Using Context Information to Avoid Service Interactions in IP Telephony

Manuel Görtz, Ralf Ackermann, Andreas Mauthe, and Ralf Steinmetz

Multimedia Communications, Darmstadt University of Darmstadt,
64283 Darmstadt, Germany
{Manuel.Goertz, Ralf.Ackermann, Andreas.Mauthe,
Ralf.Steinmetz}@KOM.tu-darmstadt.de
http://www.kom.tu-darmstadt.de

Abstract. IP Telephony is an important application for building integrated communication services on IP networks. It provides telephony functionality similar to the Plain Old Telephone System. A rich variety of novel multimedia services is expected to augment the basic call functionality of IP Telephony systems. These services make the use of IP Telephony very attractive. However, multimedia services are much more complex than plain telephony services. IP-based telephony solutions are expected to be operated in a deregulated market where many different parties develop and deploy their own services. The conflicts and interactions that result from this practice make this application area challenging on a technical but also on an operational level.
The growing number of services with different quality in IP Telephony systems result in the service interaction problem, that is a well-known phenomenon in the traditional telephony system already. Existing services may interact with each other often in an undesired behavior. In this paper a rule-based approach to avoid local service interactions is proposed. The solution uses condition statements that have to be met when a service is to be executed. Current solutions consider mainly technical aspects of the end-systems or call session process as condition expressions. In the proposed approach individual user's demands are taken into account in a novel way. This corresponds to the inherent nature of a user centric communication. Context information of the user is considered as an implicit input to the rule decision process. Additionally, a more comprehensive expressiveness of the conditions can be achieved.

1 Introduction

Communication has become a sine qua non in today's life. It is a commodity for business processes and individual relationships. Current telephony functionality is mainly provided by Intelligent Network (IN) systems. These systems operate on a single purpose circuit switched network with a limited number of dedicated access points. The set of services in the Intelligent Network is small, but carefully designed. It is provisioned and maintained by the network provider. IP

Telephony uses packet switched networks, that are shared with other communication applications such as e-mail or instant messaging. The shared networking platform facilitates the integration of real-time voice communication with other IP based applications. Innovative services are expected as a consequence.

The telephone service has evolved over a century. Its highly esteemed attributes are stability, availability, robustness, and correctness. IP Telephony has to offer comparable properties otherwise users will not accept it as a serious alternative. IP Telephony solutions are going to operate in an open multi-vendor market with strong competition. Innovative and customized services will become the main differentiator between the different providers. The distributed nature of service intelligence and operational domains makes it harder to attribute responsibilities for a robust system.

The rising amount of services is predicted to have an impact on undesired service interactions. The *service interaction problem* describes the behavior of services that may be compromised when they are interacting. This is even more aggravated through semi-professionally designed services, but also because only little attention is paid on this problem in the IP Telephony area. We propose a solution scheme to avoid local service interactions especially for the class of custom user services. To clearly distinguish between services provided by the telephone network, such as call control and supplementary services, and services that offer tailored functionality to the user, we define the term *custom User Services*. These services are typically designed by the user itself and not in a professional development process. It distinguishes from traditional telephony services where only parameterization by the user is possible.

The approach exploits the rich information conveyed in IP Telephony signaling messages such as address field, subject, and media description. This information is used in combination with an Event-Condition-Action (ECA) framework. Interaction detection and resolution is shifted to a rule-based system. Additionally, ECA rules are adapted dynamically according to the user's current context. This is achieved by the use of context information as a new type of condition criterion. In addition the system becomes more user centric, which satisfies the demands of user communication better.

The rest of paper is structured as follows: Section 2 defines the problem domain for the proposed solution. The introduced approach to consider context information as a new kind of condition criterion is described in Section 3. The paper is concluded with a summary in Section 4.

2 Problem Domain

2.1 Services

The basic telephony call functionality is similar in IN and IP Telephony. Major differences can be identified in the service development and provisioning mechanism. A *service* provides additional and supplementary functionality for the user and the network. The terms service, feature or service feature are often used interchangeably. It is often not possible to distinguish features from

services. Moreover, the service itself and service provisioning are often used synonymously. Thus, individuals tend to have their own vague understanding of what a service is. In this document these concepts are defined as follows:

> A *service* σ is defined as a meaningful set of capabilities provided by an existing or intended set of systems to all who utilize it [1].

Services are often composed of feature or service features. In this context these are defined as:

> *Features* ϕ are packages of incrementally or optional added functionality to the base service [2].

The compositions of features to form a service can be denoted as

$$\sigma = \sigma_B \oplus \phi_1 \oplus \phi_2 \oplus \ldots \oplus \phi_n \tag{1}$$

where σ_B denotes a particular basic service that is extended by the features ϕ_i. In this context the operator $\oplus$ determines an arbitrary feature-composition operation.

Different standards describe services that are using protocol primitives. For Intelligent Networks these are defined in *capability sets* such as CS-1 [3]. Services for IP Telephony using protocol primitives are defined in their core signaling specifications (Session Initiation Protocol (SIP) [4] and H.323 [5]). Additional supplementary services are defined in the ITU-T H.450.n series, RFCs and Internet drafts. IP Telephony provides mechanism for another type of services. Custom user services are realized by high-level programming concepts. SIP Common Gateway Interface (SIP-CGI) [6] and the Call Processing Language (CPL) [7] approach are main concepts. In the context of the system described later the focus is on SIP and CPL as a scripting language to develop and provision custom user services.

2.2 Service Interaction

Service development for the Intelligent Network is accomplished by a small group of telephony experts. Standardization bodies carefully designed these service to ensure their proper and flawless functionality even among different vendors. However, in specific situations the behavior of the overall system is disturbed if a new serviced is introduced to the system. This phenomenon is called *service interaction problem* or *feature interaction problem.*

An *interaction* occurs if one additional feature modifies or disrupts the behavior of the existing services in the system. This may be in a desired or necessary way, but in most cases it is an unwanted result. A service interaction can be formalized using the previous definitions of services and features. It occurs if

$$\mathcal{S} \oplus \sigma_i \models p_i \qquad 1 \leq i \leq n \tag{2}$$

and

$$\mathcal{S} \oplus \sigma_1 \oplus \sigma_2 \oplus \ldots \oplus \sigma_n \not\models p_1 \wedge p_2 \wedge \ldots \wedge p_n \tag{3}$$

are both true. $\mathcal{S}$ denotes the system. Further, let $p_1, p_2, \ldots, p_n$ be the properties of the services, such that the service satisfies the property denoted as $\sigma \models p$.

Services are often developed by different vendors. These check their services for correctness and potential implications without having full knowledge about other existing services. Verifying services against interaction with other services rises exponentially with the number of services. These are the cause of unwanted interaction. However, service interaction is a basic principle to combine services in order to create more sophisticated functionality, as expressed in Equation 1. Service chaining is another use for desired service interaction. Therefore, service interaction is an inherent and persistent feature of service development and execution, hence methods for a correct handling are needed.

Different kind of service interactions exist. A taxonomy on the different feature interaction classes based on the *nature of interaction* is given in [8]. We focus on the class of *Single User Single Component* (SUSC) interactions from this taxonomy. A service interaction of the SUSC class arise because of *functional ambiguities* between concurrently running services. This type of interaction corresponds well to the targeted class of custom user services where multiple services can be active at the same time in order to achieve a user centric communication.

Example. The following example of parallel active *Voice Mail* (VM) and *Call Forwarding on No-Reply* (CFNR) services illustrates the situation of local service interaction. Figure 1 shows the corresponding situation during an incoming call and a no-reply condition of the subscriber. The system can not execute both services that provide a no-reply treatment properly. This problem arises because the *CFNR* service forwards the incoming call to a specified address, after the phones rings a certain time. The *VM* service in turn offers the caller that a voice message is recorded for the original callee. □

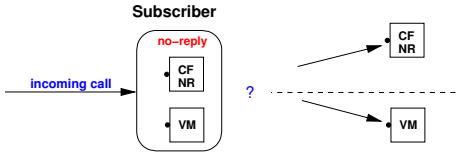

Fig. 1. The parallel active services Voice Mail and Call Forward on a no-reply condition

2.3 Related Work

The service interaction problem is well known in the context of the Intelligent Network and research effort in academia and industry has been dedicated to this problem. Different solution approaches can be applied at individual phases of the service development and deployment process. The solution space can be divided into three categories. In the first category there are *off-line techniques* which are characterized by the application for formal methods. *On-line techniques* are

another approach that provides a combination of detection and resolution mechanisms. They are only applicable if the targeted interaction problems can be resolved at run-time. The combination of on-line approaches with knowledge about services and interactions from off-line methods characterizes the domain of *hybrid techniques* [9].

Off-line techniques are well suited for application in the design phase. Services are typically described and analyzed in a formal description language. The usage of these analytical methods have compelling arguments. They have a mathematical foundation and use techniques such as extended Finite State Machine (FSM), process algebra, and logic. The repertoire of off-line techniques can be expressed as some form of analysis for reachability, termination, deadlock, nondeterminism or consistency. A constructive approach is provided by the *Distributed Feature Composition* (DFC) [10]. This framework serves as a virtual architecture for exposing and managing service interactions in multi-party, multi-feature and multi-media sessions in telecommunication networks. Individual services are treated as black-boxes that communicate via an internal message exchange. The origin target platform of this approach was the PSTN, but an implementation of the DFC concept for IP networks is available as well [11].

A quick time-to-market support is one of the key benefits of on-line techniques. They also fit better to the multi-vendor market, where global knowledge about all services disappears. Services interaction is typically first detected when the services are actually deployed and activated. The on-line method adopts to probe service and their interactions in a testbed. Usually a priori knowledge about services that presumably interact is used. A number of approaches aim at probing without an interaction matrix which is gained afore. This is achieved by run-time collection of data in an isolated test-bed process. Solutions use centralized entities [12] with the capability of observing and controlling the call process. Alternatively, features need the ability to communicate with other features in an out-of-band manner and negotiate a proper resolution [13].

2.4 Summary

Existing off-line techniques for service interaction detection have originally been developed for IN telephony services. Nevertheless, an analysis show that these techniques are also well suited for application in the context of service design in IP Telephony. This is especially true for services that use protocol primitives. These services are the outcome of a thoroughly standardization process. Additionally, these services are typically designed by experienced developers.

Mechanism that allow the design and provision of custom user services are an advantageous feature that IP Telephony offers. This enables the user to create own services that fit best to their individual needs. However, the users are often not familiar with any of the above mentioned formal description techniques. Existing mechanisms do not reasonable support the development of high-level custom user services by users.

3 Approach

The approach proposed in this paper fills the gap that has been highlighted in the previous sections. It avoids local service interactions with a focus on custom user services. Services developed by 3rd-party service developer and provider as well as skilled end users are the targeted group. Formal description techniques as used in many off-line techniques require additional skills from the user and have a steep learning curve. This is a not negligible additional burden. It is also in contradiction to the intention of easy service creation. The basic principle of the proposed avoidance is the utilization of an Event-Condition-Action rule framework to control service execution.

3.1 Usage of Event-Condition-Action rules

An approved concept is used to control service execution. The *Event-Condition-Action* (ECA) rule framework is well-known in the area of active databases [14]. However, it is novel to use this concept in the context of service interactions. Special schemes for the detection and resolution of conflicts within the rule sets exist in this context. These are considered helpful for the approach introduced here. An ECA rule r can be written as

$$r : \{E\} : C \rightarrow A$$

where $\{E\}$ denotes the set of triggering events. C states the rule's condition and A states the rule's action. After an event or a combination of events triggers a rule, the condition statement of the rule is evaluated. If the condition expression is true the service specified in the action part is executed.

Triggering is achieved by session related events like incoming or outgoing calls but also by external events. Several operators support the combination of single event descriptions [15, 16] to express more powerful statements. A disjunction $(e_1 \vee e_2)$ is true if either one or both of the events occur. A conjunction $(e_1 \wedge e_2)$ specifies that both events have to occur within a specified time interval without considering their order. The sequence $(e_1; e_2)$ yields a true result if both events occur in exactly the described order. Negation (not e_1 [*interval*]) is true if the event has not occurred during the specified time interval.

3.2 Condition and States

Conditions can be any combination of predicates that evaluate time, address, state information, etc. Time conditions offer an appropriate way to specify the system behavior. The conditions C are evaluated after a rule is triggered. The condition statements are represented in a Disjunctive Normal Form (DNF):

$$(c_{1,1} \wedge c_{1,2} \ldots \wedge c_{1,n_1}) \vee (c_{2,1} \wedge c_{2,2} \ldots \wedge c_{2,n_2}) \vee \ldots \vee (c_{m,1} \wedge c_{m,2} \ldots \wedge c_{m,n_m})$$

where $c_{i,j}$ denotes the single conditions that are either `TRUE` or `FALSE`. A DNF fits well with the natural expression of condition statements where all conditions

within an alternative statements have to be true. Of all rules that match (the *candidate set*) one is selected using a *conflict resolution policy*. The resolution scheme uses priorities to chose the rule. If multiple candidates are still left additional information like time-stamps is taken into consideration. The first rule in the candidate set is chosen if no unambiguous decision exist. Finally, the selected rule is *fired*, which is that its action part is executed.

Within the condition statements a time specification that is compatible to the time definitions in the Internet Calendaring (iCal) standard [17] is used. Time conditions offer an appropriate mechanism to arrange the services corresponding to the user's daily routine. Different time scales and expressions, (such as time ranges and recurring events) can be described with the time condition. Another condition type describes address information.

The signaling semantic of SIP is exploited to obtain call specific information like addresses, subjects, and media descriptions. The SIP header contains information which can be evaluated for address selection. A complete match but also substring matches (e.g. only domain name) can be applied on the address header fields. Address information together with white and black lists allow a comprehensive mechanism to classify incoming calls. If an address is matched with a specific address in a white-list the address condition is true otherwise it is false. Black-list entries explicitly deny service execution if the matched address is in the list. In all others cases it is allowed.

Session states can also be utilized as condition statement. These states represents specific phases of the session. These states can be described on different abstraction levels. The *Basic Call State Model* (BCSM) provides a set low-level states. The model describes the transition between the specific states during call setup. Session states can typically be obtained by the analysis of the exchanged signaling messages. The signaling in SIP is achieved by the exchange of requests and responses. These signaling messages are standardized and allow to conclude the current status of the caller, callee, and the session.

The above mentioned information provides a useful basis for fine grain condition expressions. However, there are drawbacks with these kind of condition statements. The set of high-level call states is very limited. Moreover, they describe generally only the end-system or the session process. Low-level states are implementation depended and restricted to a specific end-system. Most notably, the condition expressions mainly describe a static state of the system. This does not correspond with the goal to support a user centric communication.

A benefit is gained if the current user situation is considered as a condition statement as well. This situation is referred as the user's *context*. The use of contexts information allows to cover the dynamic behavior and communication demand of a user. The behavior of service execution controlling can be automatically adapted to the user's context. Context provides a novel condition statement that enables to execute the most appropriate service in a certain situation. An automatic context estimation process serves as an implicit input to the conflict resolution strategy of the system.

3.3 Context as an Adaptive and Dynamic Momentum for Service Execution

Most people have a general idea about what context is. However, there are diverse (and often vague) notions about what the term actually describes. Throughout this paper the following definition of context adapted from [18] is used:

> *Context* is any information that can be used to characterize the situation of a subject and its interaction with optional objects. Objects are persons, places, or applications that are considered relevant to the subject.

The combination of several context values provides a very powerful mechanism to determine the current situation. Location, entity activity and time are typical context sources and are forming the *primary context*. Knowledge of the current location and time together with a user's calendar lets an application have a good estimation of the user's current social situation. It is preferable that the user's context is detected automatically and used as an implicit input to the rule evaluation process. Applications that consider context information are able to *adapt* to the situation. This property is add to the system that evaluates the condition part of the introduced approach. This allows a more user centric avoidance of service interactions. Carefully specified rules may be fitting well but can be inappropriate in certain situations. This depends on the user's context and therefore this should be considered. Time specifications that can be used to estimate the agenda of a user can only provide a coarse schedule. The specification can be described with recurring time periods (lunch, working hours, etc.) but also with time spans over a longer time, such as holiday, travel. Context is advantageous in coping with dynamic daily activities.

3.4 Context Determination

The following issues need to be addressed to incorporate context into the Event-Condition-Action rule-based framework to avoid service interactions. The context has to be established with an appropriate certainty. Afterwards it has to be announced to the place where the condition statements of the rules are evaluated. Context can be obtained on different abstraction levels. Physical and logical sensors represent the lowest input layer. The collected raw data is processed and finally mapped onto a context. Single context information can be combined to form a higher-level context. The context of being *in a meeting* for instance can be determined by a combination of location, time and calendar information.

Location information is an often used and valuable context. The type of obtained information exists with different granularities. The kind of information can be in a symbolic notation or refer to a physical geographic position. Further, location sensing can be accomplished with different methods. A device can actively measure its position. Alternatively, the infrastructure provides location information in form of beacons (e.g. in RFID tag environments) that are sent out. Accurate positioning for outside usage that is determined by the device itself can be achieved e.g., by the Global Position System (GPS).

For indoor location estimation the RADAR method [19] provides a suitable approach for the proposed system setup. This is an proximity approach that uses an IEEE 802.11 wireless LAN infrastructure. The signal strengths (SS) between mobile clients and access points (AP) are measured to perform a comparison with a pre-measured signal strength map. A calculated distance measure yields the approximate location. Experiments in our lab show that the coverage and also the gained accuracy is sufficient to determine the room a user is currently in [20]. This type of location sensing is a beneficial side-effect provided by a wireless network infrastructure. A drawback can be seen in the fact that the user has to carry a PDA-like device with a wireless network card. This is needed for active location sensing but also for the detection of the device by the infrastructure.

Other high-level context information that describe the user's current situation are *presence information*. The feasibility of presence information has been proven in large scale Instant Messaging and Presence systems. Typical presence information are *busy*, *available*, *out-of-office*. For instant messages using SIP the Rich Presence Information Data Format for SIP (RPIDS) [21] is a suitable format. It consists of a Common Presence and Instant Messaging (CPIM) compliant format and defines tags such as privacy, placetype, and category. RPIDS focuses on the automatically deriving of presence information since users are often not diligent in updating their status.

3.5 System Architecture

Several requirements have to be fulfilled by the proposed system to allow a user centric avoidance of service interactions. Firstly, the system must be able to control and execute the services on behalf of the user. Therefore, it needs to be aware of the session state as well as of the current context of the user. This functionality is combined in the logical *ECA entity*, which is shown on the right in Figure 2. The entity consists of three main building blocks described next. Vovida Open Communication Application Library (VOCAL) was chosen as the platform for this system. VOCAL provides a fully fledged Open Source SIP-centric IP Telephony system. The system covers all parts, defined in the SIP core standards, as well as additional functionality.

The *Rule Enforcement* point is the core of the ECA-based solution. The stored rules are retrieved, selected and evaluated if a specific event occurs. Conflict resolution policy is achieve in this component. Services and CPL scripts are executed in the *Service Execution* component according to the matched rule. Therefore, the ECA entity has to be located in the signaling path in order to react on incoming and outgoing calls events. Further, this enables the ability to parse SIP messages. Access to the information conveyed in the messages is required to assay the condition statements based on session state and header fields. The *B2BUA* component is chosen to provide the demanded functionalities. It is a logical entity that contains two SIP User Agents (UAs) working back-to-back. It appears like a pure SIP proxy, but unlike a pure proxy the B2BUA allows 3rd party call control. It handles different calls legs, remains in the call and maintains the complete call state.

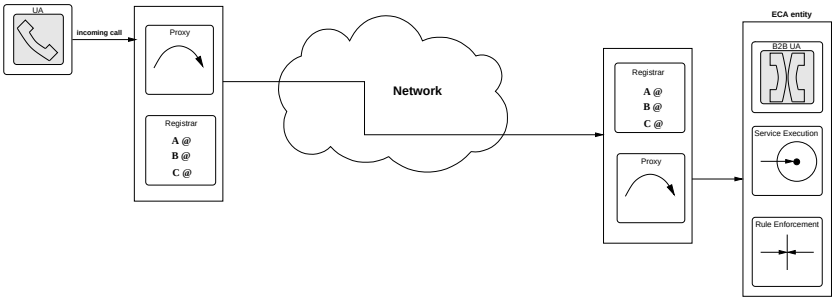

Fig. 2. The ECA-SIP System

For the evaluation of context condition criteria in the rules the system needs to be aware of occurring context changes. The client communicates context changes via an asynchronous transmission mechanism such as an event messaging technique. The push mechanism is in favor of the pull mode. The push mode guarantees shorter response time, but typically results in a larger number of exchanged message. This kind of event message transmission and the subscription mode for the proposed system is provided by the SIP event framework [22]. SUBSCRIBE and NOTIFY messages fit well into the SIP environment that is used for IP Telephony systems. The system setup is shown in Figure 3. The ECA entity subscribes to the clients for a specific context change using SUBSCRIBE messages. The user location is the context of interest in this context. The client's device determines its own location as describe in Section 3.4. Alternatively, it can use other techniques. If a location change occurs, the client sends a NOTIFY message with its current context to all subscribers.

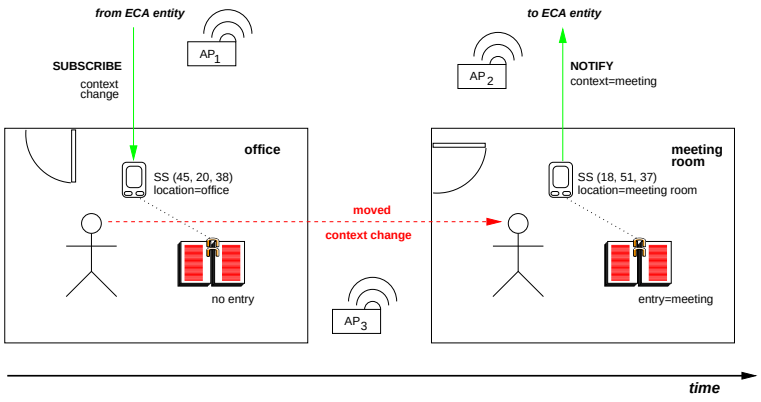

Fig. 3. Context change of the user is announced to the ECA entity via a SIP NOTIFY message

3.6 Evaluation

The example in Section 2.2 with simultaneously active *Voice Mail* (VM) and *Call Forwarding on No-Reply* (CFNR) services will be used to illustrate how the system operates. An incoming call which is not answered within a specified time triggers all services that have subscribed to this event. The potential conflict can be resolved if rules prevent both services to be executed at the same time by specifying condition statements. The user's context is considered for the evaluation of the condition statement. If the user is in a meeting the call should be forwarded to his secretary (in case it is urgent) otherwise a voice mail should be recorded. A rule may look like the following (in pseudo notation):

$$\texttt{on}\ \textit{incoming call}\quad \texttt{if}\ \textit{context} = \textit{meeting}\quad \texttt{do}\ \textit{CFNR}(\textit{addr} = \textit{secretary})$$

The context *meeting* is determined when the user is located in the meeting room and an appointment in the user's calendar is marked as "meeting". The according NOTIFY message is issued to the ECA entity when the user enters the meeting room. An incoming call triggers the event mechanism and the condition expression is evaluated. Since the user is in the context *meeting* the *CFNR* services is executed and the call is redirected to the specified address.

4 Conclusion

IP Telephony offers mechanisms for a variety of new multimedia communication services. The rising number of services will aggravate the service interaction problem in IP Telephony. The proposed approach tackles this problem with an Event-Condition-Action framework. Local interaction can be avoided using a rule resolution strategy. We have enhanced this approach by adding a user centric component to the condition part. Context information is considered as an additional condition criterion. Context such as location and status gives the systems the ability to determine the service that is most appropriate for the user's current context. The system subscribes to context changes with the clients. After a context change occurred the subscriber will be notified. This shifts the service control from a purely call-related to a more user-centric treatment.

References

1. TINA-Consortium: TINA-C Glossary of Terms (1997)
2. Bowen, T., Dworack, F., Chow, C., Griffeth, N., Herman, G., Lin, Y.J.: The feature interaction problem in telecommunications system. Software Engineering for Telecommunication Switiching Systems (1989) 59–62
3. International Telecommunication Union: Introduction to Intelligent Network Capability Set 1. Recommendation Q.1211, Telecommunication Standardization Sector of ITU, Geneva, Switzerland (1993)
4. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol. RFC 3261 (2002)

5. International Telecommunication Union: Packet based Multimedia Communication Systems. Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland (2000)
6. Lennox, J., Rosenberg, J., Schulzrinne, H.: Common Gateway Interface for SIP. RFC 3050 (2001)
7. Lennox, J., Schulzrinne, H.: Call Processing Language Framework and Requirements. RFC 2824 (2000)
8. Cameron, E., Griffeth, N., Nilson, Y.J.L.M., Schnure, W., Velthuijsen, H.: A feature-interaction benchmark for IN and beyond. IEEE Communications Magazine **31** (1993) 64–69
9. Aggoun, I., Combes, P.: Observers in the SCE and SEE to detect and resolve feature interactions. In: 4^{th} International Workshop on Feature Interactions in Telecommunications and Software Systems. (1997) 192–212
10. Jackson, M., Zave, P.: Distributed feature composition: A virtual architecture for telecommunincation services. IEEE Transactions on Software Engineering **24** (1998) 831–847
11. Bond, G.W., Ivanić, F., Klarlund, N., Trefler, R.: ECLIPSE Feature Logic Analysis. In: IP Telephony Workshop, New York (2001) 100–107
12. Marples, D., Magill, E.H.: The use of rollback to prevent incorrect operation of features in intelligent network based systems. [23] 115–134
13. Griffeth, N.D., Velthuijsen, H.: The negotiating agents approach to runtime feature interaction resolution. [24] 217–235
14. Widom, J., Ceri, S.: Active Database Systems. Morgan-Kaufmann, San Mateo, California, USA (1995)
15. Chakravarthy, S., Mishra, D.: Snoop: An expressive event specification language for active databases. Technical Report UF-CIS-TR-92-041, University of Florida (1993)
16. Gehani, N.H., Jagadish, H.V., Shmueli, O.: Compose: A system for composite event specification and detection. Technical report, AT&T (1992)
17. Dawson, F., Stenerson, D.: Internet calendaring and scheduling core object specification (iCalendar). RFC 2445 (1998)
18. Dey, A.K.: Providing Architectural Support for Building Context-Aware Applications. PhD thesis, Georgia Institute of Technology (2000)
19. Bahl, P., Padmanabhan, V.N.: RADAR: An in-building RF-based user location and tracking system. In: IEEE INFOCOM, Tel-Aviv, Israel, IEEE Computer Society Press (2000) 775–784
20. Goertz, M., Perez, A., Ackermann, R., Mauthe, A., Steinmetz, R.: Location Sensing with RADAR. Technical Report TR-KOM-2003-09, Multimedia Communications Lab, Darmstadt University of Technology (2003) ftp://ftp.kom.tu-darmstadt.de/pub/papers/GPA+03-1-paper.pdf.
21. Schulzrinne, H., Kyzivat, P., Gurbani, V., Rosenberg, J.: RPIDS - Rich Presence Information Data Format for Presence Based on the Session Initiation Protocol (SIP). Internet draft (2003) Work in Progress.
22. Rosenberg, J.: A Presence Event Package for the Session Initiation Protocol (SIP). Internet draft (2003) Work in progress.
23. Kimbler, K., Bouma, L.G., eds.: Feature Interactions in Telecommunications and Software Systems V. IOS Press, Amsterdam, The Netherlands (1998)
24. Bouma, L.G., Velthuijsen, H., eds.: Feature Interactions in Telecommunications Systems. In Bouma, L.G., Velthuijsen, H., eds.: Feature Interactions in Telecommunications Systems, Amsterdam, The Netherlands, IOS Press (1994)