Enhanced Communication Services through Context Integration

Romelia Plesa[1], Luigi Logrippo[2,1]

[1]School of Information Technology and Engineering, University of Ottawa, Canada
[2]Département d'informatique et ingénierie, Université du Québec en Outaouais, Canada
rplesa@site.uottawa.ca, luigi.logrippo@uqo.ca

Abstract. With the emergence of ubiquitous computing as the new trend in personal communication, there is also a growing need for services to be tailored to the user's specific needs and preferences. While most of today's telephony communication services are context free, new systems are emerging that take into consideration context information in order to manage the use of a phone or other communication services. We propose an architecture that combines presence and context information in the processing of a call and show how our approach can be used to build an enhanced image of the user's presence, one that is not device or application dependent.

1 Introduction

As devices that enable mobile communication are becoming more and more popular, there is an increasing need for the users to control and customize their communication, based on the context in which this takes place. Most of today's telephony communication services are *context free* [**1**], providing no context regarding the purpose or the circumstances of a phone call. However, there is sufficient reason to believe that the user would like to have services enabled by the availability of context information in order to manage the use of the phone or other communication services. Presence-aware communication is a new metaphor promising to fundamentally transform communication. The current presence-based solutions are tightly coupling an individual's presence with a device or application. Building an enhanced image of the user's presence, independent of device or application, can obviate the inadequacy of this model. This can be accomplished by introducing context awareness in communication, thus allowing communication to go beyond the limitation imposed by static rules and offering the users the possibility to customize the behavior of their devices.

This paper presents our approach to build what we call "the consolidated user presence". Our architecture is applicable to scenarios where users define complex policies on how their communication should be handled, based on their current context. Communication requests are handled according to these policies, thus offering a very high degree of customization. The rest of the paper is organized as follows: Section 2 defines "consolidated user presence". Section 3 presents our proposed architecture and our approach to represent policies. Conclusions are presented in Section 4.

2 Consolidated Presence Information

Presence is defined in [**2**] as the ability of a user to communicate with others. The notion of presence in our work is broader than the usual "on-line/off-line" indicators. We want it to include the physical location of the user (*at home*), call state (*ready, currently on a call*), the role of the user or his willingness to communicate (*available*, *in a meeting*). It also includes indicators that show if a user is logged into a network and whether he is active and the preferred medium for communication (*voice*, *IM*, *e-mail)*.

According to [**3**], context is information that characterizes the situation of an entity. Human related context includes information about the users (e.g. their habits) or the user's social environment (e.g. co-location with others). Context related to physical environment includes location, but also infrastructure and information about surrounding resources for computation and communication.

A combination of presence information and context information can be used to build an intelligent picture of a user's current situation, status and accessibility. We call this the "**consolidated presence information**" for the user. It represents aggregated information from various sources that results into a unified view of an individual's current status. Consolidated presence information means not only raw presence and context data, but also deduced information.

This allows users to dynamically set policies that govern the particulars of how they interact with each other. Introducing presence and context information provides ground for new services: context-based services (1), availability (2), notification (3) or personal addressing services (4). Following are examples of policies for each case:

- *All calls from my students will have announcement X played out. (1)*
- *Secretaries are not available to answer enquires during lunchtime. (2)*
- *Remind me of the 3 pm group meeting if I am not already in the meeting room. (3)*
- *If the call is from a person involved in project X, redirect it to the team leader. (4)*

The processing of these policies requires knowledge about the users and the environment in which their activities take place, as well as intelligent mechanisms for deriving knowledge from the raw information that is available to the system.

3 Proposed Solution

3.1 The architecture

In order to support policies as the ones presented above, the architectural model needs at least the following functional requirements: collection of context information using sensors as well as dissemination of context information, publishing of presence information from users and their devices; description of user policies and preferences; user preferences-based ubiquitous handling of communication

Fig. 1 presents the architecture. One of our goals it to make the architecture independent of the communication protocol (it can be SIP [**4**], H.323 [**5**] or other session protocol). Every message that arrives for a user is intercepted and dealt with in order to extract information contained inside the message (e.g. the identity of the caller). The **Context Information Server** updates, stores and distributes the context information. The **Policy Server** manages the user's policies. *Personal policies* allow users to establish preferences about how their calls should be handled. *Subscription/ Notification policies* allow users to project different presence to different persons. While there are a number of languages for specifying policies ([6], [7]), in Section 3.3 we present and justify our approach.

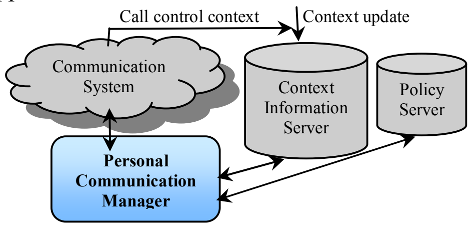

Fig. 1. The Architecture

A new functional entity is introduced, the **Personal Communication Manager (PCM),** which is a *software agent* that represents each user. PCM receives request messages (such as INVITE for a SIP-based architecture) and decides how they should be handled. PCM has three components: The **Presence Information Manager** uses a rule-based process to manage raw presence and context indicators and build the "consolidated presence information". The **Presence Directory** is a repository in which all known and deduced presence information is deposited. The **Policies and Preferences Manager** contains the preferences logic to respond to requests to contact an entity.

3.2 Presence Management Strategy

A user's consolidated presence is built from presence and context indicators that come from access networks, user's terminals or from third party information.

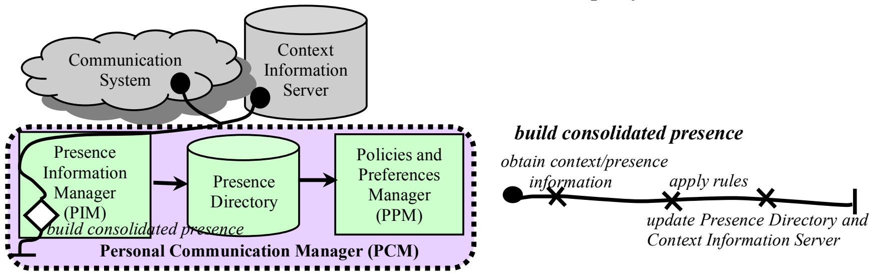
Communication System
Context Information Server
Presence Information Manager (PIM)
Presence Directory
Policies and Preferences Manager (PPM)
build consolidated presence
Personal Communication Manager (PCM)
build consolidated presence
obtain context/presence information
apply rules
update Presence Directory and Context Information Server

Fig. 2. (a) Consolidating presence (b) The "build consolidated presence" stub

This has to be done in real-time so that the user's presence can be projected before any attempt to communicate. PIM is aware of any notification of a change in the presence information. The context information is obtained by querying the Context Information Server. Having the presence information and the context information, PIM will apply the rules and will build and deposit the consolidated presence for the user into the Presence Directory. Some context information may change, so the Context Information Server must be updated as well. **Fig. 2**(a) uses Use Case Maps [8] to show this process. For simplicity, we used a stub to describe the actions of the PIM. The detailed sequence of responsibilities included in this stub is shown in **Fig. 2**(b).

The call model that we propose includes context update, service selection based on context information and user personal policies as well as service execution. Context update is a process that is done continuously. The service selection and execution mechanisms will be incorporated into the Personal Communication Manager, more precisely in the **Policies and Preferences Manager (PPM)** component of it.

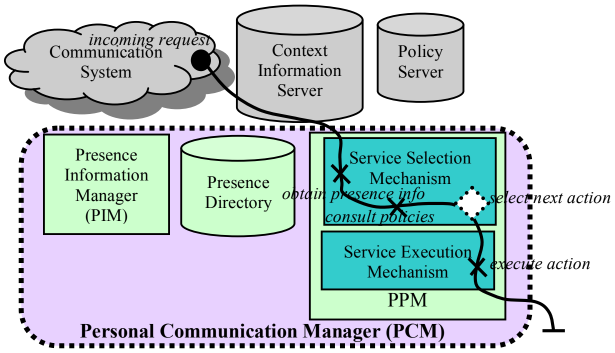
incoming request
Communication System
Context Information Server
Policy Server
Presence Information Manager (PIM)
Presence Directory
Service Selection Mechanism
obtain presence info
select next action
consult policies
Service Execution Mechanism
execute action
PPM
Personal Communication Manager (PCM)

Fig. 3. PPM's responsibilities

PPM consults the policies and the information stored in the Presence Directory and decides about the handling of any request. From the various alternatives available to it at a certain moment, PPM needs to select the appropriate actions to execute. We call this the ***Service Selection Mechanism.*** In **Fig. 3**, we use again Use Case Maps to show the actions of the PPM. The selection of the next action is a complex mechanism, which can be implemented in different ways (see the dynamic stub).

3.3 Representing Policies and Context

We chose to implement the PPM as a BDI agent [9] that conforms to the AgentSpeak(L) formalism, an abstract framework for programming BDI agents described in [10]. An AgentSpeak(L) agent consists of a *set of beliefs* and a *set of plans*. A plan *p* is written as: `p ::= te : ct <- h`, where *te* is the triggering event, *ct* is the plan's context, and *h* is a sequence of actions.

We decided to implement the PPM as a BDI agent that conforms to this formalism. Its beliefs will be the content of the Presence Directory. Policies (stored in the Policy Server) will be represented as plans. Following is an example of how beliefs and plans are represented for an agent that represents a user (Alice) and manages her communication. The partial AgentSpeak(L) code for the agent is given in below. We have annotated each plan with a label, so that we can refer to them in the text below.

Beliefs

```
        user(Alice); status(Alice, idle); colleague(Bob); lunch_time(11-12).
```

Plans

```
+invite(X, Alice) : lunch_time(t) <- !call_forward(Alice, X, Carla)          (p1)
+invite(X, Alice) : colleague(X)<- !call_forward_busy(Alice, X, Denise)      (p2)
+invite(X, Y): true <- connect(X,Y)                                          (p3)
+!call_forward(X, From, To) : invite(From, X)
                    <- +invite(From, To),-invite(From,X)                     (p4)
```

```
+!call_forvard_busy(X, From, To) : invite(From, X), status(X, not(status(X, idle)))
                         <- +invite(From, To), -invite(From,X)                    (p5)
```

The agent's initial beliefs are about the users in the system, their status and the relationships between them. Plan p1 implements the policy: *"during lunch time, forward calls to Carla"*. The triggering event is an incoming call, denoted by the event invite. If the event occurs, the context is checked to see if the time is within the interval defined by lunch_time. If this is true, then a subplan is triggered by the event call_forward. Plan p2 implements the policy *"when I am busy, calls from colleagues should be forwarded to Denise"*. This plan is used when the agent perceives that there is an incoming call and by checking the context establishes that the caller is a colleague. If the context holds, a new subgoal should be achieved, triggered by the event call_forward_busy. Plan p3 specifies the default action in case of an invitation. Note that connect is a basic action that the agent is capable to execute. Plan p4 is the subplan triggered when forwarding is necessary. The plan's context is the fact that there is an invitation for a call for the user. Plan p5 is similar, but the context when this plan is applicable is different.

We find that this formalism provides us with an elegant way of representing the policies as well as the beliefs (the context) of the system. For implementation, we use **Jason** [11], an interpreter for AgentSpeak(L).

4 Conclusions and Future Work

Technological evolution and user demands justify the introduction of new and more personalized services based on contextual information. We introduced the elements of an architecture that allows the definition of more enhanced services that take into consideration the user's current context. A prototype of such architecture is under construction. One aspect that needs future work is the representation of context and policies and the definition of the selection function.

The authors are grateful to the National Science and Engineering Research Council of Canada for partial support of this research.

References

1. Turner, K., Magill, E., Marples, D. – *Service Provision. Technologies for next generation communications*, Wiley Series in Communications Networking & Distributed Systems
2. Rosenberg, J. - *SIP Extensions for Presence*, Internet Draft, 2001
3. Schmidt, A., Beigl, M., Gellersen, H. - *There is More to Context than Location*, Computers &Graphics, vol. 23/6, Dec. 1999, pp. 893-902
4. Rosenberg, J., Schulzrinne H. - *SIP – The Session Initiation Protocol*, RFC 2543, 1999
5. H.323 Information Site - http://www.packetizer.com/voip/h323/
6. Lennox, J., Schulzrinne, H. - *Call Processing Language Framework and Requirements,* Internet Draft CPL-Framework-02.
7. Reiff-Marganiec, S., Turner, K. J. - *APPEL: The ACCENT project policy environment/language,* Technical Report CSM-161, University of Stirling, UK, 2004
8. Use Case Maps, www.usecasemaps.org
9. Rao, A. S., Georgeff, M.P.- *BDI Agents: From Theory to Practice*. In *Proceedings of the first International Conference on Multi-Agent Systems*, 1995.
10. Rao, A., - *AgentSpeak(L): BDI agents speak out in a logical computable language*, in Proceedings of the 7th Workshop on Modeling Autonomous Agents, 1996
11. Jason - http://jason.sourceforge.net