–27 Nov. 2009.
[9] P. Capelastegui *et al.*, "Group Management in Value-Added Services Over IMS Networks," *IEEE Latin America Trans.*, vol. 8, no. 2, Apr. 2010, pp. 120–26.

BIOGRAPHIES

ALBERTO HERNANDEZ (albertoh@dit.upm.es) holds Ph.D. and M.Sc. degrees in telecommunication engineering from Universidad Politécnica de Madrid (UPM), Spain. He is a research fellow of the Telematic Engineering Department at UPM. His current research interests include highly interactive multimedia multiparty applications (collaborative work, online gaming, etc.) as well as next-generation networks (NGN) and innovative services, both infrastructure-based and infrastructureless.

PEDRO CAPELASTEGUI (capelastegui@dit.upm.es) received an M.Sc. degree in telecommunications engineering from UPM in 2007, and is currently studying for his Ph.D. in telematic systems engineering. He works in the Telematic Engineering Department at UPM as a researcher. His research interests include IMS, service enablers, and immersive multimedia communications.

ENRIQUE VAZQUEZ (enrique.vazquez@upm.es) received his M.Sc. and Ph.D. degrees in telecommunication engineering from UPM in 1983 and 1987, respectively. Presently, he is a full professor in the Department of Telematic Engineering of UPM. He has worked in Spanish and European R&D projects on several areas of telecommunications and computer networks, including protocol performance evaluation, traffic engineering, network convergence, and quality of service in next generation networks.

FRANCISCO GONZALEZ VIDAL (francisco.gonzalezv@upm.es) received his M.Sc. and Ph.D. degrees in telecommunication engineering from UPM in 1973 and 1993, respectively. He worked in several engineering and management positions at Alcatel until 2002, in the switching and broadband access divisions. Currently, he is an associate professor in the Department of Telematic Engineering at UPM. He has worked in Spanish and European R&D projects on several areas of telecommunications networks, traffic engineering, network convergence, and quality of service in next generation networks.

ANTOINE DE POORTER (antoine.de.poorter@ericsson.com) holds a B.Sc. degree in electrical engineering from the Higher Technical School in Breda, Netherlands, but he has worked most of his professional life in the telecommunications area. He started working with analog, mobile systems, then the GSM system and later the UMTS. His main areas of competence are related to network user databases, mobile network security, mobile multimedia systems, and 3G applications. He currently works in the innovation unit at Ericsson R&D in Spain.

The principles discussed in this article can help promote the development of new advanced multimedia applications for groups of users, which is the key to success in areas like e-health, e-learning, social networks, and online gaming, to name a few.