Using a Multi-Layer Approach to tackle the Service Interaction Problem in Telephony Scenarios

Manuel Görtz, Ralf Ackermann, Martin Karsten, Ralf Steinmetz
Darmstadt University of Technology
Multimedia Communications (KOM)
Merckstr. 25, 64283 Darmstadt, Germany
{Manuel.Goertz, Ralf.Ackermann, Martin.Karsten, Ralf.Steinmetz}@KOM.tu-darmstadt.de

Abstract

The inauguration of IP telephony as a new and promising technology has opened the door to a variety of enhanced communication services and applications, which differ from those of the PSTN. The shift from circuit switched to packet-based networks, from a central to a more decentral organization, from professional and vendor driven development to an open development environment enables a multitude of opportunities.

A variety of tools and methods for service creation has be developed to ease the process of design and implementing services. But these new freedoms hold new risks (well-known from the PSTN area), which are currently not in the main focus of standardization. The rising problem can be overcome by the use of formal methods and checks which also tackle the important problem of feature interaction. In this paper we present an multi-layer concept using a mixture of formal and informal techniques like service pattern to approach this topic.

1 Introduction

The primary goal of *IP Telephony* was to offer telephony-like services over packet-based networks. There is a consensus in using RTP [34] and its companion protocol RTCP for the media transport. Several competing protocols are currently striving to act as the dominant signaling protocols though. The major actors are the ITU (International Telecommunication Union) driven H.323 protocol suite [20, 26], and IETF's (Internet Engineering Taskforce's) approach – the Session Initiation Protocol (SIP) [18] and Megaco [7].

Activities are now also focusing on one of the main promises of IP based telephony, the boundless possibilities to create new and innovating communication services, that use the whole range of possibilities that arise when using feature-rich signaling protocols.

These *value added services* are believed to be *the* distinction to the Plain Old Telephony System (POTS). IP telephony forms the promising technology to provide enhanced and new communication and application scenarios to a number of users. IP telephony benefits from the opportunity to design protocols, telephony components and means for service creation from scratch, avoiding known drawbacks from the Public Switched Telephone Network (PSTN).

In analogy to these new mechanism a shift in the conceptual view on service creation, deployment and execution as well as on the relationship model is taking place. IP telephony follows the "fast in the core and smart at (or near) the edges" concept and thus reflects the distributed nature of the Internet. In general, endsystems have control of the call state and are the convergence point of the signaling and the media path.

Having expressive signaling protocol semantics for signaling and providing mechanisms for customization leads to the effect of greediness of feature designer to start creating services. While this is an desired goal in service creation, it can quickly lead to services with "amateur" character, being written without taking the interactions with other already existing services into account. Further these services will probably not be formally checked by a comprehensive set of tools.

Besides the gained freedom in feature development and deployment, the proven concepts and tools in feature interaction and formal methods from the PSTN should not only be kept in mind, but transfered and established in the IP based communication world.

The paper is organized in the following way. Section 2 shows the differences between PSTN and IP telephony especially under the focus of service creation. The next sections covers the aspect of service interaction and, while Section 3 examines possible and existing solutions. Our approach using a multi-layer concept is explained in detail in Section 4. Finally the paper is concluded by a summary and an outlook on the future of communication services and their verifications for IP telephony.

2 Communication Services

The term “communication services” covers a multitude of different, alternative or contrary features, based on a communication network. This section tries to classify these services with the constraint, to concentrate on IP telephony and its distinctions to the legacy PSTN (shown in Figure 1) and the Intelligent Network (shown in Figure 2).

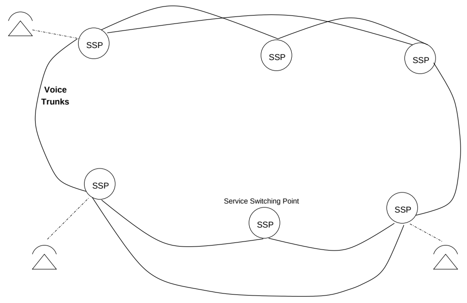

Figure 1. PSTN scenario

2.1 Conceptional differences between PSTN and IP Telephony

First general properties and differences between these two architectures are given, before a detailed description of the service types and location follows. The underlying setup for this investigation is consisting of “intelligent” endsystems with user-interaction and IP-network connectivity.

The Intelligent Network is a service-independent architecture that allows the provision and operation of new services. It consists of an packet-based signaling overlay network with a few central Signaling Transport Points (STP) connected with each of the Service Switching Points (SSP). The Common Channel Signaling System No 7 (SS#7) uses an out-of-band signaling system for the trunk communication.

In packet based networks and also in the special case of IP telephony, the signaling path can be different from the media path. This is probably the largest architectural distinction to the legacy PSTN.

Figure 3 contains IP telephony infrastructure components (server and gateways) and fits into the setup of a general SIP or H.323 IP telephony scenarios. Another, but here not considered model is using gateway control protocols such as MGCP or Megaco also known as H.248 [27] to provision services by call controllers steering the behavior of gateways or end devices [19].

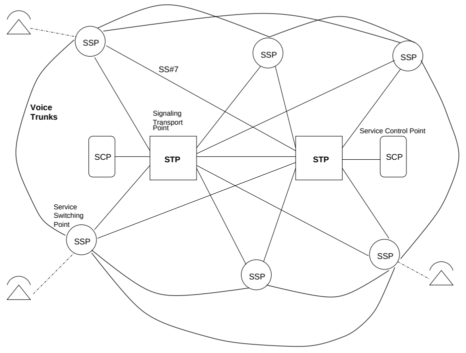

Figure 2. Intelligent Network (IN) scenario

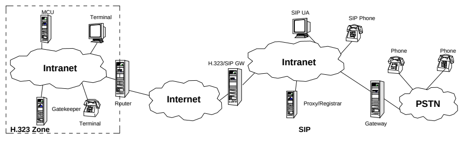

Figure 3. IP Telephony scenario

The IP telephony setup has a more decentralized approach using in-band signaling protocols. Servers are usually placed near the endpoints and often become part of an intranet domain. While this approach leads to a flexible architecture it complicates an global consistent maintenance. Services can be deployed by any administrator of a domain, without central control – this installed service that may cause unwanted service interaction.

2.2 Service Creation

A *Service* is defined as a meaningful set of capabilities provided by an existing or intended set of systems to all who utilize it. Further is a service an incremental or optional unit of functionality added to the base functionality [10]. A range of different services classes can be distinguished in the area of IP telephony. Going up in the layers we find *Call Routing Services*, *Call Control Services* and finally *Call Application Services*.

The POTS is offering Call Routing services with its basic means of setting up conversations and billing for features, whereas services like Call Forwarding or Call Completion, modify the basic telephony service and belongs to Call Control services. On top reside additional applications like small program or scripts altering the communication behavior and reacting to signaling or user events.

Today’s telecommunication networks, its services and applications are under control of the network operator’s domain. The Intelligent Network system is well suited for simple applications, that serve the mass market. But this approach offers not enough flexibility to cope with the need of easy creation and rapid development of innovative applications that integrated voice communication into the enterprise or even the user system.

To overcome this drawback a joint effort, inspired by the TINA [3] work, of two groups, namely Parlay [2] and JAIN [1], continue with the development of an open API, that allows applications to access the core functionalities of the network. This framework called Parlay/OSA (Open Service Access) offers, among other features, interfaces between the core network and the Service Network, here reside (logically) the applications. The development and deployment of these services is independent of of the access and the underlying core network.

In IP telephony the integration of voice and data has reached its maximum. Both transported in packets using either TCP or UDP, can be processed in a similar fashion. Using this techniques both the administrator of the network, but also third-parties can develop services or features for the IP telephony. This paper concentrates on service creation for IP Telephony systems. For theses different mechanism exist, which are described below in more detail.

2.2.1 APDU or Methods to enable Call Routing and Call Control Services

To provide services on the "lowest" signaling level the use of protocol intrinsic elements is needed. Within each protocol of the H.323 suite (H.225.0 [23], H.245 [25], H.450 [21]) (A)PDUs (Application Protocol Data Units) are defined, which carry the signaling information in their protocol elements. The IP telephony equivalent to establish, modify or tear down a call is the utilization of messages to realize a request and response model. Each of the messages used here consists of a header, with a number of header lines and a body part, how it is done in SIP. The creation of own new services is possible through the use of new headers or methods (SIP) or the definition of special vendor elements (H.323). A use is then only possible within a vendor specific network, while other entities are relaying theses messages or sending errors back. Though these mechanisms exist and the definition of own elements is allowed, it is not desired, because it complicates interoperability and interworking. Changes affect the whole protocol suite, endpoints and all interworking facilities. For interoperable changes and extensions they have to go through an iterative standardization process before moving to maturity.

2.2.2 SIP CGI

The *SIP Common Gateway Interface* (SIP CGI), proposed in [31], offers a program language independent mechanism to realize services. The possible use of high-level languages (C, Java) as well as the scripting languages (e.g. Tcl/Tk, Perl) for rapid prototyping makes this approach very attractive. SIP has a number of similarities to HTTP [15] and so has SIP CGI to the CGI known from the web. SIP is inheriting its client-server interaction and much of its syntax and semantics from HTTP, which makes this approach familiar to web programmers. CGI offers programming language independence, the exposes of all headers, the creation of responses and component reuse, as well as the ease of extensibility. The main difference is the *persistence model* in SIP CGI that allows a script to maintain control through multiple message exchanges. HTTP CGI has no persistence for its scripts. The state is maintained by allowing the CGI to return an opaque *token* to the server. When the CGI script is called again for the same transaction, this token is passed back to the CGI script. When called for a new transaction, no token is passed.

2.2.3 Call Processing Language – CPL

The *Call Processing Language* (CPL) [32] approach primarily addresses service parameterization by untrusted developers or users. It uses XML for notating the processing of parameters and the intended functionality. Following the IN-Service model, CPL is strictly formalized and uses a decision graph technique, hence a *Directed Acyclic Graph* (DAG), to describe the control flow. This allows methods for analyzing worst-case paths and a guarantee for well-defined termination. CPL scripts can be transported and placed either by out-of-band means but preferably using core protocol mechanisms such as the transport as payload of a SIP REGISTER message.

2.2.4 Servlets

Another approach is the use of the Servlet technology. A *SIP Servlet* [30] is a specialized Servlet which executes telephony service logic. It is written in Java code and interacts with a SIP server or a "Servlet-Engine". The standardized Server API [12] allows to run the code on all Servlet-enabled servers. Because it defines a possibly standardized framework and JAVA compile- and runtime checking as well as security precautions, it can conceptually be seen inbetween the stringent restrictions of a CPL and the complete (but often also undesired and dangerous) openness of SIP CGI.

2.2.5 Telephony APIs

In between providing a complete language that is capable of direct reaction and creation of protocol specific mechanism are the Telephony APIs. The most conversant and widespread ones are Sun's Java TAPI (JTAPI) [33] and Microsoft's TAPI Version 3.0 [11]. The intention of providing an API is to offer a standard set of methods to create calls and to hide the underlying telephony hardware from the application logic. Support for IP telephony applications in existing APIs is just starting to emerge and does not cover the full set of signaling protocols nor functionality yet.

2.2.6 Summary

For creating services for IP telephony a variety of different approaches exist. Most of them are aiming at the Call Control Services and have their background in techniques known from the web. Programmers familiar with these methods can easily start developing services for IP-based telephony.

The focus of these tools lie in simplification and increasing productivity, which broadens the range of possible developers and the number of services. There is no main attention on formal verifications, which holds the risk of semi-correct services possibly leading to undesired services interactions.

2.3 Service Interaction

As the amount of features and developers expand the issue of *feature interaction* or *service interaction* arises. The term of feature interaction was first coined in the early 80s by Bellcore. A first framework dealing with this large problem area was provided by [8].

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. Many services are developed isolated without having all possible interactions with other existing services in mind. The services can be verified and tested for itself though. The phenomenon occurs only when these services are combined. A further observation that can be made is, that there is a different peculiarity for local and distributed systems.

2.3.1 Local interaction

A local interaction can appear, when two services, a service is shown in a schematic view in Figure 4, conquer for the same input stimulus. The conflict can be detected in the endpoint, where the services are executed. The resulting logical conflict can also be solved in the endpoint directly. Either by providing priorities to each service or by signaling the information back to the user, who can decide which service should be active.

An example for this behavior is, when a user has accidently activated on his phone call forwarding on no-reply (CFNR) as well as the voice mail (VM) service, where the caller can leave a message. An incoming call that has to be processed with a "no-reply" routine can not be admitted to both, CFNR and VM at the same time.

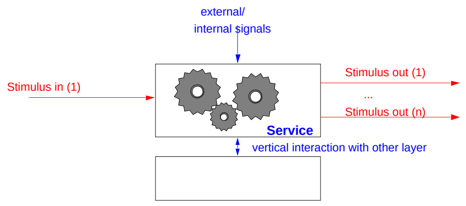

Figure 4. Schematic view on a service

2.3.2 Distributed interaction

While the local interaction can be detected and solved in the endsystem itself, the *distributed interaction* is more difficult to discover. Generally two well-defined and function services are activated on two different devices and problem arises, when these influence the behavior of each other. While most of the unwanted behavior be fixed through error handling, timer or exceptions, some issues are on logical level and can be solved only by the user.

To illustrate the phenomenon the following services and their interaction is used. The *Call Forward* service [22] permits a served user (A) to have incoming calls addressed to the served user's number redirected to another number (B) depending on the condition of the service (unconditional, busy, no-reply).

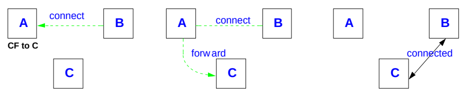

Figure 5. Call Forward service

Another service from the H.450 series is the *Call Completion* service [28], which is a supplementary service that is offered to a calling User A. On encountering a not available called User B, it allows User A to request that User Bs endpoint monitor User B and notify User As endpoint when User B becomes free. On response by User A to that notification, User As endpoint shall attempt to complete the call to User B. The fact is depicted in Figure 6

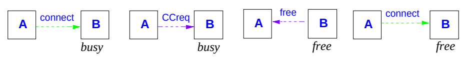

Figure 6. Call Completion service

If now these two service interact, as seen in Figure 7, an undefined and possible unwanted state can be reached. User B is busy, so A request a call-back from B (2). While B is still busy A activates a call forward to user C (3). B's call back is now forwarded to C (4), while B thinks he reaches A.

State (5) may be unwanted and undefined as well. The interaction can lead to a *technical* problem, because C is unable to understand the call completion messages. But the interaction can have also a *social* component, because A doesn't want that C knows, that he wanted to talk to B.

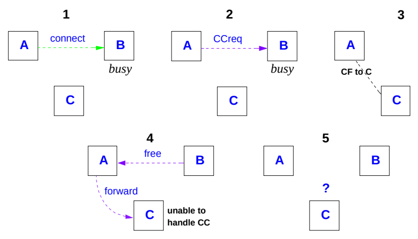

Figure 7. Call Completion service and Call Forwarding interact with each other

2.4 Observation

The flexibility of potentials that the rich signaling protocols offer and the multitude of carriers and developers existing in IP telephony environment increases the probability of features and services interacting in an unwanted and unforeseeable way. Numerous approaches tried to tackle the feature interaction problem for the IN, using formal techniques for verifications.

Whilst these methods are widely used for the IN, their broadening and utilization for the next generation of telephony is lacking behind, but the risk of feature interference is there and will be grow especially with the emerging of Application Services.

The ease with which everyone can realize and deploy its own communication services is cure and disease at the same time. Communication services show a temporal and occurrence behavior which not only interacts with services installed on the same entity, but also with other services active on other entities in the system. To cope with such issues formal methods are proposed and already used in the area of PSTN and IN. Some of the promising techniques for IP telephony are examined next.

3 Formal Methods for Communication Services

With the rising number of entities using IP networks for telephony and their services, the risk of behavior grows. If IP telephony moves away from an experimental platform and becomes a commercial service, failures will cause monetary costs, which is not tolerable.

IP telephony aims to replace (parts) of the existing PSTN and has to match with it in terms of reliability and correctness. Formal Methods (FM) have been advocated as a useful tool to achieve an increase in software reliability.

3.1 Existing Formal Methods for Communication Services

A survey and comparison on todays formal methods especially for communication services is done in [13]. It is also shown that even the most liberal formal methods are rarely used in the industry [17]. On the other hand the use of the three standardized formal description techniques (FDTs) LOTOS [5], Estelle [6] and SDL [24] have found several design errors in a number of communication protocols.

Investigations on industry projects, that are using formal methods to validate their communication service design and implementations, show that the industry favors SDL, Z [29] and Promela [14].

Beside the validation of a protocol design for communication services, a variety of research has been focusing on the phenomenon of *feature interaction*. A *feature* in this context is an incremental of optional unit of functionality added to the base functionality. An interaction occurs when 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 [36].

3.1.1 Estelle

The formal description technique Estelle, defined within ISO (International Organization for Standardization), is designed for the specification of distributed and concurrent processing systems, in particular communication protocols. An Estelle specification describes a system of communicating extended finite state machines (EFSMs). These allow the internal storage and modification of values/variables. Estelle permits to separate the description of the communication interfaces between components of a specified system from the description of the internal behavior of each such component. The internal processes are using asynchrony communication via channels.

3.1.2 SDL

The formal method named Specification and Description Language (SDL) grew out of an engineering notation (based on flow diagrams) used in telecommunications for describing protocol flows. SDL was standardized by the CCITT (now the International Telecommunications Union) and is the most widely used FDT in the telecommunications industry, satisfying most of their needs. Its technique is based on the model of extended finite state machines (already known and similar to Estelle), in a structured hierarchy. Its main difference from Estelle is that it developed from a graphical notation, while the former was purely textual. SDL's graphical representation (SDL/GR) can be transformed into a equivalent textual representation (SDL/PR) as well.

3.1.3 Promela

The Promela Language is a verification modeling language, which provides means for making abstractions of protocols that suppress details that are not relevant to interaction. Programs written in Promela consist of global processes and global or local messages and variables.

3.1.4 Z

The Z language is a formal specification language that makes it easier to write mathematical description of complex dynamic systems such as software. The descriptions are usually smaller and simpler than any programming language can provide. They should contain a mixture of formal and informal parts. The central part of Z is based on the mathematics of set theory and first order predicate calculus. There are means for defining and checking types of Z elements and Z schemas for structuring specifications.

3.1.5 LOTOS

The FDT LOTOS (Language Of Temporal Ordering Specification) was developed by ISO to support standardization of OSI (Open Systems Interconnection) and was initially based on the formal specification language CS (Calculus of Communicating Systems). Although LOTOS was originally applied to OSI, it has now been applied more widely to sequential, concurrent, and distributed systems generally. It follows the concept of process algebra based on Calculus of Communicating Systems, using finite observation FSM as internal processes. The states of a process are "stored" in its process history and information can be obtain by external observation

3.2 Formal Methods in IP Telephony

All these formal methods were investigated in the context of the communication services and protocols for the PSTN and the Intelligent Network (IN). Whilst the academic community has promoted formal methods and applied a large variety of these to communication services, only a few methods have gained the acceptance of the telephony industry and were used for verification of design and implementation of services and their possible interaction.

With the introduction of IP telephony the attention to validate against feature interaction seems to be forgotten. Formal methods or checking are usually only applied to separate protocols or notation, description or programming languages like CPL. The protocols of the H.323 suite are described in the Recommendations also in SDL, IETF's RFCs and drafts to SIP and associated items.

Further the ITU-T based protocols feature a large case of failure situations and solutions. Whilst this is so far not seen in IETF's IP telephony approach SIP and its call control drafts, which lies in the different background and other modus operandi.

Other constrains why these techniques are not widely used in current IP telephony scenarios can be found also in the time-to-market issue. The competition between big and established vendors and small and flexible start-up companies has gained a new dimension in IP telephony. The market is more open and attractive for new and innovative ideas and products, that are unfortunately often not thoroughly tested.

4 Multi-Layer Approach

To draw more attention to the issue of service interaction and to provide usable mechanisms a multi-layer approach to conquer the problem is introduced. The different layer that are incorporated are shown in Figure 8.

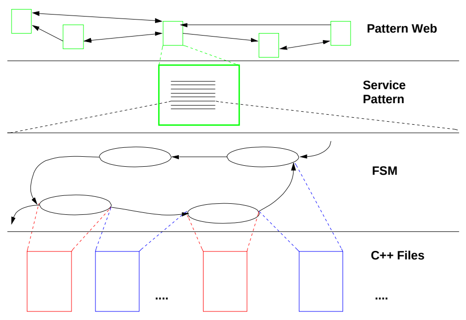

Figure 8. The different layer, that are incorporated in the multi-layer approach

From top to bottom we find the pattern web, which is the view on the relationship between the individual service pattern, which describe services in a structured way. Each service can be representated by an (extended) finite state machine, which in turn can be also representated in FDT.

On the lowest layer, we find the individual program files, which can be in an 1-to-1 mapping of states and transitions, which is for example the case in Vovida's VOCAL [4] IP telephony system.

The two main concepts are the *Service Pattern* and the *decomposition of FSM*, which are explained in more detail.

4.1 Service Pattern

To offer an attractive and easy to use entrance into a more formal creation of services a high-level and informal mechanism, termed *Service Pattern* is proposed. A Service Pattern expresses the relation between a certain system and the forces which occur by the services and represents a solution to the arising problems within a particular context.

The principle idea behind the service pattern is the establishment of a *formalized documentation* from each individual service. Each pattern consists therefor in structured way of mandatory and optional field, that describe different aspects and properties of a service.

The informal nature of this concept lies in the possibility to write descriptive text into each of the fields. This lowers the burden to gain knowledge about a specific Formal Description Technique, but a created service can easily be described by everyone.

The formalized part within a service pattern comes from the strict structure of each pattern by defined field entries. An entry can either be mandatory (e.g. name, description) or optional (e.g. alias, usage).

Advantages of using patterns for an formal documentation are that it is a known concept from the OO-programming, where design pattern [9, 16] are used. A programmer of services might therefor be already familiar with pattern. The service pattern also inherit ideas from *Security Pattern* [35].

The power of service pattern is obtained by the *Pattern Language*, that defines a collection of patterns and the rules to combine them into an architectural style. This language builds a layer that consists of the single pattern and the relations between them and is called in Figure 8 *pattern web*.

The goal is to visualize the relations and especially the interactions between the different services. Through the relation ”is_a” it is possible to use the transitivity between services and its derived variants. If Call Completion and Call Forwarding interacts, than will Call Forwarding On Busy also interact with Call Completion.

The web of service patterns offers expert knowledge for intermediate users and helps them to avoid design errors. While this concept aims at a more informal approach, sharing knowledge and offering a low barrier access to the area of formal verified communication services, it does not provide a formalized technique, which is done on the layer of FSM.

4.2 FSM decomposition

The formal description techniques SDL, Estelle and Lotos shown in Section 3. These techniques rely on communication processes and extended Finite State Machines. This fact is taken into account and provides the base for the concept of decomposition of FSM into pattern.

In analogy to Software Engineering we find two known concepts. From the Object Oriented programming the *Structural Pattern* is used to map the internal structure of the service, while the techniques of *Communication Pattern* from parallel programming is used to describe the inter-service communication.

The goal is to lower the complexity to build a complete extended FSM for a service and the communication between another service in order to analyze the possible wanted or unwanted interaction.

To achieve this goal FSM representation with a corresponding description in a FDT of existing services are analyzed and de-compositioned into segments that are verifiable for itself and is contained in other services as well.

These small fragments are fed into a FDT and can be reused in other services, where this pattern also occurs. If the services itself can be split into building blocks, which can be mapped onto verified FSM blocks, new services can be combined using formal correct blocks and additional functionality, then the complexity of verification will be lowered.

4.3 Evaluation

The multi-layer approach, that was proposed, tackles the service interaction problem from two sides. From an informal and descriptive documentation using service pattern and from the de-composition of FSM into pattern, providing together with FDTs an formal approach.

The concepts relies on know techniques, like pattern, and includes proven tools of the formal description of telecommunication services, like SDL.

The high-level mechanism offers an attractive semi-formal procedure to avoid design errors for programmers, who are unfamiliar with existing FDTs, but also an tool to visualize dependencies and relations between services.

To cope with the time-to-market constrains and still verify services against service interaction, the pattern web offers a pre-selection, which service combinations need to be checked and which not. The de-composition and re-use of FSMs lower the complexity of the overall process, thus reduces the time needed to check for correctness.

The idea to tackle with a multi-layer concept rises the possibility to be used as valid and needed tool for the service creation in IP telephony, because it combines different techniques and entries to the interaction problem and gives solutions on different scales and granularity in terms of time, effort and formal correctness.

5 Conclusion

The majority of standards are written in natural language, which makes it difficult to be precise and unambiguous. The problem that there is not an uniform understanding of the standard is given. The scale and complexity of communication protocol standards lead to the development of formal description techniques, to provide a common and formal correct view on the definitions.

This approach helps the specifier to write standards in a clear, concise and verifiable way. It also helps the implementors as a guidance on how to build their products. Finally, test suites who need to check and evaluate implementation options, can precisely be defined as the basis for rigorous conformance tests.

In the area of traditional telephony networks (PSTN, IN) a multitude of these formal method approaches from industry and academia exist. The formal methods not only tackle the correctness of a communication design and implementation, but also their interaction, known as the feature interaction or service interaction problem.

Currently no equivalent view on the importance of the feature interaction exist in the IP telephony environment for the used protocols and the rising amount of services. Our paper names this situation, categorizes the aspects that have to be investigated and proposes a multi-layer approach to this topic. The benefit of our concept is the offering of different granularities in terms of effort to put into formalization, quality of correctness to obtain and time to spent on verifications. This will form the basis to apply formal methods to the IP telephony domain.

References

[1] JAIN. http://java.sun.com/products/jain/.

[2] Parlay Group. http://www.parlay.org.

[3] TINA-Consortium. http://java.tinac.com.

[4] Vovida vocal system. http://www.vovida.net/cgi-bin/download_vocal.pl?stack=vocal.

[5] I. I. 8807. LOTOS – A formal description technique based on temporal ordering of observation behavior. *ISO/TC97/SC21*, Nov. 1988.

[6] I. I. 9074. Estelle – A formal description technique based on extended state transition model. *ISO/TC97/SC21*, 1999.

[7] M. Arango, A. Dugan, I. Elliott, C. Huitema, and S. Pickett. Media Gateway Control Protocol (MGCP). *RFC 2705*, October 1999.

[8] T. Bowen, F. Dworack, C. Chow, N. Griffeth, and Y. Lin. The feature interaction problem in telecommunication systems, 1989.

[9] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture, A System of Patterns*. John Wiley & Sons Ltd, Chichester, England, 1996.

[10] E. Cameron. A feature interaction benchmark for in and beyond, 1994.

[11] M. Corp. Telephony application programming interface (tapi) version 3.1. *http://msdn.microsoft.com/library/default.asp ?url=/library/en-us/tapi/tapi30portal_7k85.asp*.

[12] A. P. Deo, K. R. Porter, and M. X. Johnson. The SIP Servlet API. *Java SIP Servlet API Specification*, April 2000.

[13] F. Dietrich and J.-P. Hubaux. Formal methods for communication services.

[14] P.-A. Etique. *Service Specification, Verification and Validation for the Intelligent Network*. PhD thesis, Swiss Federal Institute of Technology, Lausanne, 1995.

[15] R. Fielding, J. Gettys, J. Mpdgiö, H. Frysyk, and T. Berners-Lee. HTTP: Hypertext Transfer Protocol – http/1.1. *RFC 2068*, January 1997.

[16] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Abstraction and Reuse in Object-Oriented Designs. In O. M. Nierstrasz, editor, *ECOOP'93: Object-Oriented Programming - Proc. of the 7th European Conference*, pages 406–431, Berlin, Heidelberg, 1993. Springer.

[17] D. Gries. The need for education in useful formal logic, 1996.

[18] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. *RFC 2543*, March 1999.

[19] C. Huitema, J. Cameron, P. Mouchtaris, and D. Smyk. An architecture for Internet Telephony services for residential customers. *IEEE Network*, 13:50.57, May/June 1999.

[20] International Telecommunication Union. Visual Telephone Systems and Equipment for Local Area Networks which provide a non-guaranteed Quality of Service. *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, May 1996.

[21] International Telecommunication Union. H.450.1: Generic functional protocol for the support of supplementary services in H.323. *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, February 1998.

[22] International Telecommunication Union. H.450.3: Call diversion supplementary service for H.323. *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, February 1998.

[23] International Telecommunication Union. H.225.0: Call signalling protocols and media stream packetization for packet-based multimedia communication systems. *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, September 1999.

[24] International Telecommunication Union. Specifi cation and Description Language SDL. *Series Z: Programming Languages. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, Nov. 1999.

[25] International Telecommunication Union. H.245: Control protocol for multimedia communication. *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, June 2000.

[26] International Telecommunication Union. Packet based Multimedia Communication Systems. *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, November 2000.

[27] International Telecommunication Union. Recommendation H.248: Gateway control protocol. *Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, June 2000.

[28] International Telecommunication Union. Recommendation H.450.9: Call completion supplementary service for H.323.

Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, November 2000.

[29] ISO/IEC. Z – base standard version 1.0. *ISO/IEC JTC1/SC22*, Nov. 1992.

[30] A. Kristensen and A. Byttner. The SIP Servlet API. *Internet Draft draft-kristensen-sip-servlet-00.txt*, Sep 1999. Work in progress.

[31] J. Lennox, J. Rosenberg, and H. Schulzrinne. Common Gateway Interface for SIP. *RFC 3050*, January 2001.

[32] J. Lennox and H. Schulzrinne. Call Processing Language Framework and Requirements. *RFC 2824*, May 2000.

[33] S. Microsystems. Java Telephony API (JTAPI). *http://java.sun.com/products/jtapi/*.

[34] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *RFC 1889*, January 1996.

[35] J. Yoder and J. Barcalow. Application security. In *The 4th Pattern Languages of Programming Conference, 1997.*, 1997.

[36] P. Zave. Feature-oriented description, formal methods, and dfc.