A Generic Layer Model for Context-Aware Communication Adaptation

Bassam El Saghir Noel Crespi
RS2M Department
GET / Institut National des Telecommunications
Paris, France
{bassam.el_saghir, noel.crespi} @int-edu.eu

Abstract— **Personal communications are facing many challenges created by mobility and convergence in today's communication networks. People often find themselves interacting with their devices in attention-constrained environments and deal with a bewildering variety of communication services, devices and access technologies. Many solutions tried to resolve this issue by relying on context-awareness. However, they suffered from drawbacks that hinder their deployment in the consumer market. In this paper, we discuss the implementation of INCA (Intelligent Network-based Communication Assistant), a multi-layered agent for context-aware adaptation of personal communications. This agent relies on a generic layer model that is applied to each of its layers, therefore enabling an intuitive and easy to implement architecture.**

I. INTRODUCTION

The concept of personal communications has much evolved since the early days of telephony. This evolution is driven by two main factors: mobility and convergence. Both factors provide undeniable advantages to the user. Mobility (mostly associated to the development of wireless communications) allowed users to be reached anywhere and anytime. Convergence, on the other hand, allowed users to access a wide variety services using any available device and/or network.

Still however, both factors have also their negative impacts. For example, mobility allowed users to be reached in inconvenient situations (e.g. work meetings) and attention-constrained environments (e.g. driving). On the other hand, convergence, which is occurring at three levels simultaneously (service, terminal and network levels [8]), creates confusion for the average user by forcing him to choose among a plethora of services (e.g. voice call, SMS, email, instant messaging), devices (e.g. fixed phones, mobile phones, laptop PCs, PDAs, smartphones) and networks (e.g. GSM, GPRS, UMTS, WLAN, xDSL).

Most of the solutions developed to address mobility and convergence issue relied on context awareness, which consists of using information about the user and his environment (such as user activity, location, device capabilities and network status) to assist him in handling his communications. The problem however is that these solutions focused more on the modeling and provisioning aspects of context information and did not sufficiently tackle the adaptation aspect, in other words the exploitation of this information to adapt communications according to communication constraints and user preferences.

We believe that tackling the adaptation issue is very important for building successful context-aware communication solutions, and that it involves studying three distinct albeit inter-related levels: session level where user communications are materialized as protocol-specific sessions for signaling and media exchange, communication level where user communications are integrated into a full fledged communication environment involving various concepts (including users, locations, devices and networks) and user level where user profiles, preferences and expectations are modeled to provide personalized services. Studying the relations between these levels is also very important in order to mirror, as closely as possible, the relations between these levels in the real world.

In this paper, we discuss the design and implementation of INCA, an Intelligent Network-based Communication Assistant that provides services for user communications (e.g. filtering, redirection, content adaptation and automatic initiation of sessions) based on context information about the user and his surrounding environment [4]. We start by defining a generic layer model (section II). This model is then used to build INCA's architecture by applying it successively to each of its layers (section III). Finally, we provide a brief discussion of related work in this field (section IV).

II. INCA'S GENERIC LAYER MODEL

A. Motivation for layered control

A key issue for achieving successful context-aware adaptation of communications is relying on a comprehensive model of the real world where communications are actually taking place. This model is gradually built and updated using context and communication signaling information and provides support for deciding on actions that affect this environment. We view the world model as being decomposed into three main levels: *session level* involving protocol-specific sessions for signaling and media exchange, *communication level* involving a communication environment representing various concepts (such as user identities, locations, devices and networks) and *user level* focusing on user-specific information such status, profiles and preferences. Therefore, the best way to deal with this decomposition is to develop a multi-layered architecture where each layer interacts with the world model at one of its levels.

Another reason for favoring the layered approach is performance. In fact, most communication standards adheres to strict real-time constraints aimed at improving user experience, particularly by minimizing session establishment time but also by reducing overhead time for mid-session changes (which may prove critical for real-time and interactive communications such as voice calls). Unfortunately, context-aware adaptation tends to significantly increase communication processing and signaling overhead, thus creating serious performance issues. In this context, adopting the layering approach optimizes overall performance by delegating simple tasks that require fast processing to lower layers, while leaving more complex and time consuming tasks to higher layers.

Therefore, as part of our efforts to develop an intuitive and easy to implement architecture over Jena [12], we designed and implemented a generic layer model that could be reused at each layer by changing only the object on which it operates. The next section introduces this generic model and describes the interaction between two consecutive layers implemented according to this model.

B. *Description of INCA's layer model*

Fig. 1 represents two consecutive layers modeled according to the generic layer model. Each of these layers keeps track of its corresponding object by maintaining an internal state model of it, and relies on the layer immediately below for receiving *events* and executing *actions*.

Both events and actions have impact on the internal state model of the object. An event received from the layer below is processed by the event module to produce an *assumption*, which represents the state that the corresponding object is assumed to have when the event was dispatched. This assumption is applied by updating the object's current state to the assumed state. Similarly, the execution of an action by the action module produces an *expectation*, which represents the state that the object is expected to have after being subjected to this action. This expectation is materialized by the addition of the expected state to the object state model.

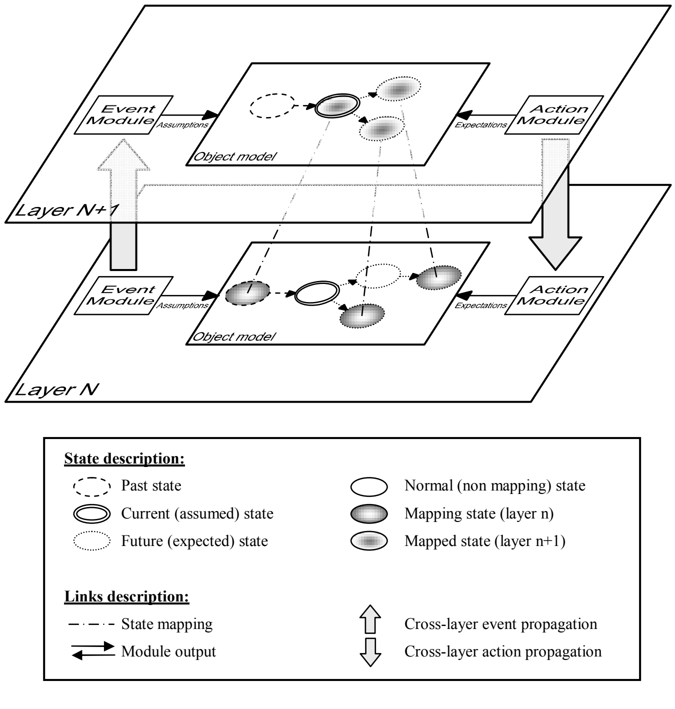


Figure 1. Generic layer model

When the object model in one layer evolves (as a result of a received event), it may reach a state that may be relevant to the evolution of the object model in the upper layer. This state is called a *mapping state* and is associated with a *mapped state* in the upper layer. In order to enforce this mapping between states, the event module of the lower layer detects the evolution of its corresponding object model to a mapping state and generates an event that is transmitted to the event module of the upper layer. The latter module translates this event to a mapped state and transmits it to its own object model, therefore causing this model to evolve to this new state. We note that the process of interpreting an event as a mapped state represents actually the assumption process.

When the object model in one layer needs to evolve to a particular goal state (e.g. as a result of a predefined rule in this layer), the action module of this layer infers the required action from this evolution and transmits it to the action module of the layer below. The latter module translates this action to a *plan*. A plan consists of a sequence of expected states and a list of associated actions where each action governs the transition from one expected state to another. It may also contain a sequence of events to be received in case it is correctly executed. A plan contains one or more intermediate states, but must always end with a mapping state in order to signal the completion of the plan to the upper layer. This latter layer then updates the current state of its object module by transforming its expected state into an assumed state.

III. Implementing INCA Layers

A. *Overview*

INCA comprises four layers ordered in increasing level of complexity: message layer, session layer[1], dimension layer and user layer. Communication between two consecutive layers follows the "interaction" concept which was already introduced in [4]. The next sections describe how each of these layers were implemented using our generic layer model.

B. *Message layer*

The message layer represents the bottom (and the most basic) layer in INCA's architecture. Unlike the upper layers, it does not implement any internal object model, but it provides the event and action modules that the upper layer (session layer) relies on.

This layer deals with message-level interactions, also called "reactive" interactions because they require simple yet fast processing. An example of such interaction is detecting

[1] Formerly known as operation layer in [4]

messages that require reception acknowledgment and dispatching the required responses. Event recognition at this layer consists of parsing incoming messages and extracting their headers and values. Reasoning consists of reactive rules that analyze these values in order to determine header values for new messages to send (if any). Action execution consists of structuring these values into messages which are then dispatched.

C. *Session layer*

1) Overview

This layer is responsible for managing communication sessions. It maintains information about session states for devices communicating with it. At the current stage of implementation, it natively supports the Session Initiation Protocol or SIP [15] (a SIP stack was especially integrated for this purpose). However, it may indirectly support other protocols by interfacing with special gateways via SIP (as demonstrated for SMS [1] [2] and ISDN [3] [14]).

2) Object model

For SIP based sessions, the *object state* maintained by this layer is normally the dialog state of the session at the remote user agent. However, we designed the dialog FSM (Finite State Machine) so that it can simultaneously represent the state at both the local and the remote user agents, therefore eliminating the need to maintain two separate dialog states per session. Consequently, for sessions directly involving INCA with another party, one dialog state is sufficient for tracking both dialog states at the local and remote user agents. Similarly, when INCA is acting as a back-to-back user agent (B2BUA) for a session between two different parties [15], it maintains two dialog states (one for each of the user agent client/server pairs) instead of the normal four.

3) Events and actions

Events received from the message layer consist of messages parsed as header/value pairs. Each received message is processed by the event module in order to compute the new current state that the session's dialog at the other party is assumed to have when it sent that message. In Fig. 2(a) for example, the reception of a "200 OK" response triggers the *assumption* that the dialog at the other party has already moved to the "confirmed" state. Therefore, the internal dialog state is updated to this *assumed state*.

Actions executed by the action module of the session layer consist of header/value pairs that are transmitted to the message layer. Before transmission of each action, the action module computes the state that the session's dialog at the other party is expected to have after receiving the message. In Fig. 2(a) for example, the transmission of an "INVITE" request triggers the *expectation* that the dialog at the other party should move to the "initiated" state after receiving this request. Therefore, this *expected state* is added to the internal dialog state model.

4) Interaction with the dimension layer

An event is generated from the session layer to the dimension layer when the dialog of a session reaches a *mapping state*. For SIP sessions, we defined three mapping states: "Initiated", "Established" and "Terminated". Events generated by these states respectively indicate that session establishment is in progress, session is already set up and session has ended.

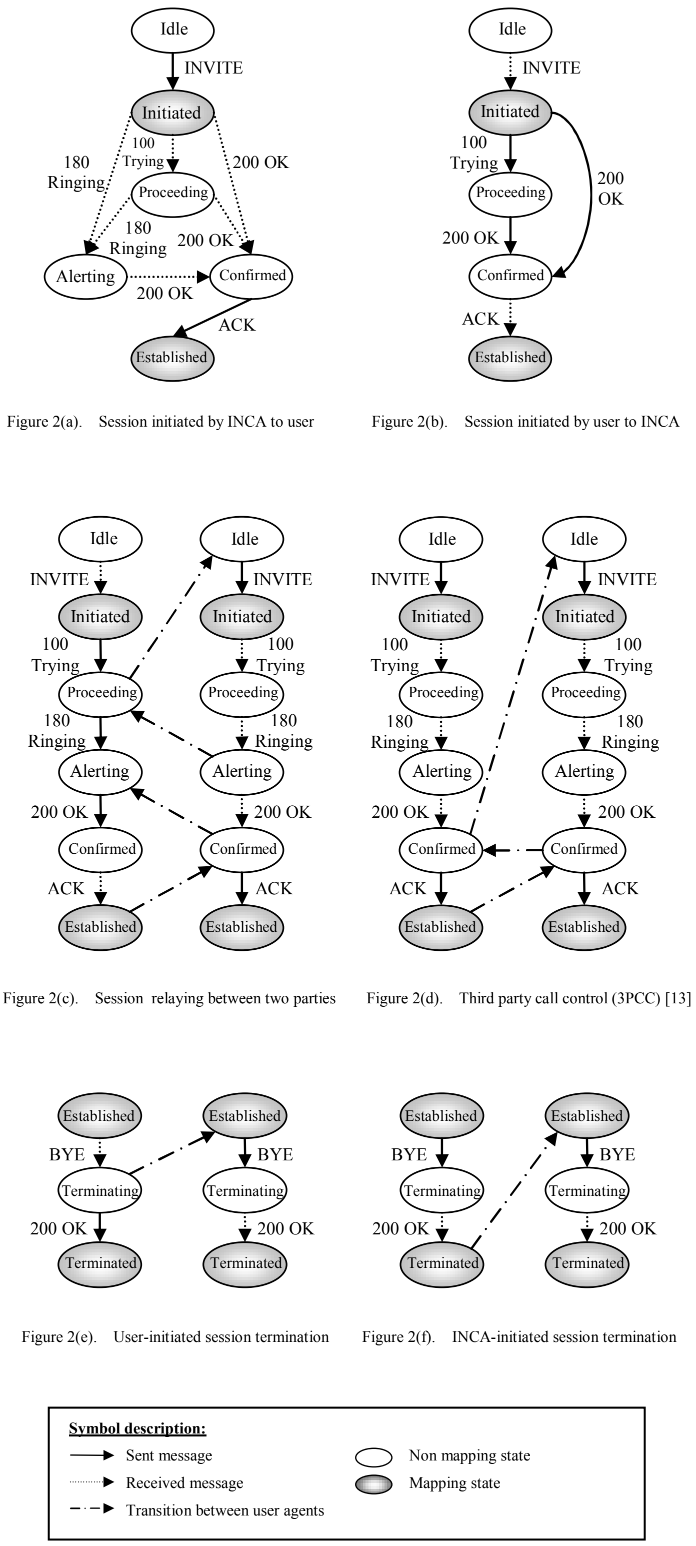


Figure 2(a). Session initiated by INCA to user

Figure 2(b). Session initiated by user to INCA

Figure 2(c). Session relaying between two parties

Figure 2(d). Third party call control (3PCC) [13]

Figure 2(e). User-initiated session termination

Figure 2(f). INCA-initiated session termination

Figure 2. State diagram representations of SIP sessions for six scenarios

Each action received from the dimension layer is analyzed by the action module of the session layer in order to retrieve the corresponding *plan* for executing it. For SIP based sessions, a plan consists of a sequence of dialog states and the corresponding sequence of sent messages (actions) and received messages (events) needed for realizing it (Fig. 2). Therefore, when an action is received from the dimension layer, the dialog states of the corresponding plan are successively added to the session model as expected states. At the same time, the associated actions are successively executed and appropriate events received, resulting in the progressive transformation of the expected states into assumed states. We note that if a particular plan fails (e.g. because of timeout or because of the reception of an unexpected message), a backup plan is executed and the dimension layer is notified accordingly.

D. Dimension layer

1) Overview

This layer aims at integrating information about different sessions as well as context into one single model known as the communication environment model. This model relies on two concepts, *dimension* and *communication*, which will be first introduced in the following sections.

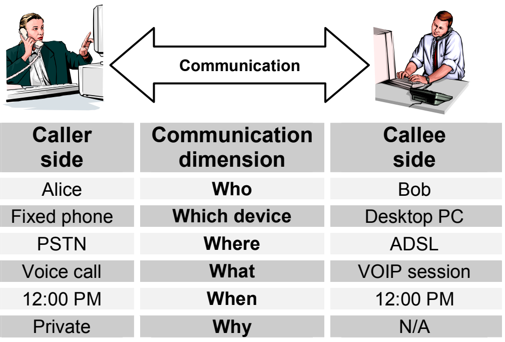
2) The dimension concept

A dimension is defined as a variable that describes a particular aspect of the user's world, such as user identity, location, device… Each dimension has its own distinct list of possible values that cannot be shared by any other dimension. Two types of dimensions could be distinguished: communication dimensions and situation dimensions. Communication dimensions are used to characterize a communication (see section III.D.3) and comprises six dimensions (also known as the six W's of communication): *Who* (involved persons), *Which Device* (communication devices), *Where* (network), *What* (content type), *When* (time parameters) and *Why* (subject or purpose of communication). In contrast, situation dimensions do not directly describe a communication but identify factors that may influence a communication such as location and connection bandwidth.

3) The communication concept

A *communication* is defined as an abstract relation between two or more parties involving a unidirectional or bidirectional information exchange over a communication network. It is identified by two sets of values of communication dimensions: one describing the caller and the other describing the callee (Fig. 3). Since we are focusing primarily on communication adaptation on the callee's side, the communication dimensions mentioned in this paper refer by default to the callee, and all caller dimensions will be ignored with exception of two: the caller identity (*Who*) which we will refer to as "Caller" for disambiguation, and the session subject (*Why*) which is applicable only to the caller.

Based on this definition, we consider a *session* as being a protocol-specific implementation of a communication. We also note that, unlike communication dimensions, the parameters that characterize a session are specific to the session type (e.g. phone numbers for voice calls, email addresses for emails …), and that each parameter could be mapped to one or more communication dimensions. For example, a fixed phone number is mapped simultaneously to a *Who* value (e.g. "Bob") and a *Which Device* value (e.g. "fixed phone").



Caller side	Communication dimension	Callee side
Alice	**Who**	Bob
Fixed phone	**Which device**	Desktop PC
PSTN	**Where**	ADSL
Voice call	**What**	VOIP session
12:00 PM	**When**	12:00 PM
Private	**Why**	N/A

Figure 3. An example representation of a communication

4) Object model

The *object* maintained by this layer is the environment where communications are taking place. This environment is described by the set of all possible values that dimensions may have. The *state* of this environment at a given time is determined by two factors: established communications and context. Established communications are represented by temporary bindings between dimension values that describe these communications. For example, a communication involving Bob's desktop PC would be represented by links between the values "Bob" (from the *Who* dimension) and "Bob's desktop PC" (from the *Which device* dimension) and a temporary dimension value representing the communication.

As for *context*, however, we chose not to represent it by specific dimensions (such as location or device) but by *relations* between dimensions (e.g. a device located in a particular location or a user owning a specific device). In our communication model, context is materialized by *links* between dimension values. The graph resulting from these links is known as the *dimension diagram*.

Fig. 4 shows a part of this dimension diagram. It includes four communication dimensions (*Who*, *Which Device*, *Where* and *What*) and two situation dimensions (*Location* and *Bandwidth*). A link between two communication dimensions implies that, for any two values chosen from these dimensions, these values cannot be parameters of the same communication unless there is an actual link between them. A link between two communication dimensions and a situation dimension implies that, for any two values chosen from the communication dimensions, these values cannot be parameters of the same communication unless they are linked to the same value in the situation dimension.

Information about user and environment context (such as user presence, device profiles and locations, network capabilities…) are received by context modules (Fig. 4) mainly through subscriptions to context information servers (e.g. Presence server [7], terminal capabilities server [5] and location server). These modules interpret the received information as changes in the appropriate links and integrate them in the dimension diagram.

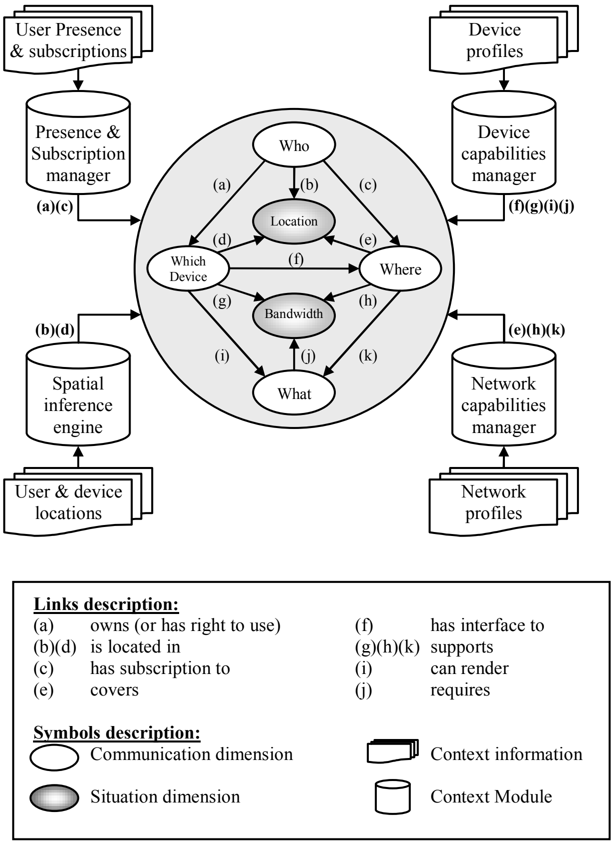


Figure 4. Dimension layer diagram

5) *Events and actions*

The event module of the dimension layer receives *events* about session state changes from the session layer (see section III.C.4)). For each event it receives, it first maps the concerned session to a communication by converting the session parameters into communication dimension values using information about user subscriptions. Then, it analyzes the new state of the session (mapping state). In SIP based sessions for example, if the mapping state is an "Initiated" state, the event module of the dimension layer adds an *expected state* to its object model indicating that the corresponding communication will be established. If the mapping state is an "Established" state, the latter state becomes an assumed state (i.e. the communication is already established). If the mapping state is a "Terminated" state, the communication is assumed to be terminated and the temporary links between the corresponding dimension values are torn down.

Actions at this layer consist of control instructions aimed at starting, relaying, modifying and terminating communications. Examples of control instructions for a communication with a single party or between two parties were already given in Fig. 2. For each action aimed at a particular communication, the action module converts the dimension values of the concerned communication into session parameters before relaying the action to the session layer.

6) *Interaction with the dimension layer*

An event is generated from the dimension layer to the user layer when the dialog of a session reaches a *mapping state*. Unlike the session layer however, the mapping states for the dimension layer are not predefined, but depend on the user preferences at the user layer. In fact, the condition part of each preference in the user layer is used to compute the links of interest of this preference in the dimension layer diagram. When changes in the diagram involve these particular links, they create a mapping state that, in turn, triggers an event which is propagated by the event module of the dimension layer to the user layer.

E. *User layer*

1) *Overview*

This topmost layer in INCA hierarchy is mainly responsible for implementing user preferences. These preferences mainly consist of monitoring the communication environment for relevant changes (in context as well as communications) and triggering actions that manipulate communications such as forwarding, modification of parameters, termination and initiation of new communications. The next sections describe how user preferences are implemented according to the generic layer model.

2) *Object model*

Each preference in the user layer is a combination of two parts: condition part and action part. The condition part is associated to a *condition verification state* that, when assumed, indicate that the corresponding conditions are met. The action part is associated to an *action execution state* that, when assumed, indicates that the corresponding actions were correctly executed. Preferences may share the same condition part (meaning that they are triggered by the same events, e.g. preferences (b) and (c) in Fig. 5), or they may share the same action part (meaning that they trigger the same actions, e.g. preferences (c) and (d) in Fig. 5).

3) *Events and actions*

The condition part of each preference specifies the links in the dimension layer diagram that must be monitored in order to trigger this preference. When these links reach the desired

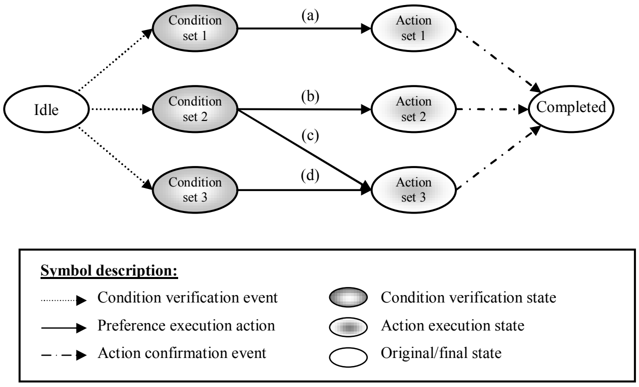


Figure 5. User layer preference model

states, the event module of the dimension layer detects a mapping state and transmits a *condition verification event* to the event module of the user layer (see section III.D.6). This latter module matches the event to the relevant preference(s), causing the corresponding condition verification state to become an *assumed state*. Then, the corresponding action parts are executed by the action module, causing the corresponding action execution state to become an *expected state*. Once the user layer receives event confirming the completion of these actions, this latter state becomes an assumed state, indicating that the corresponding preferences were correctly applied.

IV. Related Work

Many context-aware systems have been developed to demonstrate the benefits of context-aware computing. One of the earliest systems of this kind was the Active Badge Location System [17] which provides phone call redirection based on the location of the called person. Although it was developed more than fifteen years ago, this solution, and later context-aware solutions, did not make their way through from research labs to the consumers market. This is especially true for the personal communications domain, in spite of the readily available wireless network infrastructures.

We believe that the main reason behind this problem is that most of the previous solutions focused mainly on developing infrastructures and mechanisms for context provisioning (i.e. acquisition, modeling and dissemination) and did not sufficiently consider the next and more crucial step: context-based adaptation. For example, the Context Broker Architecture (CoBrA) [11] was primarily developed for supporting context-aware systems by addressing issues such as context modeling and reasoning, knowledge acquisition from different context sources and user privacy protection. The Service-Oriented Context-Aware Middleware (SOCAM) proposed in [10] aimed at supporting acquisition, discovery, and interpretation of context information to build context-aware services.

Even the few existing work that deals with communication adaptation (such as [6] and [16]) did not propose any detailed adaptation mechanisms. The only significant work in this area is the personal assistant proposed by Cisco [9]. Its main drawback, however, is that adaptation rules are very limited and should be explicitly specified by the callee.

V. Conclusion and Future Work

In this paper, we described the implementation of INCA (Intelligent Network-based Communication Assistant). We first introduced a generic layer model that could be reused across INCA layers, therefore enabling a well-structured and easy to implement architecture. Then, for each layer, we described how this model is implemented and how interaction with adjacent upper and lower layers takes place. Future work will mainly focus on extending INCA's compatibility to include session protocols other than SIP. Performance and conflict management issues at each of INCA layers will also be studied.

References

[1] 3GPP TS 23.204, "Support of Short Message Service (SMS) over generic 3GPP Internet Protocol (IP) access; Stage 2 (Release 7)," v7.3.0, June 2007.

[2] 3GPP TS 24.341, "Support of SMS over IP networks; Stage 3 (Release 7)," v7.1.0, June 2007.

[3] 3GPP TS 29.163, "Interworking between the IP Multimedia (IM) Core Network (CN) subsystem and Circuit Switched (CS) networks (Release 7)," v7.7.0, June 2007.

[4] B. El Saghir, N. Crespi, "An Intelligent Assistant for Context-Aware Adaptation of Personal Communications", IEEE Wireless Communications and Networking Conference (WCNC), March 2007.

[5] B. El Saghir, N. Crespi, "A New Framework for Indicating Terminal Capabilities in the IP Multimedia Subsystem", IEEE GLOBECOM-ISET, November 2006.

[6] W. Li, F. Kilander and C. G. Jansson, "Toward a Person-Centric Context Aware System," Workshop on Requirements and Solutions for Pervasive Software Infrastructures, May 2006.

[7] 3GPP TS 22.141, "Presence service; Stage 1 (Release 7)," v7.0.0, December 2005.

[8] 3G Americas, "Convergence: An Outlook on Device, Service, Network and Technology Trends," July 2005.

[9] Cisco Systems, "Cisco Personal Assistant 1.4," Datasheet, 2005.

[10] T.Gu, H. K. Pung and D. Q. Zhang, "A service-oriented middleware for building context-aware services," Journal of Network and Computer Applications, Vol. 28, Issue 1, pp. 1-18, January 2005.

[11] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware Systems," Ph.D. Dissertation, University of Maryland, December 2004.

[12] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne and A. Wilkinson, "Jena: implementing the semantic web recommendations," Proceedings of the 13th international World Wide Web conference, alternate track papers and posters, New York, USA, May 2004.

[13] J. Rosenberg, J. Peterson, H.Schulzrinne, G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", RFC 3725, April 2004.

[14] G. Camarillo, A. B. Roach, J. Peterson and L. Ong, "Integrated Services Digital Network (ISDN) User Part (ISUP) to Session Initiation Protocol (SIP) Mapping," RFC 3398, November 2002.

[15] J. Rosenberg et al., "SIP: Session Initiation Protocol," RFC 3261, June 2002.

[16] A. Schmidt, A. Specker, G. Partsch, M. Weber and S. Hoeck, "An agent-based telecooperation framework," In Proceedings of CoBuild'98, Darmstadt, Germany, February 1998.

[17] R.Want, A.Hopper, V. Falcao and J Gibbons, "The active badge location system," ACM Transactions on Information Systems, Vol. 10, Issue 1, pp. 91-102, January 1992.