CONCEPTUAL FOUNDATIONS OF USER PREFERENCE MODELING

Zohair Chentouf, Ahmed Khoumsi, and Soumaya Cherkaoui
Department of Electrical and Computer Engineering
University of Sherbrooke
CANADA

Abstract: Emerging telecommunications architectures, such as IP telephony, offer more signaling flexibility, elaborated feature creation techniques and user preference expression possibilities. The present article focuses on the last point. At our knowledge, no work that presents a thorough analysis of the user preference concept has been reported in the literature. The present article aims at inciting interest to this research axis. It specifies a *user preference conceptual model* to determine the different concepts that may be subject to user preferences. This conceptual model allows us to categorize the *preference expression schemas* that are possible to perform on the different conceptual model elements, and specify the architectural and language requirements necessary to implement those preference expression schemas. Examples of new services obtained by the user preference growing expression possibilities are then given. The benefits of the proposed user preference conceptual model are to direct the research work in this area and to provide a theoretical tool to analyze the expressive power of user preference modeling languages.

Key words: user preferences, user preference conceptual model, service creation, emerging telecommunications architectures, IP telephony, SIP.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35703-4_21](https://doi.org/10.1007/978-0-387-35703-4_21)

D. Gaïti et al. (eds.), *Network Control and Engineering for QoS, Security and Mobility II*

© IFIP International Federation for Information Processing 2003

1. INTRODUCTION

Service creation is one of the main activities in the deployment of any telecommunication service. With the growing popularity of the Internet, IP telephony operators are seeking for more elaborated service creation technologies. So are the operators of emerging architectures, through standards like Parlay [1], 3GPP [2], and JAIN [3]. Service creation in these architectures is growing in maturity.

One of the main preferred qualities of telecommunications services is to handle personalization. That is why the effort to sophisticate service creation has led early to looking for more elaborated ways to offer user preference expression tools.

The potential of service creation in emerging architectures promises to increase service number, variety and complexity. The present article explores this idea. It does not assume a particular environment even if we sometimes refer to SIP [4] for clarity purposes. SIP protocol mention is only motivated by the authors experience, since the present work is a part of a project that aims at implementing a Feature Interactions solution [5-7] in a SIP environment. The experimental framework we are working on is NIST-SIP provided by the National Institute of Standards and Technology [8].

In this paper, a conceptual model is specified to determine the concepts that underlie the user preference modeling. This conceptual model allows us to precise the user preference concept, categorize the different ways a user preference can be expressed through the conceptual model elements, and specify the architectural and preference language requirements necessary to put into practice the preference expression possibilities. Examples of new features that result from preference expression performed using the conceptual model are then given.

Section 2 defines some of the vocabulary we will use in this article. The user preference conceptual model and personalization schemas are specified in Section 3. Section 4 determines the requirements that should be fulfilled when implementing the user preference conceptual model. Section 5 presents related work on user preference modeling. Examples of personalized services are given in Section 6. Finally, Section 7 concludes the article.

2. DEFINITIONS

Some concepts that will be used throughout this article need to be clarified:

a) Feature, service

The ITU-T Q.1211 recommendation [9] considers services as what IN vendors provide to customers, and features as the elementary services that are used as building blocks to elaborate services. In this article, we adopt these definitions and add the following: in appropriate architectures, services may be elaborated by end users too. For example, Call Forwarding is a feature. It simply forwards calls to a given address. The Call Forwarding service, however, may be built on the feature by adding the possibility for the subscriber to activate or deactivate the feature and to specify the address to which calls will be forwarded. In appropriate architectures, such as IP telephony, non telephony applications (web page, email, etc.) may be combined with telephony features. We will then consider these applications as features and their program logic as service logic.

b) User preferences

User preferences are user intentions about service behavior that are expressed in an appropriate manner. For example, the Call Forwarding service offers the possibility to enter the forward-to address.

c) Personalization

Personalization is the operation of expressing user preferences. The expression means are associated to an execution environment that applies the expressed user preferences to the corresponding service behavior.

3. USER PREFERENCE CONCEPTUAL MODEL

In the IN (Intelligent Network) model, for example, service creation is done by the service vendor which may also be the network operator. As stated in Section 2, a service may be designed by combining several features (building blocks). The features of the deployed services in IN, only offer the user the possibility to enter a value of one or more *feature parameters* that belong to the service logic. In the case of the Call Forwarding service, for example, the only available parameter is the forward-to address. In the IN model, feature parameters are fixed and users can not add other ones.

In the IP telephony model, for example, services are designed by vendors but also by end users. The latter may also have the possibility to combine features to elaborate more complex services. This is a personalization possibility. Another one is to specify feature parameter values as in the IN model. For email for example, which is a non telephony application, the address where to send the message is the feature parameter. A third personalization possibility is to specify *preference parameters* and their values. Preference parameters are used by users in order to control feature invocation. They need a preference expression language that permits to specify them.

These are the three possible user preference expression ways:

- a) Updating a feature parameter.
- b) Specifying a preference parameter and value.
- c) Combining features.

In order to further clarify these three personalization possibilities, let us take the example of SIP architecture. SIP offers a set of built-in features: *call setup*, *call redirection*, *user location*, *call hold*, *call transfer*, *third party call control*, etc. Since SIP is situated at application level, these features are accessible from user programs. Therefore, users can write service programs that combine features with each other, determine feature parameter values, and/or specify preference parameters and determine their values. For example, Call Forwarding may be designed in SIP architecture using the SIP built-in feature that is *call redirect*. The latter simply redirects the call request towards the forward-to address. So, personalization of this feature, which results in user created service, may be done as follows:

- a) users have to determine a value for the *feature parameter* that is the forward-to address,
- b) they may add some *preference parameters*, such as time of day, callee availability, etc. and determine their values that trigger the call redirect feature: at what time incoming calls shall be forwarded, on which callee device state: a specified number of rings, no response, busy, ...?
- c) they may also combine this feature with email feature, for example, in order to send an email to the caller when callee is busy.

One suitable representation formalism for user preferences is the *condition – action rule* schema. It is the same formalism adopted by CPL (Call Processing Language) [10, 11], the language commonly used to express user preferences. In the remaining of the present paper, we will use

this schema. Each rule will be considered to express a *preference rule*. A user may have several preference rules written in some language. A preference rule may be considered as a tailored user designed service. A set of preference rules may be considered as a more complex service or as an elaborated service kit.

Since the beginning of the present section, many concepts have been mentioned. They all relate to user preferences. We now need to precise them into a well-defined user preference conceptual model.

Feature and preference parameters are information that belong either to the signaling flow or to network or external data sources. For example, call destination address, response content, etc., are extracted from the signaling flow or obtained from some network resources. This information may be used as feature or preference parameters. Information that is extracted from external sources, such as statistics logged by the user device using a personal program, can also be a feature or preference parameter.

A telecommunications network can be seen as a big machine whose components produce, process and exchange events. The user preference conceptual model we introduce builds on this general concept of *event* in the following manner:

- a) An *action* may be performed on an event to:
 - issue the event, or
 - update some of the event attributes.
- b) An action may be performed on a *device* to update some of its attributes.
- c) An action is performed under a condition that is a set of events.
- d) A *feature* contains a set of actions.
- e) Features are located on *devices*.
- f) Some devices are owned by users.
- g) Users have *preferences*.

The underlined words point out the concepts of the user preference conceptual model. Each concept has attributes that detail it (see Table 1).

Table 1. User preference conceptual model

Concept	Attribute	Attribute type	CPL
Event	name	String	No
	issuer	Address	
	destination	Address	
	subject	String	
	content	Media – text	

	priority	Integer	
	data visibility	Boolean	
Device	address	Address	
	owner	String	
	availability	Event	
	simultaneous calls number	Integer	No
	waiting calls number	Integer	No
	language	String	
	features	Feature set	No
	media capabilities	String	
Action	actor	Address	
	condition	Event set	
	event (to issue)	Event	
	event attribute (to update)	String	
	device attribute (to update)	String	
Feature	location	Address	
	condition	Event set	
	actions	Action set	
	activated	Boolean	
Date and time	date and time	Date-and-time	
History	implementation dependant		No
Ordering	implementation dependant		

All the user preference conceptual model attributes are also attributes of *preferences* concept because they relate to user preferences. It means that all Table 1 content should be written again as attributes of preferences concept, but this not shown in Table 1 in order to avoid having a long table.

Most of the attributes are parameters extracted from the signaling flow or network nodes. Feature and preference parameters may be provided by external data sources. Three concepts can provide this type of parameters: *Date and time*, *History* and *Ordering*. Example of date and time preference parameter: “on Saturday, between 8.00 and 10.00”. History needs an application that logs the suitable information about previous incoming and outgoing calls. Ordering is performed on actions and features in order to run them in a given order. History and ordering attributes depend on the implementation. Some explanatory examples will follow in Section 6. Ordering can be as elaborated as allowed by temporal logic operators available in the preference expression language. These three concepts belong to the user preference conceptual model (Table 1).

Given architecture and user preference expression language and execution environment may not permit to implement the whole user preference conceptual model. Also, they may permit some implemented attributes to be preference ones only or feature ones only. For example, an architecture may allow appropriate user devices to handle many simultaneous calls or not. If this attribute is implemented, that is, user devices may handle many calls simultaneously, the architecture or preference language may permit to have the corresponding attribute as feature parameter. In such a case, a suitable feature should be available. They may allow the attribute to be a preference parameter and in this case, the user preference language execution environment should be able to detect how many simultaneous calls are currently handled or to ask the corresponding feature (simultaneous call queue, for example) about this information. The architecture and user preference language may also permit to have this attribute as feature parameter only because, for instance, there is no mean for the user preference language execution environment to be provided with the information about the currently handled simultaneous calls. We can also imagine the case where the information is available, which allows the attribute to be a preference parameter, but there is no means to control the number of simultaneous calls in the corresponding feature, because the latter fixes this number and does not allow to modify it. In such a case, the attribute can not be a feature parameter.

Thus, having an architecture and preference expression language that implement a given concept attribute means the following for each concept, according to whether the attribute is a preference or feature parameter:

a) Preference parameter:

- Event, device, action, feature, date and time, and history concepts:
 - the network allows the attribute value to be provided and thus envision suitable network features that provide that value,
 - suitable user preference language execution environment features collect the attribute value,
 - user preference language syntax contains the necessary expression items.
- Ordering concept can not provide a preference parameter. If some ordering information is a preference parameter, it is provided by history information that relates to history attribute.

b) Feature parameter:

- Event, device, action, feature, date and time, history, and ordering concepts:
 1. if the attribute (feature parameter) to update is controlled by a device feature,
 - the device administration policy allows updating the attribute,
 - suitable user preference language execution environment features invoke the controlling feature to update the attribute,
 - user preference language syntax contains the necessary expression items,
 2. otherwise,
 - suitable user preference language execution environment features update the attribute,
 - user preference language syntax contains the necessary expression items.

For example, ringing tone associated media content (concept: event, attribute: content) is controlled by the network or user device. This means that the network or user device have a suitable feature that controls ringing tone. In order to update this parameter value, the network or user device should make it possible for preference language execution environment to invoke the controlling feature. However, when updating a date and time feature parameter, it may not need for the preference language execution environment to invoke external feature to do that.

Notice that when we say given architecture and user preference language execution environment allow a concept attribute to be a preference or feature parameter, this does not mean it is so for every concept extension. For example, a given event attribute, such as content, may be a preference or feature parameter for some kinds of events but not for others.

4. IMPLEMENTATION REQUIREMENTS

We said that the architecture and preference language and execution environment may allow, for each attribute, to be or not to be a preference or feature parameter. This depends on requirements we have enumerated some of them in the end of the precedent section and will present the others in the present section. Let us categorize them into user preference language

requirements and architectural and user preference language execution environment ones.

a) User preference language requirements:

- The language should contain necessary items to express the implemented conceptual model attributes.
- It should contain appropriate operators and connectives in order to allow the personalization schemas presented in Section 3.

b) User preference language execution environment and architectural requirements:

- The user preference language execution environment should provide the necessary features to communicate with the network and external data sources.
- In the limits of security needs, flexibility may be needed in devices in order to permit specifying and updating certain preference parameters and issuing certain events. This flexibility should be offered by suitable features, may be an API. For example, in order to be possible for users to personalize network issued tones, the corresponding nodes should be flexible enough.
- Some attributes represent sophisticated user device or network component capabilities, for example, handling many simultaneous calls and managing a waiting call queue. As for the precedent requirement, this one also may be satisfied by suitable features.
- User devices start to be designed to room and run features. Some user preferences that concern user device-located features may need corresponding preference rules to be run on the device rather than on a centralized preference server. For example, for a preference rule to put a condition on invoking a user device-located service that is triggered by a user device generated event, it should be run on the device.
- Some preferences may need to combine features with user device-located applications. For example, user designed services based on the history attribute may need to combine a statistics program in the user device with a network-located feature. This requirement should also be satisfied in the limits of security needs.

5. USER PREFERENCE SPECIFICATION RELATED WORK

As it has been said in Section 3, in the Intelligent Network (IN), service personalization possibilities are very limited. All what users can do is to enter values of a few feature parameters, which are generally of address type. With IP telephony, user preference expression possibilities are more important because of the flexibility of the underlying architecture and signaling protocols. IP telephony offers the possibility to specify preference parameters and combine features, the latter being telephony or non telephony ones. For example, SIP offers a set of protocol built-in simple features, such as *accept call*, *reject call*, *redirect call*, *proxy call request*, etc.

SIP CGI [12] and *SIP servlets* [13] are two service creation technologies that may underlie a user preference language. At our knowledge, there is no language that is based on these two technologies. CPL (Call Processing Language), which is a user preference language, permits end users to specify their own preferences that are loaded and executed in a centralized CPL server. Because end users are non-trusted, the language limits their action possibilities. It does not allow variables, loops or external program running. The language primitives describe preferences within a decision graph that contains condition-action nodes. CPL interacts directly with SIP and focuses on some SIP features, mainly user location under conditions like date and time and user availability.

Since CPL preference expression possibilities focus on user location and availability, it does not permit to express some attributes of the user preference conceptual model introduced by the present article. The last column in Table 1 shows the attributes that are not expressed by CPL. Some of these attributes would not be difficult to express and implement in a CPL server, history for example. SIP CGI and SIP servlets are situated at low-level compared with CPL. They are not preference languages but can provide necessary tools to implement one that satisfies more language and execution environment requirements (see Section 4). A preference language based on SIP CGI or SIP servlets does not suffice to implement the whole conceptual model. The requirements stated in Section 4 have to be satisfied by the corresponding devices.

6. EXAMPLES

Here are some examples of preference rules. Each one focuses on a given attribute of the conceptual model. The examples are new feature types that, at our knowledge, have not been reported before. A simple and intuitive rule language is used to model them.

- a) If Do-not-Disturb service is activated and caller address is ayoub@happy.com, do not run Call Back service.

Concept: *device*, attribute: *features*.

Preference parameter and value: Do-not-Disturb-activated = true.

Preference parameter and value: caller-address = "ayoub@happy.com".

Combine two preference parameters: Do-not-Disturb, caller-address.

Preference rule: Do-not-Disturb-activated = True,
 caller-address = "ayoub@happy.com"
 -> activated := False, invoke Call-Back with activated.

- b) If subject is in some key word list, reject incoming call.

Concept: *event*, attribute: *subject*.

Preference parameter and value: subject belongs to given-key-word-list.

Preference rule: subject belongs to given-key-word-list
 -> invoke reject-call with caller-address.

- c) If somebody is trying to call for the 4th time in less than one hour, forward to home phone.

Concept: *history*, attributes: *trying-times*, *passed-time*.

Preference parameter and value: trying-times = 4.

Preference parameter and value: passed-time < 1 hour.

Preference rule: trying-times = 4, passed-time < 1 hour
 -> invoke forward-call with home-phone-address.

- d) If I'm on long distance call, play notification sound after each 10 seconds.

Concept: *ordering*, attribute: *interval*.

Preference parameter and value: call-type = "long-distance".

Combine two features: ordering, notification-tone.

Preference rule: call-type = "long-distance" ->
 interval := 10 seconds,
 invoke ordering with notification-tone, interval.

7. CONCLUSION

Emerging telecommunications architectures are expected to offer features that combine telephony, multimedia and data, and allow more user preference expression possibilities. This leads to complex sophisticated user preference expression and services.

At our knowledge, no work that presents a thorough analysis of the user preference concept has been reported in the literature. The present article aims at inciting interest to this research axis. It proposed a user preference conceptual model that permitted to specify service personalization logic schemas and architectural and language requirements that are necessary to fulfill in order to reach the biggest user preference expressive power. We have noticed that SIP telephony, coupled to CPL, which is a preference expression language, do not permit to implement the whole preference conceptual model, but do better than the IN (Intelligent Network) model.

Future work will report how the proposed conceptual user preference model has directed the adaptation of the service modeling language presented in [5-7] to express user preferences.

REFERENCES

1. 3GPP, <http://www.3gpp.org>.
2. Parlay Group, "Parlay Specification 2.1," 2000, <http://www.parlay.org>.
3. JAIN, <http://java.sun.com/products/jain>.
4. J. Rosenberg *et al*, "SIP: Session Initiation Protocol," RFC 3261, IETF, 2002.
5. Z. Chentouf, A. Khoumsi, and S. Cherkaoui, "Détection hors-ligne d'interactions de services dans les réseaux hétérogènes de télécommunications," Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'2002), Montreal, May 2002.
6. Z. Chentouf, S. Cherkaoui, and A. Khoumsi, "Implementing online Feature Interaction detection in SIP environment: early results," International Conference on Telecommunications (ICT'2003), Tahiti, French Polynesia, February-March 2003.
7. Z. Chentouf, S. Cherkaoui, and A. Khoumsi, "Mapping SIP onto a Feature Interaction Management Language," 7th International Conference on Telecommunications (ConTel'2003), Zagreb, Croatia, June, 2003.
8. NIST, <http://www-x.antd.nist.gov/proj/iptel>.
9. International Telecommunication Union, "General recommendations on telephone switching and signaling intelligent network: Introduction to intelligent network capability set 1," Recommendation Q.1211, Telecommunication Standardization Sector of ITU, Geneva, March 1993.

10. J. Lennox and H. Schulzrinne, "Call processing language framework and requirements," RFC 2824, IETF, May 2000.
11. J. Lennox and H. Schulzrinne, "CPL: a language for user control of Internet telephony services," IETF Draft, November 2000.
12. J. Lennox, H. Schulzrinne, and J. Rosenberg, "Common Gateway Interface for SIP," RFC 3050, IETF, January 2001.
13. "SIP servlet API," Java Community Process, JSR 116.