The Feature and Service Interaction Problem in Telecommunications Systems: A Survey[*]

Dirk O. Keck , Paul J. Kuehn
Institute of Communication Networks and Computer Engineering (IND)
University of Stuttgart, Pfaffenwaldring 47, D-70569 Stuttgart (Germany)
Phone +49 711 685-7964
FAX +49 711 685-7983
e-mail: {keck, kuehn}@ind.uni-stuttgart.de

October 1998

Abstract

Today's telecommunications systems are enhanced by a large and steadily growing number of supplementary services, each of which consists of a set of service features. A situation where a combination of these services behaves differently than expected from the single services' behaviours, is called service interaction. This interaction problem is considered as a major obstacle to the introduction of new services into telecommunications networks. In this contribution, we give a survey of the work carried out in this field during the last decade. After a brief review of classification criteria that exist for feature interactions so far, we use a perspective we call the *emergence level view*. This perspective pays respect to the fact that the sources for interactions can be of many different kinds, like, e. g., requirement conflicts or resource contentions. It is used to rationalise the impossibility of coping with the problem with one single approach. Afterwards, we present a framework of four different criteria in order to classify the approaches dealing with the problem: The general kind of approach taken, a refinement of the well-known detection, resolution, and prevention categories, serves as the main classification criterion. It is complemented by the method used, the stage during the feature lifecycle where an approach applies, and the system (network) context. The major results of the different approaches are then presented briefly using this classification framework. We finally draw some conclusions on the applicability of this framework and on possible directions of further research in this field.

Index Terms (Keywords): Telecommunication Service, Supplementary Service, Service Interaction, Feature Interaction, Service Interference, Feature Interference

[*]This report has been published as a regular paper in **IEEE Transactions on Software Engineering, Vol. 24, No. 10, October 1998, pp. 779–795**. This work has been partially supported by the Software-Labor of the University of Stuttgart, Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany.

1 Introduction

The term “Feature Interaction Problem” in telecommunications systems has been coined in the early 80s by Bellcore. In 1988, Bowen *et al.* [1] have taken the first effort to provide a framework to this large problem area. Since then, researchers and practitioners from academia, research centers, and industry are working towards a viable solution. The work carried out in this area has already grown enormously, discovering many new aspects of the problem. This article is going to give an overview on the research that has already been done, and also some help to structure this large field.

Our aim is to provide a common base for the classification of approaches towards the problem, to make it easier for researchers to compare their own work with existing approaches, and for other people interested in the field to have a concise source of information on the state of the art.

1.1 Definition of Terms

Very often, an imprecise definition of the terms used in this field is a source of confusion. For this reason, an informal definition of important key terms is summarised in this section.

- *Service (Telecommunication Service).* According to the ITU-T [2], a service is offered by an administration to its customers in order to satisfy a specific telecommunication requirement. Telecommunication Services are divided into two broad groups: A *bearer service* is a type of telecommunication service that provides the capability for the transmission of signals between user-network interfaces. A *teleservice* is a type of service that provides the complete capability, including terminal equipment functions, for communication between users according to protocols established by agreement between administrations. In the class of teleservices, a number of basic teleservices can be identified. Examples are telephony, facsimile, or data transmission.

- *Supplementary Service.* In ITU-T terminology [3], a “supplementary service modifies or supplements a basic telecommunication service. [...] It must be offered together with or in association with a basic telecommunication service. The same supplementary service may be common to a number of telecommunication services.” Well-known examples for *supplementary services* are Call Forwarding Unconditional (CFU), Call Waiting (CW), or Credit Card Calling (CRED).

- *Service Feature; Feature.* The term *feature* is defined as a “unit of one or more telecommunications or telecommunications management based capabilities a network provides to a user.” in [4]. ITU-T defines the term *service feature* as the smallest part of a service that can be perceived by the service user [5]. In that context, the term service refers to a telecommunication service.

- *Feature Interaction; Service Interaction.* Using the definitions above, a distinction between the terms feature interaction and service interaction has to be made, as already pointed out in [6]. The term *feature interaction* refers to situations, where different service features or instances of the same service features affect each other. This can take place within the same service as well as between features of different services. In these situations, the term *service interaction* applies. Generally, service interactions can be observed when two telecommunication services affect each other. It has to be noted, that in this context, a telecommunication service is represented by a basic telecommunication service plus an arbitrary set of supplementary services.

- *Feature Interference; Service Interference.* The term interference is used to express the *undesirability* of an interaction in the sense presented above [7]. However, sometimes it is difficult if not impossible to generally specify whether the behaviour of a service or a service feature is desired or not.

The term *feature interaction* is applied in most publications as the most general term for the description of the problem, because most interactions between services or supplementary services can be tracked down to interactions between single service features, and because an interference is defined as an undesired interaction, i. e., a special case of an interaction.

1.2 The Problem

The increasing demand for new telecommunication services has led to a rapidly growing number of new services, as well as to the enhancement of existing services with new service features. This trend is expected to continue in the future [8]. There are many different aspects, why the phenomenon of feature interactions is considered as a major problem. Some of these will be discussed in this section.

In order to facilitate the procurement of new services, the architectural concept of the Intelligent Network (IN) [5, 9, 10] has been developed. It divides the functions of the basic call processing from the execution of supplementary services and thus presents a well-defined interface for the realisation of new services, easing that procedure. At the same time, the danger of interactions is increased due to the larger number of new services, the less substantial experience needed to develop such services, and the lack of automatic interaction treatment tool support.

As the number of features and services grows, the amount of work which has to be invested into the treatment of interactions explodes, because the number of possible combinations of features and services grows exponentially with their number. Thus, the interaction problem dominates the software development period [1]. For the same reason, the complete testing of the services and features in combination has become almost unmanageable. The interaction problem has become a major obstacle to a time and cost efficient introduction of new services, and also increases the risk for a service provider.

Two examples from the area of telephony features illustrate the problem and demonstrate the character of real interactions. Consider a situation where participant A has subscribed to the service *Originating Call Screening (OCS)* and does not want calls to party C to be put through. He calls party B, who has activated the service *Call Forwarding Unconditional (CFU)* to party C. What is going to happen with A's call? If he is not put through, he may be confused, as B is not on his screening list. Otherwise, the intention of *OCS* not to be connected to C is invalidated. Whether this is acceptable depends from the forwarding party's intention (see Zave [11] for more details).

In the second example (by *E. J. Cameron*), a problem is sketched that occurs only if three service features are active in a special constellation, although pairs of these features do not show any interference. Party A, subscriber of an *Unlisted Number* service calls party B, subscriber of an *Automatic Recall* of the last caller. If B uses this feature, A's number appears on B's phone bill, as he is also subscriber of *Itemised Billing*. This is clearly a violation of the intention of A's *Unlisted Number* service. For more and detailed examples, the reader is referred to [12, 13, 14]. Some more introductory and tutorial papers on the problem are [15, 16, 17].

The consequences of introducing interactions into a network can be significant. Although Kuhn [18] reports no failures of the PSTN (public switched telephone network) due to feature interactions in an investigation carried out between April 1992 and March 1994, the results reach from confusion and annoyance of users up to financial loss for customers, service providers and network operators. Furthermore, the cost of the development of a new service may be increased enormously or, even worse, the service does not reach the marketplace at all.

Interestingly, the feature interaction problem is not limited to the telecommunications environment, but occurs also in other large evolving software systems, as pointed out by Aho and Griffeth [19]. Therefore, the results achieved should be considered relevant to large areas of software and systems engineering research.

1.3 Taxonomy of the Problem

A sound taxonomy for a large problem area is a valuable step towards the management of the problem. There is already a number of schemes in the literature, the most important of which we would like to outline here.

- The *causal view* [13] is the most widely used taxonomy, as it already provides hints to the source of the individual problems as well as it indicates where possible solutions may be found. In this view, three main problem sources are identified: limitations of network support, intrinsic problems in distributed systems, and violations of assumptions.
- The *lifecycle view* [6] classifies feature interactions according to the phase during a software lifecycle where they can be managed best, as displayed in Table 3 later in this contribution.
- The *configuration view* [13] identifies the logical entities involved in an interaction case. Cameron *et al.* differentiate the number of users and the number of components involved, and they distinguish between user and system features.
- The *organisational view* [6] focuses on the responsibility for an interaction and its consequences. Thus, it is an interaction management issue.

These taxonomies have proven to be valuable in order to address the different aspects of the problem and are applied to classify the contributions listed in [20], although each of them covers the problem only partially. For the selection and assessment of methods tackling the feature interaction problem, we consider a scheme sketched in [21] as a better starting point, as it provides a framework that points to the nature of the problem and can be exploited in order to decide where a specific problem has to be addressed. We call it the *emergence level view*.

Interactions emerge from different levels in a system. Combes *et al.* [21] separate the logical level, the network level and the implementation level. This view is adopted in principle in the present contribution, although the levels are called *logical level, abstract architecture,* and *concrete architecture,* the architectural terms taken from [22]. This model matches the conceptual model underlying the Intelligent Network (INCM) [5] quite well. The highest level, the *logical level* is used to model concepts like purpose and intention, focused to the users' perspective. It corresponds to the service plane in the INCM. The middle level uses an *abstract architecture* in order to model services on a network-oriented level, that corresponds to the Global Functional Plane and Distributed Functional Plane in the INCM. The Physical Plane of the INCM should be understood as a concrete architectural view. However, a concrete architecture takes into account all the irregularities of real system implementations, like, e. g., physical limitations (memory, bandwidth), interworking with legacy systems and different signalling protocols and procedures, which are not or incompletely covered by the Physical Plane of the INCM.

In this model, several facts can be observed, that are relevant for an overall solution of the interaction problem. Each of the levels can be the source of interactions, however of different kinds. Thus, it is practically impossible to have one single method that is able to cope with all these kinds of interactions. Interactions can be inherited from higher to lower levels, becoming manifest in a different way. In these cases, it is possible to treat the interaction at this lower level, although this is not advisable, as it is only the treatment of the symptoms and not of the source of the interaction, which is situated in the level the interaction emerges from. It is also possible, that an interaction is not inherited during the mapping of a higher level to a lower one, as, e. g., an ambiguity in the execution order is replaced by a deterministic decision. This may lead to hazardous situations, because the interaction is still present in principle, but does not become manifest. These observations are visualised in Fig. 1.

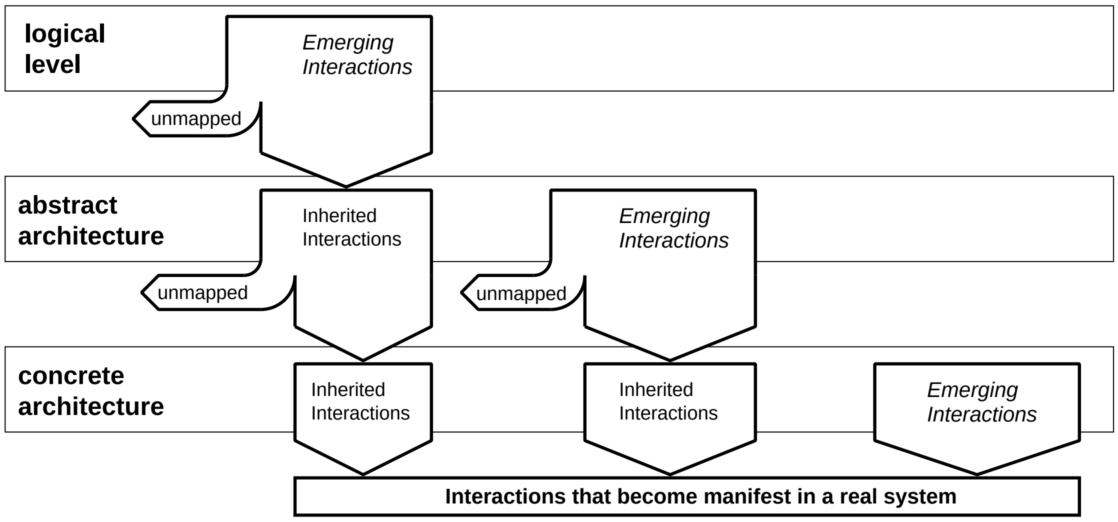

Figure 1: Inheritance of interactions in the emergence level view.

The following conclusions can be drawn: Interactions should be addressed at the levels they emerge from, using methods suitable to the application at these levels. It is necessary to have a framework that supports the mapping from one level to the next lower one, like the Intelligent Network conceptual model, which goes into the right direction, but still has several imprecision, like, e. g., the interworking with legacy services. Furthermore, the question of suitable granularity for each level and the achievement of a correct mapping from one level to the next one is still a topic for research.

2 Classification Framework for Approaches

In this section, a classification framework for methods dealing with interactions is introduced, that will be used throughout the rest of this contribution. Although the emergence level view described above is valuable for the understanding of the complexity of the problem and to rationalise the necessity of having a combination of methods in order to solve it, it does not provide enough resolution to classify methods that deal with interactions.

The generally accepted categorisation of approaches based on the terminology of avoidance, detection and resolution introduced by Cameron and Velthuijsen [6] and refined by Bouma and Velthuijsen [23, Introduction] to the grouping displayed in Fig. 2 is chosen as a starting point. Although this framework has proven to be useful, the huge amount of research carried out in this field and the broad spectrum of methods that are applied suggest a refinement and extension that is given in Section 2.1.

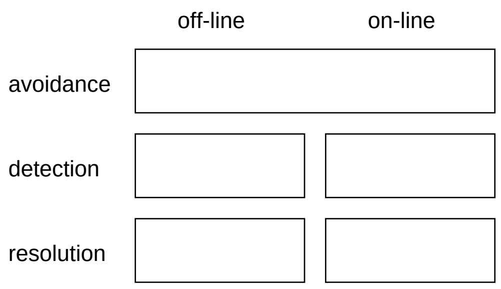

Figure 2: Categorization of approaches to addressing feature interactions (after [23]).

As a second measure, the *method* which is applied is selected. Although, of course, there is a wealth of methods in the different individual approaches, these can be fit quite easily into a comparatively small and sound number of categories (which are described in Section 2.2). The scheme is supplemented by the state during a *feature lifecycle*, where a given method applies. Here, we consider the simple division into "off-line" and "on-line" as too coarse. Independent of the three categories above, it makes sense to differentiate the *system context* of the different solutions. These categories are described in the following subsections.

2.1 Approach

Table 1 displays a refinement of the well established terminology of detection, resolution and prevention.

- Detection
  - Interaction
  - Interference (undesired)
- Resolution
  - Restriction
    - General
    - Situation-specific
  - Integration
  - Cooperation
    - Conscious
    - Oblivious
- Prevention (Avoidance)
  - Structural (system structure)
  - Procedural (specific design process)
- Management

Table 1: Classification of approaches

Detection denotes the identification of the existence (presence) of interactions. As it is practically (for realistic systems) impossible to prove the absence of mistakes and an interaction not necessarily is undesired, it cannot be expected that a network is free of any interactions. Concerning the presence of interactions, two directions within the approach can be identified: the identification of *interactions* in general has no notion of the undesirability of an interaction, but merely detects situations where certain criteria are met. The detection of *interference* situations needs the concept of desirableness, as only undesired interactions are identified, desired ones are ignored.

Resolution usually should be the consequence of detection. If an interaction between two service features is detected, resolution may follow several different routes. The simplest approach is *restriction,* i. e., it is avoided that both features find themselves in a situation where they interact with each other. Characteristic for this approach is that each feature itself does not modify or adapt its normal behaviour. Two sub-approaches can be identified: *general restriction* is characterised by a static set of rules which disallows the activation of one feature if another feature is present, *situation-specific restriction* is characterised by a set of rules that takes into account the current situation in which the features are activated.[1] *Integration* of both features into one with larger functionality may be another strategy to handle interactions. *Cooperation* of the interacting features is the most sophisticated approach in order to resolve interactions. In this approach, the features change or adapt their behaviour in order to resolve the problem. *Conscious cooperation* is achieved, if one of the features possesses explicit knowledge about the other and vice versa, and if one or both are equipped with mechanisms to resolve their specific interactions. In *oblivious cooperation*, there exist general mechanisms that require no service specific knowledge in order to resolve interactions with unknown counterparts.

There is a number of approaches that include or integrate the detection and resolution of interactions. In such cases, the two aspects of the methods are discussed separately, as no separate class for methods integrating detection and resolution is introduced for sake of clearness.

Prevention is used for approaches where feature interactions cannot occur. These approaches can be divided into two subgroups: *structural* approaches avoid the occurrence of interactions by a suitable system structure, including, e. g., architecture and protocols. *Procedural* approaches attack from another direction, introducing new strategies or restricting existing processes of service creation, or giving sets of design guidelines, for example. This distinction has already been identified in [6], although not formulated generally. We prefer the term prevention to avoidance, because the difference to approaches integrating detection and resolution becomes clearer.

As the problem is so complex, and the number of feature combinations grows exponentially with their number, it is inevitable to add *management* issues to the list of approaches. Under this heading, all techniques are collected that consider aspects of the management of the problem as a whole, like, e. g., managing the huge number of test cases or identifying and filtering relevant scenarios out of the large number of possible feature combinations. Sections 3 to 6 on detection, resolution, prevention and management of feature interactions are ordered according to Table 1.

2.2 Method

- Design oriented methods
    - Feature design
    - Feature execution control
    - Feature execution environment
    - System design and architecture
- Analytical methods
    - Formal techniques (verification)
    - Informal methods (heuristics)
    - Experimental techniques (testing, simulation)

Table 2: Classification by method applied

[1] As an example serves the well-known circular CFW problem. A general restriction can be achieved by a limitation of the number of forwarding steps, while a situation-specific restriction could be achieved by checking if a circular situation really has occurred.

Two different kinds of methods that are applied can be isolated. *Design oriented methods* like *feature design,* the application of mechanisms for *feature execution control,* or the provisioning of a *feature execution environment* as well as issues of overall *system design architecture* attack the problem by tailoring the design of some system parts appropriately. *Analytical methods* are applied in order to cope with feature interactions in existing systems or system models. In this context, *formal techniques (verification),* play a major role, although *informal methods (heuristics),* and *experimental techniques (testing, simulation)* have to be mentioned as important counterparts.

2.3 Lifecycle

For a thorough classification of approaches to the feature interaction problem, the two categories of "off-line" and "on-line" are too simplistic. A more detailed scheme, which is mainly an adaption of the ETSI (European Telecommunications Standards Institute) reference model [24], is used in this section. However, as the lifecycle view is not really orthogonal to the approach (Section 2.1), it is not applied consequently in the classification of the actual work in Sections 3 to 6. The value of the lifecycle category lies in its suitability as a foundation for the definition of service interaction handling processes [25].

Off-line		
	Specification	
		Requirements (User level)
		Network level
		Detailed specification
	Implementation	
On-line, but not yet publicly available		
	Testing	
	Deployment	
On-line, publicly available		
	Provisioning	
	Registration	
	Activation	
	Invocation	
	Course of execution	
	Deactivation	
	Erasure	
	Withdrawal	
Feature removal		

Table 3: Classification by lifecycle stage

The off-line part starts with the *specification* phase that can be further subdivided. The requirement specification level, also termed user level, corresponds approximately to the logical level in the emergence level view. The network level specification and the detailed specification correspond to the abstract architecture in terms of the emergence level view. In the latter category falls the work supported by a Service Creation Environment (SCE) in the IN context, if the internal specification of the service independent building blocks are regarded as a part of this specification. The *implementation* phase concludes the sequence of off-line service development. In the IN approach, the implementation step is replaced by an automatic generation of an implementation from a detailed specification using a software tool.

After an implementation is available, an on-line phase begins, where the feature is not yet publicly available. In a testbed, careful and methodical *testing* takes place. If the tests are passed, the feature will be *deployed* into the network. With the *provisioning* of the feature, it is available to the network users, and they may now *register* with the feature. Usually, a feature like the call-forwarding supplementary service has then to be *activated* explicitly by the user. It is then *invoked,* e. g., by an incoming call. The *course of execution* denotes the phase in which the feature is actually being executed. *Deactivation* of a service by a user, *erasure* of the users registration data, and *withdrawal* of a service from the network seem to be unimportant to the feature interaction problem on the first sight, but have significant consequences, if, e. g., resolution is carried out restrictively. *Feature removal* has to be considered as an important phase if integration or conscious cooperation has been applied in order to resolve feature interactions.

2.4 System Context

- Telephone systems (narrow band)
  - Conventional switching systems (including N-ISDN)
  - IN/AIN structured systems
  - PABX systems
- Broadband systems
  - New approaches / Future systems (TINA, “Touring Machine”)
  - Broadband ISDN systems

Table 4: Classification by system context

Most work in the context of feature interactions has been done for *telephone systems (narrow band),* which include *conventional telephone switching systems* in general and *narrow band ISDN systems.* Very important is the problem of feature interactions for *IN/AIN structured systems.* Private *PABX systems* are also an issue. If not explicitly stated otherwise, the approaches discussed here apply to IN/AIN structured systems, sometimes with inclusion of systems with traditional switch-based services.

Much less work aims at *broadband systems.* Nevertheless, two streams can be separated: *New approaches* on future systems like TINA (Telecommunications Information Networking Architecture) or the TOURING MACHINE [26], and *broadband ISDN systems,* studied, e. g., by Magill *et al.* [27] and Tsang *et al.* [28], who report results of the impact of the feature interaction problem to networked multimedia services.

The generality of this problem type makes it likely for this scheme to require extension in the future, possibly also to non-telecommunications areas.

2.5 Relation to the emergence level view

In this section, we relate the emergence level view of interactions to the classification of approaches outlined above and to the survey of service interaction research in the remainder of this contribution. Detection of interactions does not contribute directly to the reduction of the number of interaction emerging at each level, i. e. the width of the arrows in Fig. 1. Resolution and prevention contribute in different ways: resolution requires the presence of interactions in order to take countermeasures, while prevention removes the source for their emergence. If the sources are addressed, the risk of further interactions with the introduction of new services is reduced; therefore, prevention is more powerful than resolution.

The approaches described in the remainder of this contribution only address interactions emerging from the logical and abstract architectural level. This is probably caused by the fact, that the concrete architecture of communications switching systems is a huge evolving legacy system, and new features are introduced rather by adding small pieces than by drastically re-engineering larger parts of the system. This observation explains the reluctance of researchers to address the problem at such a low level, however also points to the necessity of creating coherent frameworks and architectures and to apply them throughout all abstraction layers of the emergence level view.

In the following four sections, the research of the last decade in this field is presented briefly, structured according to the criteria listed in Table 1 and supplemented where suitable by the methodological criteria of Table 2.

3 Detection

3.1 Detection of interactions

In this section, approaches that detect interactions irrespective of their desirableness or undesirability are addressed. The counterpart, the detection of interference, i. e., undesired interactions, can be found in Section 3.2.

3.1.1 Formal techniques

Formal techniques are a suitable means to address the development of distributed systems, as they allow the choice of a suitable abstraction level and, with proper tool support, the verification of a system's correctness, at least with respect to a number of criteria. A large amount of work addressing the feature interaction problem relies on the application of formal description techniques for the specification of the system and its features in combination with formal verification. In [29], Velthuijsen reviews several formal method approaches and discusses the problems comprised. Bredereke [30] gives a detailed survey on this kind of approaches.

The classical way to use formal techniques in order to ensure the correctness of a system, is to create a functional specification of the system and its components using a formal description technique and to verify this specification using verification techniques, mostly model checking. An important precondition for the successful application of formal techniques is the availability of a suitable formal model for the system. There is much research with focus on the creation of formal system and service models, like [31, 32, 33] for conventional and Intelligent Network services, and [34, 35] for object oriented systems and architectures. Compared to the approaches described in the following paragraphs, that publications do not focus on interaction detection.

Verification using general criteria. Interactions are detected checking general correctness criteria like presence of deadlock and lifelock, transitions to invalid states, ambiguities, etc. This kind of approaches are termed *general-property approaches* by Bredereke [30, 36].

There are a number of process algebraic approaches using the standardised formal description technique LOTOS [37] or similar languages. Stepien and Logrippo [38] detect a category of interactions which manifest itself in ambiguous actions, i. e., non-deterministic behaviour. LOTOS is used as a formal description technique. In order to avoid the state explosion problem, a necessary condition for these ambiguities is established; the system is verified using backward execution of the specification in order to show if these ambiguities are reachable.

Based on the assumption that architectural weaknesses are the major source for feature interactions, Turner [39] proposes a description language called ANISE (Architectural Notions In Service Engineering) supporting the modelling of network and service behaviour on a user level. It is semantically closely related to LOTOS and applies notions and mechanisms of OSI services to the specification of telecommunications services. A precise description of features supports the detection of interactions by enabling the highlighting of potential areas for their occurrence. However, a detection method itself is not described in that paper.

Due to their sound theoretical foundations, Petri nets are considered as a suitable means to ensure a distributed system's correctness by Choi *et al.* [40]. They model an IN on the Distributed Functional Plane abstraction level using Petri nets for the description of the call state models and for the features as extensions of these models. Interactions are detected using reachability analysis, invariants, deadlock and trap detection, and net simulation. A tool prototype is presented.

Inoue *et al.* [41, 42] detect interactions which manifest themselves by non-deterministic state transitions, by the absence of an executable state transition (deadlock), and by transitions to so called "abnormal states", i. e., logical contradictions in the relationship between the state of terminals, using the State Transition Rules (STR) description technique. Interactions are resolved by adding rules which select one of the non-deterministic transitions or introduce a new state transition respectively.

Harada *et al.* [43, 44] specify a telephone system and its features using also the STR description technique. Conflicting pairs of rules, i. e., rules triggered by the same event, are identified using a kind of reachability analysis and applying a number of constraints to the identified pairs afterwards, resulting in a list of critical scenarios together with the conflicting rules for these scenarios that have to be evaluated manually by the service designer. In [45], Ohta and Harada show how several categories of interactions (described in an earlier section in that paper) can be identified using the mapping onto their finite state machine manifestation. Resolution within the specification stage takes place using priorisation, additional states, and correction of terminology. Kawarasaki and Ohta [46] extend this detection method by addressing the problem of transitions in the network that occur invisible to a user, leading to a difference between user knowledge of the network state and the actual network state, and propose a verification algorithm for this problem. Additionally, the problems of retries due to user input errors, and the problem that different actions appear synonymously in interaction detection are addressed. On the finite state machine level, the problem of illegal and lost transitions is addressed. They present a method of avoiding the state-explosion problem arising when reachability analysis is applied based on coloured Petri nets.

Nakamura *et al.* [47] demonstrate, how the P-invariant of a Petri net model, which presents a necessary condition of the reachability of a state, can be used to circumvent the state space explosion problem, that usually occurs if exhaustive state enumeration techniques are applied, as, e. g., in [44]. A case study demonstrates, that the resulting quality is comparable to conventional techniques, while the computational effort is smaller by several orders of magnitude.

Brothers *et al.* [48] describe the FIDO tool used to detect interactions based on a real-time transition model [49] of the services and features. The transition and constraint rules for these descriptions are extracted from the service logic programs (SLPs) of the Intelligent Network semi-automatically.

Bredereke [50, 30] presents an automata theoretic formalisation of the feature interaction problem. He introduces a specification style that is based on the principles of modification on a coarse-grained level and addition of transitions as the only allowed change of a specification (deletions are not allowed). Based on these principles, the term of feature interaction is formally defined, and necessary conditions for interactions are presented. The detection technique is implemented in the tool CONFINE based on the formal description technique ESTELLE [50, 51]. A resolution mechanism is sketched, which is based on the introduction of precedence relations between transitions and on the specification of additional behaviour in order to resolve the conflict.

Klein *et al.* [52] introduce a specification technique for features based on State Transition Diagrams (STD), where a feature is seen as a refinement of such a STD, and a feature interaction can be defined formally as a situation where it is not possible to find a common refinement for a system incorporating both (interacting) features.

Khoumsi [53] shows how interactions can be detected in systems specified in the form of extended finite state machines (EFSM) with guarded transitions and initial values for variables. The criteria for interactions are based on invariants involving variables local to each of the two participating services, as well as some static properties like deadlock, lifelock, and non-deterministic situations. Once identified, resolution of interactions takes place using conscious cooperation.

Thistle *et al.* [54] apply an automata theoretic approach called supervisory control theory to the problem. In this approach, modularity of design is emphasised. Events are divided into the classes "observable" and "controllable". A service is modelled in this framework as a supervisor which has the ability to disable controllable events on the basis of the past sequence of observable events. Interactions are detected using the languages (i. e., event sequences) expressed by the automata. A method to resolve interactions is sketched.

Chen *et al.* [55] also use supervisory control theory. Interactions can be detected by reachability of previously defined critical states or non-satisfaction of a behaviour represented by a minimal language, or by detection of blocking behaviour. Resolution takes place by a priorisation of different features controlling the same basic system; if a feature is overridden by another one, it follows the system's behaviour passively and is able to resume its operation at a later point in time.

Dssouli *et al.* [56] use scenarios for specification of features and derive finite state machines (timed automata) automatically from these scenarios. Interaction detection is carried out identifying operational inconsistencies (i. e., non-deterministic transitions), temporal inconsistencies, and inconsistencies against invariants.

Iraqi and Erradi [57] apply an object-oriented formal description technique called MONDEL to the feature interaction problems. Detection of interactions is carried out by generating the product automaton out of the specification and applying several automata-theoretic criteria like non-determinism. It is also outlined how feature interactions can be resolved: by mutual exclusion in the case of ambiguities, by hiding of events from the other features, or by establishing a protocol between the conflicting features.

Verification using problem specific criteria. A number of approaches extend the general correctness criteria by own problem specific criteria for the presence of interactions, like, e. g., access conflicts to system resources or read/write conflicts to system-wide variables. However, these criteria only point to the presence of interactions, they do not necessarily express that an interaction is unwanted.

Braithwaite and Atlee [58] present a call processing model which is based on layered state machines. Features are modelled as additional layers, which are able to pass through, consume, or modify events. This architecture implicitly resolves interactions imposed by non-determinism through priorisation, which is described in greater detail by Pomakis and Atlee [59]. Interactions of the types of result conflicts, resource contention, invalidation of information needed by other features, and assertions raised by other features are detected. A tabular description technique and reachability analysis for the verification of the specifications are applied. Au and Atlee [60] evaluate the approach described in [58, 59] by a case study with respect to the Bellcore benchmark [12, 13].

Dworack [61] develops a specification framework in order to specify service features nearly independently from each other, using a finite state machine approach. He identifies several manifestations of feature interactions in his approach.

Frappier *et al.* [62] apply relational specifications that map an input history of events (non-deterministically) to an output space, as a description technique for features. An interaction occurs if a number of features, although defined for a specific input history, cannot agree on one common output. It is shown how the method can be applied in practice using the Prolog language.

Lee [63, 64] uses the formal specification language Object-Z to describe basic call processing and features. Interactions are identified by analysing the specification and finding out if there exist state variables which are manipulated (read/written) by several features and thus are the common state being affected and a potential source for interactions.

Satisfiability approaches In some approaches, the system is described logically, usually using temporal, predicate or modal logic. Using this description, it is checked whether service requirements are consistent or contradictive. Capellmann *et al.* [65] uses the term *satisfiability approach* for this kind of methods.

Bergstra and Bouma [66] describe a process algebraic method for the detection of interactions using so called interworkings, i. e., a description technique similar to Message Sequence Charts (MSC), but synchronous, and timed frames, a kind of labelled transition systems including a notion for time steps. Interactions are detected on the level of logical inconsistencies within the model.

Blom *et al.* [67] use first-order linear-time temporal logic for the description of the basic service and the features. This specification technique is described more detailed in [68]. The description can be mapped automatically onto finite state machines. Interactions can be identified if they are based on conflicts in the logical formula denoting the invocation state, or on the effects caused by an event. In their framework, reachability analysis has to be applied in order to verify whether the conflicts are reachable and thus result in interactions. In [69], the framework TUSSILAGO for the specification of telephone services is presented. The interaction detection approach is based on the previously described method [67], but the modelling technique is enhanced using Message Sequence Charts and a modular specification style.

Rochefort and Hoover [70] show the application of a constructive proof system, i. e., a system where a satisfiability proof and the generation of an executable model are combined, to the feature interaction problem.

3.1.2 Informal methods

Artificial intelligence and natural language processing seem to be promising approaches for detection of interactions on the requirement level. Dankel *et al.* [71] propose a requirements capturing system based on artificial intelligence which is able to parse natural language and to convert it to predicates. With the aid of these predicates, ambiguity can be removed from the specification and a set of potential interactions can be generated, which has to be examined manually by the designer.

Charnois [72] uses natural language processing to identify feature interactions in textual requirement descriptions. Interactions manifest themselves in the form of semantic anomalies.

Methods of object oriented software engineering are another means to cope with the interaction problem. Kimbler and Søbirk [73] show the application of use cases to the feature interaction problem. Use cases on the network level are established, and by their analysis, possible feature combinations are examined using the simple but effective criterion that two features are interaction-prone if they access the same service or call specific data. The interaction-prone scenarios have to be analysed manually.

Kimbler *et al.* [74] report results from the PEIN (Pan-European IN) project consortium supervised by EURESCOM. In this approach, the features are first categorised with respect to the roles they play and resources they use in order to identify interaction-prone combination by possible conflicts of their categories. Features are analysed in a service context using a structured approach. For features considered as interaction-prone, a detailed investigation has to be performed manually.

3.2 Detection of interference

3.2.1 Formal techniques

The detection of interference requires a notion for the desirableness or undesirability of an interaction. A number of approaches achieve this by using two different specifications, a behavioural one to express the behaviour of the system and a property-oriented specification expressing the properties to be fulfilled by the specification. These specifications not necessarily have to be expressed in different languages. This kind of approach is termed *specified-property approach* by Bredereke [30], or *satisfaction-on-a-model* by Capellmann *et al.* [65]. There are also approaches that verify the satisfaction of general, system-wide criteria in order to guarantee the absence of interference.

Approaches using two different specification techniques. A number of approaches using different specification languages for the behaviour and property parts are presented in the following sections.

Thomas [75] describes a model of a telephone network on the user level formalised using LOTOS. Its properties are described using temporal logic. She demonstrates how feature interferences can be identified using model checking for temporal properties, however, she has reached the limits of the used Cæsar/Aldébaran toolkit quite soon. Other problems, like non-determinacies, are discovered by static analysis, e. g., by finding overlapping guards.

Capellmann *et al.* [76, 77, 65] describe a method to detect interference using a description of the basic services and service features on the distributed functional plane of the IN standards using Product Nets, a high level variant of Petri Nets. Detection of interference takes place by checking separate temporal properties of the behaviour. In order to avoid the state space explosion problem, they use abstractions based on homomorphisms, thus reducing the complexity of the specification.

Aggoun and Combes [78] present an approach for interference detection and resolution based on the concept of observers. These observers are applied at two stages during the service lifecycle. A passive observer, based on a description as a finite automaton with success and error states, is applied at the network-level specification stage, where a formal model in SDL (Specification and Description Language of the ITU-T [79]) is validated using exhaustive or guided simulation. Renard *et al.* [80] describes the tool support to carry out these steps, and the results of a benchmark evaluation. Interferences are detected by a success and error state mechanism, but no actions are taken. Active observers are applied in the course of execution and are able to trigger the resolution of interferences.

Bouma *et al.* [81] describe an approach taken in the RACE project SCORE (Service Creation in an Object oriented Reuse Environment) to the detection of interference using a property language (temporal logic) and a descriptive language (SDL). Combes and Pickin [82] show how a property language based on linear time temporal logic can be applied to a behaviour specification in a constructive language. The level of abstraction is the Global Functional Plane of the IN conceptual model. In [21], Combes *et al.* describe the formalisation and verification method in more detail, and they present experimental results of finding interferences in a number of well-known IN services. They conclude, that a number of formal techniques have to be used in a coherent way to detect interferences, a static analyser for interactions due to non-determinism and datasharing, a behavioural tool to verify temporal logic requirements, and a simulation tool to carry out a heuristic evaluation of the system.

In their work on provably correct telecommunications systems (ProCoS), Fischer *et al.* [83] and Kleuker [84] address the feature interaction problem with a method that allows to derive implementations from formal specifications using transformations that keep correctness properties. It is demonstrated how features can be added incrementally starting from a correct initial model.

Single specification approaches. In some approaches, the properties that have to be fulfilled by a behavioural specification are expressed in the same language, thus obtaining only one specification document containing properties and behaviour.

Korver [85] applies the formal description technique LOTOS in order to describe Intelligent Network service processing in the Global Functional Plane. In order to check for the fulfilment of specific properties, testers are applied, i. e., processes that encode the property to be checked and run in parallel to the specification. Verification is done using reachability analysis and the minimisation of the generated labelled transition system.

Faci [86] presents a method for the specification of telecommunications services in LOTOS. Interaction detection is carried out exploiting the testing theory for this language by generating test cases automatically from the formal specifications of the services. A necessary precondition is that both features have got disjoint signalling alphabets. Although a technique like test case generation and application is used, this method has to be regarded as a formal verification approach.

A knowledge goal oriented approach is described by Faci and Logrippo in [87]. The features are specified independently and integrated; afterwards, a set of knowledge goals for both features is defined by the designer (using a parallel LOTOS process) and it is verified whether these goals are reachable; if not, a feature interference (or a design error) is present.

The approach of Boumezbeur and Logrippo [88] is based on a constraint-oriented LOTOS specification of a telephone system's observable behaviour. Each telephone feature is specified as an individual process in order to facilitate modularity and maintainability of the specification. Interferences are identified by execution of the specification within an interpretative environment. Symbolic execution trees are derived using test sequences in the form of LOTOS processes, and these trees are checked manually for unwanted situations. In an outlook, the authors point out, how this manual checking could be improved by techniques like causal constraint analysis, complete execution tree calculation, random walk validation, and goal-oriented execution.

Stepien and Logrippo [89] use abstract data types of specification languages like LOTOS, SDL and Prolog in order to represent a notion for the violation of a features intention. The abstract data type, which is defined for each feature independently (with the important limitation that all possible actions in the whole system have to be respected), is applied in an executable specification; interference is identified using either goal-oriented or backward execution, where the goal respectively the starting point is the violation of an intention. Problems with combinatorial explosion are reported, which can be reduced giving additional directives to the tool.

Lin and Lin [90] model service features in the Intelligent Network on the Distributed Functional Plane level of abstraction using PROMELA [91]. Both properties and behaviour are specified in this language, and state-space exploration (using the SPIN tool) is used in order to verify the properties. For the management of the large number of scenarios they use a front-end tool in order to select features and scenarios from a repository automatically. In an appendix, parts of the specification and the assertions are listed.

Gammelgaard and Kristensen [92] present a description based on transition rules and predicates (first order logic). Features are an extension or modification of a basic service, interference becomes manifest as conflicts between the predicates that a certain state has to satisfy. Their work has been carried out in the SCORE project.

Boström and Engstedt [93] present a method based on a tool using predicate logic and a elaborate description technique (the DELPHI framework). Interference is detected, if the specification is not consistent and if properties specified for an individual feature are not fulfilled by the joint specification. This is shown using propositional logic and a theorem prover. In order to resolve the interactions, so called "interaction features" are introduced, which impose priority mechanisms between the conflicting features.

Global property approaches. A third kind of interference detection approach uses properties which are not stated explicitly for each feature, but are defined system-wide and have to be fulfilled by each individual feature.

Dahl and Najm [94] use the formal description technique LOTOS in order to model call and service processing in the Intelligent Network. They present a strategy to enhance the basic call service by a new service feature, leading to different specifications for each feature, and then to combine the two specifications into one. Interference is revealed as differences of the service state, data types, and of the invocation of the feature.

Gibson [95] demonstrates, using a number of examples, how several feature interferences are introduced artificially on the requirement modelling level by the use of imprecise or improper models. He concludes in a number of recommendations for the establishment of a requirements specification, like, e. g., strong typing, object orientation, invariants, avoidance of ambiguities in naming of actions, explicit appearance of features, arbitration, choice of modelling language, operational requirements, incremental development, and restrictive assumptions.

3.2.2 Experimental techniques

Among the experimental techniques are simulation of network environments and testing in real and simulated environments. As all the approaches herein have a notion of desirableness, they fall into the interference detection category.

Kelly *et al.* [96, 97] present an approach using simulation of feature behaviour on the network level (for Centrex services) and on the service plane (for IN services), using a SDL-based animation tool.

Tsang and Magill [98] propose a feature management system to detect interferences that does not require knowledge of the features. As a reference, a feature manager (FM) records the sequence of the states of a feature and its used resources which specifies the correct operation. If this sequence is not matched during the feature's operation, a report is generated. Despite of its similarities to [78], this approach is rated experimental, as the correct behaviour of a feature is recorded from a reference implementation.

Godskesen [99] formally defines the meaning of the term interference at the requirements, the specification and the implementation level. He describes formally, how tests have to be applied in order to test for the absence of interference on the implementation level using tests for all the implementations of the individual features.

3.2.3 Informal methods

Mierop *et al.* [7] present results from the European Community research project ROSA (RACE Open Services Architecture) [100]. A separation of service-induced and resource-induced interference is introduced. The detection method presented is based on service-induced interference

(e. g., ambiguous situations) only. These situations are detected during the object-oriented specification of the service, which is explained by the example of *Call Forwarding on Busy* and *Call Waiting,* where the interference is identified by the ambiguity created by the busy signal.

Informal criteria to describe the execution of IN services are used by Kuisch *et al.* [101]. They present a detection method for interactions between IN services based on templates, in which the functionality of a feature is expressed through the usage of Detection Points of the Basic Call State Models, data (information flows) and resources. This method is intended as a direct aid for service developers and is especially suitable for a (relatively straightforward) tool implementation.

Wakahara *et al.* [102] propose a detection method based on "input-output relationships" between service features. The features are specified informally, and the investigations take place using message sequence charts. The interactions are classified using the criteria duplication, redundancy, (incorrect) order of execution, inconsistency, vagueness or non-determinism, and looping. For each of these criteria, an informal detection procedure is sketched.

4 Resolution

Resolution of feature interactions that have been discovered during a previous detection process or that have already been identified in a different way, is a necessary consequence for mastering the problem. A number of approaches referenced in the following sections have already been discussed in the previous section on feature interaction detection, as they introduce a method for detection as well as a proposal for resolution. However, in the following sections, only their resolution aspects are treated. It must be noted, that in the resolution category, the distinction between interaction and interference is not relevant. Thus, the term interaction is used throughout this section.

4.1 Restriction

Historically, restriction is the classical method to be applied if feature interactions are likely to occur. In the ITU-T recommendations for ISDN supplementary services [103] for example, for each supplementary service there exists a section on the interactions with other services. The usual treatment of interactions there is the introduction of precedence or mutual exclusion rules, which obviously fall into the restriction category.

4.1.1 General restriction

General restriction is a rather rigid method which is very often chosen by practical approaches. Homayoon and Singh [104] study the problems of interactions between switching system based and IN based features, and present a complex feature arbitration policy using rules for the activation and invocation of features. The technical consequences are investigated, paying special respect to the feature processing environment (including an arbitrator) and the interfaces needed between the different functional entities (switches, Service Control Points).

Schessel [105] presents an approach towards the resolution of feature interactions between switch-based features using tables that contain information on the priority of one feature over another one, on mutual exclusion of two features, and on the current feature status for a specific call. The content of the tables is provided by the feature developers and can be changed and edited by the administration.

4.1.2 Situation dependent restriction

Situation dependent restriction is more flexible than general restriction, as it allows to decide whether to carry out a specific operation dependent on the situation encountered by the conflicting services. In contrast to general restriction, it does not affect the ability to activate and invoke the conflicting features.

Specification approaches. A number of contributions address the resolution of interactions in a restrictive way on the specification level, mostly using relatively simple precedence mechanisms.

Khoumsi [53] resolves interactions which are detected in extended finite state machines using general criteria by an algorithm with three steps. First, all undesired states in the compound description are removed; secondly, an acceptable and realisable behaviour has to be specified where necessary and finally, a common additional automaton (supervisor) is added to the conflicting services in order to achieve the desired joint behaviour by introduction of precedence and exclusion relations.

In Chen's approach [55], supervisory control with priorities is employed in order to resolve interactions detected by a method described in the same paper. An individual supervisor has got the possibility to override one with lower priority, thus resolving conflicts by restricting the control to one supervisor at a time. The other supervisors follow passively the behaviour of the system.

A similar approach is taken by Boström and Engstedt [93], who introduce so called interaction features which impose priorities between the conflicting parts of the interacting features.

Blom *et al.* [68] add an interaction handler in the form of a temporal logic term to the combination of the conflicting features. They achieve resolution of the interaction by mutual exclusion or priorisation of the conflicting parts.

Precedences between transitions are used in the resolution method sketched by Bredereke [50], however, in cases where this is not sufficient, additional behaviour has to be specified. This method is described in more detail in a technical report [106], where ESTELLE is used as a specification language.

Cheng [107] uses a LOTOS specification on the Distributed Functional Plane level of the IN Conceptual Model in order to resolve feature interactions caused by the features' invocation and/or termination due to the same event (detection point). He uses priorities and invocation precedence rules, represented by constraints on the feature specification.

Mierop *et al.* [7] use an agent based object oriented approach in order to control the problems. Interference, once detected by an informal method described in the paper, is resolved using priorisation in the user profile or by the user agent using an user-independent predefined policy.

Feature interaction managers. The ITU-T recommendation on Intelligent Networks [108] introduces the notion of a Feature Interaction Manager in the sense of an arbitrator between IN and switch based services. The resolution approaches discussed in this section proceed beyond this notion, as they use a FIM in order to resolve interactions.

The basic idea of Cain [109] is the introduction of an entity between the features and the basic call processing models that filters the signals to and from the features in order to resolve interactions. The motivation is to keep the implementation of the service features separate from the knowledge on feature interactions, and to concentrate this knowledge into one functional entity. Three different kinds of interaction managers are described briefly. A "two-pass" model issues a request (i. e., a signal) to all features and collects all responses, then decides which action to take. The features are notified by a "commit" message, if their action has been accepted, and with a "rollback" message otherwise. A "side-effect-free" model collects the responses from all features, and afterwards decides, which are acceptable and which are not. Therefore, the side effects of all features have to be buffered. An “error-free” model, where for each possible interaction situation a method is kept how to resolve this case, is explained in more detail in Cain’s work, as it is considered as the only one which is really feasible.

Fritsche [110, 111] presents a method to resolve feature interactions using a Feature Interaction Manager with a matrix holding information under which conditions a participant has the possibility to accept being involved in a newly invoked feature (i. e., taking a new role). Using the information in this matrix, it is decided, whether the feature may be activated, blocked, which changes have to be made, or which additional measures (like display messages) have to be taken. The information in this matrix has to be provided by the feature developer, which is not subject of that paper. The author presents a prototypical system realised with a SDL toolset in order to validate his concepts.

Ogino [112] describes the resolution of interactions between services that are realised within the same service node. This concentration of different services into one single network entity eases the treatment of interactions significantly. The resolution is based on a table where different interaction categories for feature combinations are noted and measures are taken according to these categories. Thus, the major effort of the proposed method lies in the construction of this table.

Active observers in the approach of Aggoun and Combes [78] are applied in order to monitor specific INAP (Intelligent Network Application Protocol [113]) messages. They have the ability to decide whether an action is desired or undesired, and cooperate with a solver that implements mechanisms in order to solve several kinds of interactions detected by the active observers.

Tsang and Magill [114] extend their detection method presented in [98] by resolution aspects. Interactions are detected using one entity of a feature interaction manager (FIM) for each subscriber. In a *learning mode,* the correct behaviour of a feature is entered into the FIM by observing the stream of messages between the basic call state models and the service processes. During operations, in a *management mode,* the actual messages are compared with the database of learned messages and an interaction is detected if there is an inconsistency between these. It is possible to resolve some conflicts in their approach. Trigger conflicts (shared triggers) are resolved by a policy preferring the trigger with the least interactions and the highest priority, more severe problems are solved using the philosophy of damage limitation by ending the call in a controlled manner. The authors claim that their approach is more viable to the problem solution than any other approach, as it requires very few modifications to the IN infrastructure.

Makarewitch [115, 116] reports partial results of two EURESCOM projects. The focus is on the runtime resolution of feature interactions using a feature interaction manager, divided into different components, each of which is specialised to one specific area of concern, i. e. charging, connection, data analysis, user interaction. These processes are placed in the Service Control Function (SCF) between the service logic process instances and the IN environment (SSF, SRF, SDF). They are able to process stimuli from the environment in order to give correct signals to the service logic process instances (SLPI), and to filter the responses from the SLPIs in order to give consistent signals to the environment. The functionality of these entities depend on the services implemented in the system, i. e., they may have to be changed for each new service. The entities are situated in the same SCF, and are unable to work correctly if the different services are executed in different SCFs. The services considered as examples are simple originating services, Freephone and Premium-Rate, Credit-Card-Calling, and Virtual Private Networks.

Practical methods. Marples *et al.* [117] describe an experimental testbed, which is able to emulate IN-based services as well as legacy services on a simulated software infrastructure. It is suitable to examine services for any kind of interactions, and can be used to implement and validate prototypical extensions, like, e. g., feature interaction managers.

4.2 Integration

Especially on the specification level it is not always easy to decide whether an approach belongs to the restriction or integration category, as the addition of special treatment of the interaction case often has got integrative as well as situation specific restrictive aspects. The criterion to decide this is the possibility to clearly assign the parts of the specification to one of both features.

In their State Transition Rules (STR) approach [45], Ohta and Harada generate new state transition rules which give priorities to features and resolve transitions into undefined states.

Inoue *et al.* [41] resolve interactions by adding rules which select one of the nondeterministic transitions detected previously or introduce a new state transition respectively, in cases where no executable state transitions are present.

Kakuda *et al.* [118, 119] also use state transition rules for service specification. They resolve feature interactions using a protocol synthesis technique. Interactions manifest itself as non-determinism in the feature specification, which is resolved inserting conditional branch functions into the protocol specification, thereby determining which sequence is selected.

4.3 Cooperation

Cooperation between features is the most sophisticated concept to resolve interactions. Cooperation goes beyond the simple decision taken in restriction approaches whether to carry out an operation or not; it is tried to find a solution or compromise which satisfies the needs of all participating features.

4.3.1 Conscious cooperation

In conscious cooperation, the cooperation mechanisms are designed aware of the purpose and implementation details of the conflicting features.

Iraqi and Erradi [57] sketch three strategies to resolve interactions between features using a composition by choosing one of the features involved, by hiding the offending events from one another (both of these strategies are restrictive), or by establishing a protocol between the features in order to find an acceptable solution by cooperation between features situated at different physical locations.

Prehofer [120] presents an approach for implementing features in an object-oriented environment which allows the treatment of feature interactions as part of the design methodology, “feature oriented programming.” This is achieved by adaptation of the services of one features to the context of another one, and by an implicit ordering of the features. With this procedure, the correctness of the functionality of one feature in the context of another feature is maintained.

Utas [121, 122] discusses the realisation of switch based and Intelligent Network features in a practical system. He presents the means that are provided by the software architecture of a real system in order to resolve feature interactions. Among the mechanisms used to achieve cooperation is a cooperative building of signalling messages. There are, however, tight limitations in real systems, imposed, e. g., by the capabilities of signalling protocols.

4.3.2 Oblivious cooperation

Oblivious cooperation is among the most promising long-term solution concepts to achieve resolution of interactions. Features or services cooperate, although they do not have any special knowledge of each other, thereby supporting one of the goals of new service-centered architectures like the Intelligent Network, the independent development of new services.

Velthuijsen [123] evaluates the suitability of distributed artificial intelligence methods comparing four different DAI approaches to feature interactions. Three of the approaches deal with network management systems, where interaction problems similar to subscriber feature interactions can be observed. The approaches address problems of data availability, different administrative domains, network resource contention and problems caused by individual instance-specific feature parameters. As cooperative problem solving is based on peer-to-peer cooperation, it is especially suitable for today's de-regulated telecommunications market, because no single trusted entity is needed. Thus, it turns out to be a promising approach in order to solve interaction problems, but there remain many open research problems, like the design of evaluation functions or the interference of different problem solving mechanisms.

Griffeth and Velthuijsen [124] use negotiating agents (indirect negotiation) in order to detect interactions by identification of conflicts in a goal hierarchy and resolve them by exchanging proposals and counter-proposals and finally agreeing on a goal which is acceptable for all involved entities. This approach is validated on an experimental platform called TOURING MACHINE [26]. This is a distributed object oriented experimental architecture for the provisioning of multimedia services. It enables the realisation of intelligent services which require knowledge of both static and dynamic information about the system (provided by a special repository called "Name Server"). Additionally, it is shown, how this approach can be applied to AIN/IN features. In [125], the conflict resolution method used in [124] is described in more detail. It respects the privacy of the agents and the finding of alternative activities to achieve the goals of all involved agents, which are expressed in a goal hierarchy. In cases where no common acceptable proposal can be found, one of the agents has to relax its constraints.

Dini and Bochman [126] study the problem of feature interactions in the context of automatic reconfiguration management. They map the terms of services and features, interaction and interference onto the object-oriented terminology. They extend the feature interaction problem to the problem of the substitutability of services, the optimisation of existing (desired) feature interactions, and the automatic adaptation to (desired) changes of feature interactions.

5 Prevention

Prevention complements detection and resolution of feature interactions. By structural means, i. e., by development and application of suitable system design and architecture, interactions of some kinds, like, e. g., resource conflicts, can be avoided. Procedural approaches enforce a well defined and strict design methodology that does not allow interaction prone sequences of actions to be used.

5.1 Structural prevention

This class of prevention approaches is characterised by a system structure that has got means to avoid feature interactions.

Van der Linden [127] proposes the application of ODP (Open Distributed Processing) principles in order to avoid unwanted feature interactions. Central to his approach are the terms of *separation* of the system entities (features) and *substitutability* enabled by a well-defined external interface. Each feature is described using a framework of statements about purpose, meaning, structure, quality and technology, leading to a possibility to avoid interactions by transparency.

Cattrall *et al.* [128] propose an architectural model which avoids a number of feature interactions by resolving its causes. It is based on a division between roles fulfilled by different users and the terminal equipment they use.

Zibman *et al.* [129, 130] propose an agent architecture that is able to avoid feature interactions by a proper separation of concerns within a telecommunications system. This is achieved using agents that have got dispatch rules and precedence rules that are independent from specific services, but based on signals used commonly in the system. They claim that their approach handles all interactions listed in [13], and demonstrate it using several examples from that paper. Finally, they give an outlook how their approach could be mapped on multimedia and IN network infrastructure.

Weiss *et al.* [131] describe an agent-based system for distributed multimedia applications, that minimises feature interactions by design. This is achieved by decoupling (i. e., the separation of policies and agent implementations), by off-line negotiation of service guarantees, by multiple levels of abstraction for the expression of service requirements, and by a learning mechanism that deals with changes to the environment.

Keck [132] analyses interaction emergence in today's Intelligent Networks and focuses on interactions that are caused by insufficient information and mechanisms provided by the network. In order to prevent these problems, he proposes a Session Configuration Profile (SessCoP) that is able to provide the missing (additional) information and mechanisms to the interacting services, thus giving them the opportunity to react adequately.

5.2 Procedural prevention

To the knowledge of the authors, there is no documented approach that can be placed in the category of procedural prevention. Such an approach would be characterised by a very strict and probably tool supported service definition, specification and implementation process, that leads to implementations that are free of interactions simply by following a set of rules. Such an approach would reduce the service interaction problem to the task of ensuring that a service is implemented correctly with regard to the rules of the prevention method. However, a number of approaches described in the previous section on structural prevention contain also aspects of procedural prevention, like, e. g., van der Linden's software structuring policies [127].

6 Management of the overall problem

During the previous three sections, methods that deal directly with the interaction problem have been discussed. It became clear that there probably will be no single method to cover all aspects of the problem at once. Bowen *et al.* [1] outline in their fundamental paper the many dimensions of the problem during the different design stages. Kung [133] points out a number of additional problems that occur in a competitive environment, where different service providers develop and offer their services in the same network.

The contributions in this section are divided into two groups: the first group is concerned with the management of the combinatorial complexity of the interaction problem, including tools for context generation and information acquisition, the second group comprises approaches concerning the whole interaction handling process.

The management of the combinatorial complexity is a research issue with respect to the applicability of specific methods. Kimbler [134] proposes a technique where feature combinations are selected for further evaluation depending of their occurrence probability and a quantification of the potential risk connected with a specific interaction scenario.

Keck [135] proposes criteria in order to reduce the large number of possible scenarios that can occur when two (or more) services are combined, based on the exploitation of symmetries and the structure of the trigger detection points used by the features.

Lin and Lin [90] present an automated environment for producing feature contexts, i. e., scenarios consisting of features and the appropriate call state models. Using these tool-generated descriptions, formal verification for a large number of interaction scenarios can be carried out automatically.

Blumenthal *et al.* [136] describe a tool to examine services for potential interactions using a knowledge base for known problem cases, including information on Bellcore requirements and actual implementations.

To master the problem as a whole, sophisticated coordination of a number of different methods and means to be employed, i. e., a well-defined interaction handling process, is required. In [137], Kimbler presents results of the EURESCOM Project P509 "Handling Service Interactions in the Service Life Cycle," where a service interaction handling process (SIHP) is defined. It consists of four phases. In the first phase, an analysis plan is established stating what feature combinations have to be analysed. A filtering step reduces in the second phase the number of scenarios in order to make the treatment possible. Afterwards, detection and finally resolution techniques make up the third and fourth steps. Georgatsos *et al.* [138] present requirements for system management that have been identified during this project. A more detailed overview on the results of the P509 project can be found in [25].

Tool support for the overall process of dealing with interactions is described by Cameron *et al.* [139], where an integrated toolset for service creation and feature interaction analysis and management is presented.

7 Trends and future directions

The service interaction problem as described in this contribution has two major aspects, which are equally important and require further consideration. In this section, these aspects and their implications on service interaction research are discussed briefly.

7.1 Telecommunications network specific issues

The discussion of the interaction problem has been triggered initially by the difficulties of integrating new telecommunication services into existing networks. This is the *practical aspect* of the problem. The steadily growing number of new telecommunications networks and services and the worldwide trend to de-regulated telecommunications markets imposes rising pressure onto service providers, network operators and equipment vendors to tackle this problem. In the current telecommunications environment, the following factors will impact the urgency of the problem significantly:

- *Heterogeneity of service provisioning.* In order to provide a new service, a number of options, like switch based, service node based, IN based, terminal equipment based or even Internet-/IP-Network based may be chosen. It is obvious that this heterogeneity exponentiates possible interactions, as well as it complicates to take countermeasures. However, economic boundary conditions dictate a protection of the large investments in the existing network infrastructure and require mechanisms for resolution and avoidance of interactions that can be smoothly integrated into existing infrastructures with only moderate changes or extensions of present systems. This aspect should also taken into account when the evolution of signalling in modern networks is discussed [140].

- *Omnipresence of supplementary services.* The growing penetration of the market with personal and terminal mobility services, number portability, voice and facsimile mailboxes, integration of Internet services (e. g., electronic mail notification, Internet shopping) marks a development towards each phone call being a supplementary-service enhanced call. In such a scenario, restrictive interaction resolution methods will soon reach their limits.

- *Network interconnection and interoperability; convergence of telecommunications and data networks.* Most services rely on information passed by signalling messages. Usually, this information is restricted if network boundaries are passed or the information cannot be evaluated outside the original network. This makes it difficult if not impossible for some services to work properly across network boundaries. However, especially in a de-regulated telecommunications market, many network boundaries exist. Thus, the establishment of standards for management of interactions between network operators or service providers is urgently required, as it is a significant and economically relevant question to decide which one of the conflicting services will have to be modified or removed if interference occurs.

- *Network security.* Service interactions are characterised by the fact that a service's behaviour is changed in the presence of other services. This may have severe consequences if this service is a security service, used for protection of privacy or anonymity, or for reliable transactions in electronic commerce.

7.2 Software and systems engineering issues

There is also a *fundamental aspect* of the problem. Typical for all large and evolving, possibly also distributed systems, is the stepwise addition of functionality. In order to achieve this, either knowledge of the whole system including all former modifications and additions is required, which is virtually impossible for large systems, or a concept or a methodology that allows the addition of functionality with a limited knowledge of the system as a whole. Research progress in the following areas with special respect to legacy systems may be able to provide solutions to these problems

- *Requirements engineering.* A significant number of service interactions are caused by the fact that the way the requirements are expressed does not reflect the concepts provided by a system. E. g., in telephone networks, the notion of a "caller" is used, even if the network supports conference calls, or the "called number" is used synonymously for the identity of the called person. This is especially difficult because such concepts may be enhanced gradually by adding new services.

- *Definition of coherent architectural frameworks and concepts.* Closely related to the former item, methods for the definition of a framework that is open for extension on the one side, but also complete in a sense that it is possible for a new extension to anticipate the possible impact of other extensions (which has to be limited by the system's well-defined capabilities for addition of functionality), are needed. Such a framework requires rigorous formal definition and well defined interfaces for extension. For telecommunications networks, the TINA initiative [141] is a step towards this direction.

- *FDTs and verification techniques.* The application of FDTs requires a formal model of the whole existing system and a proof or validation of its correctness, a enormous, if not impossible task for systems of realistic complexity. Even if systems are formally described at an abstraction level that is comparable to the level used in standardisation documents, which is actually much coarser grained as required for implementation, today's verification tools and methods rapidly approach their limits, as, e. g., reported in [75]. Here, despite of some "success stories" of tool application, still a lot of improvement is urgently needed.

8 Conclusions

During the classification of the large number of publications cited in this contribution, the classification scheme presented at the beginning has proven as very useful, as the majority of different approaches fit quite naturally into the framework. In some cases, it has been difficult to decide the category of an approach, because it contained aspects of several categories. Then, usually the category for the most sophisticated concept of the approach has been chosen. Nevertheless, it has been difficult in rare cases to decide whether a detection approach is formal or experimental, or a resolution method restrictive or integrative.

By reviewing the approaches dealing with the feature interaction problem, some interesting observations have been made. In the detection category, there is an overwhelming number of formal method approaches, using all kinds of methods, while there is very little support for real implementations, especially for dealing with legacy services. In the resolution area, there are promising approaches, but many of them rely on explicit mutual knowledge of both services, which is clearly a problem in a de-regulated, multi-provider environment. The prevention approaches are long term solutions that require a suitable network infrastructure. Here, almost no short term perspective is seen, especially if legacy services have to be covered. Management papers are still of modest number, although the complexity and size of the problem makes mastering this area a crucial issue.

This contribution documents that there is already a widespread and successful activity in the field of feature interactions. Nevertheless, many open issues remain. This paper focuses on the intricate nature of the feature interaction problem, explained using the *emergence level view,* and gives a structure to the large number of approaches, based on a refinement of the detection, resolution and prevention categories. It gives an overview to the beginner and a concise reference of the work of the recent years to the expert. Moreover, it wants to encourage and attract researchers from other fields to have a closer look at this important and relevant problem domain, that extends beyond the field of telecommunications systems.

References

[1] T. F. Bowen, F. S. Dworack, C. H. Chow, N. Griffeth, G. E. Herman, and Y. J. Lin, "The Feature Interaction Problem in Telecommunications Systems", in *Proceedings of the 7th International Conference on Software Engineering for Telecommunications Switching Systems*, London, July 1989, pp. 59–62.

[2] "Vocabulary of Terms for ISDNs", ITU-T Recommendation I.112, Geneva, 1989.

[3] "Principles of Telecommunication Services Supported by an ISDN and the Means to Describe Them", ITU-T Recommendation I.210, Geneva, 1989.

[4] "Advanced Intelligent Network (AIN) Release 1, Switching Systems Generic Requirements", Bellcore Technical Advisory TA-NWT-001123, May 1991.

[5] "Principles of Intelligent Network Architecture", ITU-T Recommendation Q.1201, Geneva, 1992.

[6] E. J. Cameron and H. Velthuijsen, "Feature Interactions in Telecommunications Systems", *IEEE Communications Magazine*, vol. 31, no. 8, pp. 18–23, August 1993.

[7] J. Mierop, S. Tax, and R. Janmaat, “Service Interaction in an Object-Oriented Environment”, *IEEE Communications Magazine*, vol. 31, no. 8, pp. 46–51, August 1993.

[8] R. L. Bennett and G. E. Policello II, “Switching Systems in the 21st Century”, *IEEE Communications Magazine*, vol. 31, no. 3, pp. 24–28, March 1993.

[9] I. Faynberg, L. R. Gabuzda, M. P. Kaplan, and N. J. Shah, *The Intelligent Network Standards: their application to services*, McGraw-Hill, New York, 1997.

[10] T. Magedanz and R. Popescu-Zeletin, *Intelligent Networks: Basic Technology, Standards and Evolution*, International Thomson Computer Press, London, 1996.

[11] P. Zave, “Secrets of call forwarding: A specification case study”, in *Proceedings of the IFIP TC6 Eigth International Conference on Formal Description Techniques: Montreal, Canada, October 1995*, G. v. Bochmann, R. Dssouli, and O. Rafiq, Eds., London, October 1995, pp. 169–184, Chapman & Hall.

[12] E. J. Cameron, N. Griffeth, Y. J. Lin, M. E. Nilson, W. K. Schnure, and H. Velthuijsen, “A Feature-Interaction Benchmark for IN and Beyond”, *IEEE Communications Magazine*, vol. 31, no. 3, pp. 64–69, March 1993.

[13] E. J. Cameron, N. D. Griffeth, Y. J. Lin, M. E. Nilson, W. K. Schnure, and H. Velthuijsen, “A Feature Interaction Benchmark for IN and Beyond”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 1–23, IOS Press.

[14] K. Kimbler and H. Velthuijsen, “Feature Interaction Benchmark”, in *Third Feature Interaction Workshop (FIW ’95)*, October 1995.

[15] N. D. Griffeth and Y. J. Lin, “Extending Telecommunications Systems: The Feature-Interaction Problem”, *IEEE Computer*, vol. 26, no. 8, pp. 14–18, August 1993.

[16] P. Zave, “Feature Interactions and Formal Specifications in Telecommunications”, *IEEE Computer*, vol. 26, no. 8, pp. 20–29, August 1993.

[17] N. Belarbi and D. Gaiti, “The Feature Interaction Problem in the IN: In search of a global solution”, in *Proceedings of the IFIP International Working Conference on Intelligent Networks*, Copenagen, August 1995, pp. 51–67.

[18] D. R. Kuhn, “Sources of Failure in the Public Switched Telephone Network”, *IEEE Computer*, vol. 30, no. 4, pp. 31–36, April 1997.

[19] A. V. Aho and N. D. Griffeth, “Feature Interactions in the Global Information Infrastructure”, in *Proceedings of the 1995 Conference on Foundations of Software Engineering*, 1995.

[20] K. E. Cheng and T. Ohta, Eds., *Feature Interactions in Telecommunications Systems, III*, Amsterdam, October 1995. IOS Press.

[21] P. Combes, M. Michel, and B. Renard, “Formal Verification of Telecommunication Service Interactions using SDL Methods and Tools”, in *Proceedings of the 6th SDL Forum 1993 (SDL ’93)*, O. Faergemand and A. Sarma, Eds. 1993, pp. 441–452, Elsevier Science Publishers.

[22] J. C. Gregoire and M. J. Ferguson, “Neglected Topics of Feature Interactions: Mechanisms, Architectures, Requirements”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 3–12, IOS Press.

[23] L. G. Bouma and H. Velthuijsen, Eds., *Feature Interactions in Telecommunications Systems*, Amsterdam, May 1994. IOS Press.

[24] "Intelligent Network (IN); Service and service feature interaction service creation, service management and service execution aspects", ETSI Technical Report ETR 137, ETSI TC-NA, Reference DTR/NA-061101, July 1995.

[25] K. Kimbler, C. Capellmann, and H. Velthuijsen, "Comprehensive Approach to Service Interaction Handling", *Computer Networks and ISDN Systems*, September 1998, to appear.

[26] M. Arango, L. Bahler, P. Bates, M. Cochinwala, D. Cohrs, R. Fish, G. Gopal, N. Griffeth, G. E. Herman, T. Hickey, K. C. Lee, W. E. Leland, C. Lowery, V. Mak, J. Patterson, L. Ruston, M. Segal, R. C. Sekar, M. P. Vecchi, A. Weinrib, and S. Y. Wuu, "The Touring Machine System", *Communications of the ACM*, vol. 36, no. 1, pp. 68–77, January 1993.

[27] E. H. Magill, S. Tsang, and B. Kelly, *The Feature Interaction Problem in Networked Multimedia Services: Past, Present and Future*, University of Strathclyde, Glasgow, October 1996.

[28] S. Tsang, E. H. Magill, and B. Kelly, "The feature interaction problem in networked multimedia services: present and future", *British Telecom Technology Journal*, vol. 15, no. 1, pp. 235–246, January 1997.

[29] H. Velthuijsen, "Issues of Non-Monotonicity in Feature-Interaction Detection", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 31–42, IOS Press.

[30] J. Bredereke, "Detection of Feature Interactions in Intelligent Networks by Verification", *Software-Concepts and Tools*, vol. 17, no. 3, pp. 121–139, 1996.

[31] J. Nyström and B. Jonsson, "A Formalization of Service Independent Building Blocks", in *Proceedings of the International Workshop on Advanced Intelligent Networks (AIN'96)*, T. Margaria, Ed., Passau, March 1996, pp. 1–14, Universität Passau, Fakultät für Mathematik und Informatik.

[32] A. Fekete, "Formal Models of Communication Services: A Case Study", *IEEE Computer*, vol. 26, no. 8, pp. 37–47, August 1993.

[33] M. Faci, L. Logrippo, and B. Stepien, "Formal specification of telephone systems in LOTOS: the constraint-oriented style approach", *Computer Networks and ISDN Systems*, vol. 21, pp. 53–67, 1991.

[34] M. Mac an Airchinnigh, D. Belsnes, and G. O'Regan, "Formal Methods & Service Specification", in *Towards a Pan-European Telecommunication Service Infrastructure - IS&N 94, Second International Conference on Intelligence in Broadband Services and Networks*, H. J. Kugler, A. Mullery, and N. Niebert, Eds. September 1994, Lecture Notes in Computer Science (851), pp. 563–572, Springer-Verlag.

[35] J. B. Stefani, "Open distributed processing: an architectural basis for information networks", *Computer Communications*, vol. 18, no. 11, pp. 849–862, November 1995.

[36] J. Bredereke, "Automata-Theoretic vs. Property-Oriented Approaches for the Detection of Feature Interactions in IN", in *Proceedings of the International Workshop on Advanced Intelligent Networks (AIN'96)*, T. Margaria, Ed., Passau, March 1996, pp. 56–70, Universität Passau, Fakultät für Mathematik und Informatik.

[37] T. Bolognesi and E. Brinksma, "Introduction to the ISO Specification Language LOTOS", *Computer Networks and ISDN Systems*, vol. 14, pp. 25–59, 1987.

[38] B. Stepien and L. Logrippo, "Feature Interaction Detection using Backward Reasoning with LOTOS", in *Protocol Specification, Testing and Verification XIV*, S. T. Vuong and S. T. Chanson, Eds., London, 1995, pp. 71–86, Chapman & Hall.

[39] K. J. Turner, “An architectural foundation for relating features”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 226–241, IOS Press.

[40] J. H. Choi, H. S. Kim, W. J. Lee, and Y. R. Kwon, “A Petri-Nets Based Approach for Detecting Feature Interactions in Telecommunications Services”, in *Proceedings of the 12th International Conference on Computer Communication (ICCC) 1995*, Seoul, August 1995, pp. 596–601.

[41] Y. Inoue, K. Takami, and T. Ohta, “Method for Supporting Detection and Elimination of Feature Interaction in a Telecommunication System”, in *International Workshop on Feature Interactions in Telecommunications Software Systems*, December 1992, pp. 61–81.

[42] Y. Inoue, K. Takami, and T. Ohta, “Automatic Detection of Service Interactions in Telecommunications Service Specifications”, in *Proceedings of the IEEE International Conference on Communications (ICC) 1994*, New Orleans, May 1994, pp. 1835–1840.

[43] Y. Harada, Y. Hirakawa, and T. Takenaka, “A Design Support Method for Telecommunication Service Interactions”, in *Proceedings GLOBECOM 1991*, Phoenix, AZ, December 1991, pp. 1661–1666.

[44] Y. Harada, Y. Hirakawa, T. Takenaka, and N. Terashima, “A Conflict Detection Support Method for Telecommunication Service Descriptions”, *IEICE Transactions on Communications*, vol. E75-B, no. 10, pp. 986–997, October 1992.

[45] T. Ohta and Y. Harada, “Classification, Detection and Resolution of Service Interactions in Telecommunication Services”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 60–72, IOS Press.

[46] Y. Kawarasaki and T. Ohta, “A New Proposal for Feature Interaction Detection and Elimination”, in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 127–139, IOS Press.

[47] M. Nakamura, Y. Kakuda, and T. Kikuno, “Petri-Net Based Detection Method for Non-Deterministic Feature Interactions and its Experimental Evaluation”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 138–152, IOS Press.

[48] L. R. Brothers, E. J. Cameron, Y. J. Lin, M. E. Nilson, and E. Silverstein, “Feature Interaction Detection”, in *Proceedings of the IEEE ICC’93*, Geneva, May 1993, pp. 1553–1557.

[49] E. J. Cameron and Y.-J. Lin, “A real-time transition model for analyzing behavioral compatibility of telecommunications services”, in *Proceedings of ACM SIGSOFT Conference on Software for Critical Systems*, New Orleans, LA, December 1991, pp. 101–110.

[50] J. Bredereke, “Formal Criteria for Feature Interactions in Telecommunications Systems”, in *Proceedings of the IFIP International Working Conference on Intelligent Networks*, Copenagen, August 1995, pp. 68–83.

[51] J. Bredereke and R. Gotzhein, “Specification, Detection and Resolution of IN Feature Interactions with Estelle”, in *Formal Description Techniques, VII*, D. Hogrefe and S. Leue, Eds., Berne, October 1994, pp. 366–368.

[52] C. Klein, C. Prehofer, and B. Rumpe, “Feature Specification and Refinement with State Transition Diagrams”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 284–297, IOS Press.

[53] A. Khoumsi, "Detection and Resolution of Interactions between Services of Telephone Networks", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 78–92, IOS Press.

[54] J. G. Thistle, R. P. Malhame, and H. H. Hoang, "Feature Interaction Modelling, Detection and Resolution: A Supervisory Control Approach", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 93–107, IOS Press.

[55] Y. L. Chen, S. Lafortune, and F. Lin, "Resolving Feature Interactions Using Modular Supervisory Control with Priorities", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 108–122, IOS Press.

[56] R. Dssouli, S. Some, J. W. Guillery, and N. Rico, "Detection of Feature Interactions with REST", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 271–283, IOS Press.

[57] Y. Iraqi and M. Erradi, "An experiment for the processing of Feature Interactions within an object-oriented environment", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 298–312, IOS Press.

[58] K. H. Braithwaite and J. M. Atlee, "Towards Automated Detection of Feature Interactions", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 36–59, IOS Press.

[59] K. P. Pomakis and J. M. Atlee, "Reachability Analysis of Feature Interactions: A Progress Report", in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA 96)*, 1996, pp. 216–223.

[60] P. K. Au and J. M. Atlee, "Evaluation of a State-Based Model of Feature Interactions", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 153–167, IOS Press.

[61] F. S. Dworack, "Approaches to Detecting and Resolving Feature Interactions", in *Proceedings GLOBECOM 1991*, Phoenix, AZ, December 1991, pp. 1371–1377.

[62] M. Frappier, A. Mili, and J. Desharnais, "Detecting Feature Interactions on Relational Specifications", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 123–137, IOS Press.

[63] A. Lee, "Formal Specification - A Key to Service Interactions Analysis", in *Proceedings of the Eight Conference on Software Engineering for Telecommunication Systems and Services (SETSS 1992)*, March 1992, pp. 62–66.

[64] A. Y. H. Lee, *Formal Specification and Analysis of Intelligent Network Services and their Interaction*, PhD thesis, University of Queensland (Australia), December 1992.

[65] C. Capellmann, P. Combes, J. Pettersson, B. Renard, and J. L. Ruiz, "Consistent Interaction Detection - A Comprehensive Approach Integrated with Service Creation", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 183–197, IOS Press.

[66] J. Bergstra and W. Bouma, "Models for Feature Descriptions and Interactions", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 31–45, IOS Press.

[67] J. Blom, R. Bol, and L. Kempe, "Automatic Detection of Feature Interactions in Temporal Logic", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 1–19, IOS Press.

[68] J. Blom, B. Jonsson, and L. Kempe, “Using Temporal Logic for Modular Specification of Telephone Services”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 197–216, IOS Press.

[69] J. Blom, “Formalisation of Requirements with emphasis on Feature Interaction detection”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 61–77, IOS Press.

[70] S. M. Rochefort and H. J. Hoover, “An Exercise in Using Constructive Proof Systems to Address Feature Interactions in Telephony”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 329–341, IOS Press.

[71] D. D. Dankel II, M. Schmalz, W. Walker, K. Nielsen, L. Muzzi, and D. Rhodes, “An Architecture for Defining Features and Exploring Interactions”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 258–271, IOS Press.

[72] T. Charnois, “A Natural Language Processing Approach for Avoidance of Feature Interactions”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 347–363, IOS Press.

[73] K. Kimbler and D. Sobirk, “Use Case Driven Analysis of Feature Interactions”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 167–177, IOS Press.

[74] K. Kimbler, E. Kuisch, and J. Muller, “Feature Interaction among Pan-European Services”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 73–85, IOS Press.

[75] M. Thomas, “Modelling and Analysing User Views of Telecommunications Services”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 168–182, IOS Press.

[76] C. Capellmann, R. Demant, F. Fatahi-Vanani, R. Galvez-Estrada, U. Nitsche, and P. Ochsenschläger, “Verification by Behaviour Abstraction - A Case Study of Service Interaction Detection in Intelligent Telephone Networks”, in *Proceedings, Computer Aided Verification 1996 (CAV 1996)*, Berlin, 1996, pp. 466–469.

[77] C. Capellmann, R. Demant, R. Galvez-Estrada, U. Nitsche, and P. Ochsenschläger, “Case Study: Service Interaction Detection by Formal Verification under Behaviour Abstraction”, in *Proceedings of the International Workshop on Advanced Intelligent Networks (AIN’96)*, T. Margaria, Ed., Passau, March 1996, pp. 71–90, Universität Passau, Fakultät für Mathematik und Informatik.

[78] I. Aggoun and P. Combes, “Observers in the SCE and the SEE to Detect and Resolve Service Interactions”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 198–212, IOS Press.

[79] “CCITT Specification and Description Language (SDL)”, ITU-T Recommendation Z.100, Geneva, 1993.

[80] B. Renard, P. Combes, and F. Olsen, “An SDL/MSC Environment For Service Interaction Analysis”, in *Proceedings of 4th International Conference on Intelligence in Networks / 4e Colloque International sur l’Intelligence dans les Reseaux (ICIN’96)*, Bordeaux, November 1996, pp. 200–205.

[81] W. Bouma, W. Levelt, A. Melisse, K. Middelburg, and L. Verhaard, “Formalisation of Properties for Feature Interaction Detection: Experiece in a Real-Life Situation”, in *Towards a Pan-European Telecommunication Service Infrastructure - IS&N 94, Second International Conference on Intelligence in Broadband Services and Networks*, H. J. Kugler, A. Mullery, and N. Niebert, Eds. September 1994, Lecture Notes in Computer Science (851), pp. 393–405, Springer-Verlag.

[82] P. Combes and S. Pickin, "Formalisation of a User View of Network and Services for Feature Interaction Detection", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 120–135, IOS Press.

[83] C. Fischer, S. Kleuker, and E. R. Olderog, "Beweisbar korrekte Telekommunikationssysteme", *Informationstechnik und Technische Informatik (it+ti)*, vol. 39, no. 3, pp. 22–28, Mai/Juni 1997.

[84] S. Kleuker, "The extension of existing telecommunication software with new services using formal methods", in *Proceedings of the International Workshop on Advanced Intelligent Networks (AIN'96)*, T. Margaria, Ed., Passau, March 1996, pp. 91–106, Universität Passau, Fakultät für Mathematik und Informatik.

[85] H. Korver, *Detecting Feature Interactions with Caesar/Aldebaran*, Tech. Report CS-R9370, CWI, P.O. Box 94079, 1090 GB Amsterdam, Amsterdam, 1993.

[86] M. Faci, *Detecting Feature Interactions in Telecommunications Systems Designs*, PhD thesis, University of Ottawa (Canada), 1995.

[87] M. Faci and L. Logrippo, "Specifying Features and Analysing Their Interactions in a LOTOS Environment", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 136–151, IOS Press.

[88] R. Boumezbeur and L. Logrippo, "Specifying Telephone Systems in LOTOS", *IEEE Communications Magazine*, vol. 31, no. 8, pp. 38–45, August 1993.

[89] B. Stepien and L. Logrippo, "Representing and Verifying Intentions in Telephony Features Using Abstract Data Types", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 141–155, IOS Press.

[90] F. J. Lin and Y. J. Lin, "A Building Block Approach to Detecting and Resolving Feature Interactions", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 86–191, IOS Press.

[91] G. J. Holzmann, *Design and validation of computer protocols*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[92] A. Gammelgaard and J. E. Kristensen, "Interaction detection, a logical approach", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 178–196, IOS Press.

[93] M. Boström and M. Engstedt, "Feature Interaction Detection and Resolution in the Delphi Framework", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 157–172, IOS Press.

[94] O. C. Dahl and E. Najm, "Specification & Detection of IN Service Interference using LOTOS", in *Formal Description Techniques, VI (FORTE '93)*, R. L. Tenney, P. D. Amer, and M. Ü. Uyar, Eds. October 1993, IFIP Transactions (C-22), pp. 53–69, North-Holland.

[95] J. P. Gibson, "Feature Requirements Models: Understanding Interactions", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 46–60, IOS Press.

[96] B. Kelly, M. Crowther, J. King, R. Masson, and J. DeLapeyre, "Service Validation and Testing", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 173–184, IOS Press.

[97] B. Kelly, M. Crowther, and J. King, "Feature Interaction Detection Using SDL Models", in *Proceedings of the IEEE GLOBECOM '94*, November 1994, pp. 1857–1861.

[98] S. Tsang and E. H. Magill, "Detecting Feature Interactions in the Intelligent Network", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 236–248, IOS Press.

[99] J. C. Godskesen, "A Formal Framework for Feature Interaction with Emphasis on Testing", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 21–30, IOS Press.

[100] A. O. Oshisanwo, M. D. Chapman, M. Key, A. P. Mullery, and J. Saint-Blancat, "The RACE Open Services Architecture Project", *IBM Systems Journal*, vol. 31, no. 4, pp. 691–710, 1992.

[101] E. Kuisch, R. Janmaat, H. Mulder, and I. Keesmaat, "A Practical Approach to Service Interactions", *IEEE Communications Magazine*, vol. 31, no. 8, pp. 24–31, August 1993.

[102] Y. Wakahara, M. Fujioka, H. Kikuta, H. Yagi, and S. I. Sakai, "A Method for Detecting Service Interactions", *IEEE Communications Magazine*, vol. 31, no. 8, pp. 32–37, August 1993.

[103] "Definition of Supplementary Services", ITU-T Recommendation I.250, Geneva, 1989.

[104] S. Homayoon and H. Singh, "Methods of Addressing the Interactions of Intelligent Network Services With Embedded Switch Services", *IEEE Communications Magazine*, vol. 26, no. 12, pp. 42–70, December 1988.

[105] L. Schessel, "Administrable Feature Interaction Concept", in *Proceedings of the XIV International Switching Symposium (ISS)*, Yokohama, October 1992, pp. B6.3–B6.3.

[106] J. Bredereke and R. Gotzhein, *A Case Study on Specification, Detection and Resolution of IN Feature Interactions with Estelle*, Tech. Report 245/94, FB Informatik, Universität Kaiserslautern, Kaiserslautern, May 1994.

[107] K. E. Cheng, "Towards a Formal Model for Incremental Service Specification and Interaction Management Support", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 152–166, IOS Press.

[108] "Distributed Functional Plane for Intelligent Network CS-1", ITU-T Recommendation Q.1214, Geneva, 1993.

[109] M. Cain, "Managing Run-Time Interactions Between Call-Processing Features", *IEEE Communications Magazine*, vol. 30, no. 2, pp. 44–50, February 1992.

[110] N. Fritsche, "Runtime Resolution of Feature Interactions in Architectures with Separated Call and Feature Control", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 43–63, IOS Press.

[111] N. Fritsche, *Vermittlungsarchitektur mit getrennter Ruf- und Leistungsmerkmalsteuerung*, PhD thesis, Technische Universität München, 1996.

[112] N. Ogino, "Service Interaction Resolution by Service Node Installed out of the Network", *IEICE Transactions on Communications*, vol. E80-B, no. 10, pp. 1537–1546, October 1997.

[113] "Interface Recommendation for Intelligent Network CS-1", ITU-T Recommendation Q.1218, Geneva, 1995.

[114] S. Tsang and E. H. Magill, "Behaviour Based Run-Time Feature Interaction Detection and Resolution Approaches For Intelligent Networks", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 254–270, IOS Press.

[115] B. Makarevitch, “Resolving Service Interactions by Service Components”, in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 213–221, IOS Press.

[116] B. Makarevitch, “An Object-based Model of the Service Control Function”, in *Proceedings of the 12th International Conference on Computer Communication (ICCC) 1995*, Seoul, August 1995, pp. 183–188.

[117] D. Marples, S. Tsang, E. H. Magill, and D. G. Smith, “A Platform for Modelling Feature Interaction Detection and Resolution Techniques”, in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 185–199, IOS Press.

[118] Y. Kakuda, A. Inoue, H. Asada, T. Kikuno, and T. Ohta, “A Dynamic Resolution Method for Feature Interactions and Its Evaluation”, in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 97–114, IOS Press.

[119] Y. Kakuda, H. Asada, and T. Kikuno, “Application of Protocol Synthesis Technique to Resolution of the Service Interaction Problem”, in *Formal Description Techniques, VII*, D. Hogrefe and S. Leue, Eds., Berne, October 1994, pp. 379–381.

[120] C. Prehofer, “An Object-Oriented Approach to Feature Interaction”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 313–325, IOS Press.

[121] G. Utas, “Feature Interaction: A Software Perspective”, in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 23–28, IOS Press.

[122] G. Utas, “GSF: An Architecture for the Evolving Network”, in *Proceedings of the XV International Switching Symposium (ISS) 1995, Volume 2*, Berlin, Offenbach, April 1995, pp. 293–297, VDE-Verlag.

[123] H. Velthuijsen, “Distributed Artificial Intelligence for Runtime Feature-Interaction Resolution”, *IEEE Computer*, vol. 26, no. 8, pp. 48–55, August 1993.

[124] N. D. Griffeth and H. Velthuijsen, “The Negotiating Agents Approach to Runtime Feature Interaction Resolution”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 217–235, IOS Press.

[125] N. D. Griffeth and H. Velthuijsen, “Reasoning about Goals to Resolve Conflicts”, in *Proceedings of the International Conference on Intelligent and Cooperative Information Systems*, Rotterdam, May 1993, pp. 197–204.

[126] P. Dini and G. v. Bochmann, “Automatic Reconfiguration for Runtime Feature-Interaction Resolution in an Object-Oriented Environment”, in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 115–126, IOS Press.

[127] R. van der Linden, “Using an Architecture to Help Beat Feature Interaction”, in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., Amsterdam, May 1994, pp. 24–35, IOS Press.

[128] D. Cattrall, G. Howard, D. Jordan, and S. Buj, “An Interaction-Avoiding Call Processing Model”, in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 85–96, IOS Press.

[129] I. Zibman, C. Woolf, P. O’Reilly, L. Strickland, D. Willis, and J. Visser, “An Architectural Approach to Minimizing Feature Interactions in Telecommunications”, *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 582–596, August 1996.

[130] I. Zibman, C. Woolf, P. O'Reilly, L. Strickland, D. Willis, and J. Visser, "Minimizing Feature Interactions: An Architecture and Processing Model Approach", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 65–83, IOS Press.

[131] M. Weiss, T. Gray, and A. Diaz, "Experiences with a Service Environment for Distributed Multimedia Applications", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 242–253, IOS Press.

[132] D. O. Keck, "Requirements and a Proposal for the Prevention of a Class of Service Interactions in Intelligent Networks", in *Services and Visualization, Towards User-Friendly Design (ACoS'98, Visual'98, AIN'97 Selected Papers)*, T. Margaria, B. Steffen, R. Rückert, and J. Posegga, Eds., Berlin, April 1998, Lecture Notes in Computer Science (1385), pp. 90–105, Springer-Verlag.

[133] R. Kung, "Open Networking: is it technically possible?", in *Proceedings of the XV International Switching Symposium (ISS) 1995, Volume 2*, Berlin, Offenbach, April 1995, pp. 212–216, VDE-Verlag.

[134] K. Kimbler, "Towards A More Efficient Feature Interaction Analysis - A Statistical Approach", in *Feature Interactions in Telecommunications Systems, III*, K. E. Cheng and T. Ohta, Eds., Amsterdam, October 1995, pp. 201–211, IOS Press.

[135] D. O. Keck, "Identification of Call Scenarios with Potential Feature Interactions", in *Proceedings of the International Workshop on Advanced Intelligent Networks (AIN'96)*, T. Margaria, Ed., Passau, March 1996, pp. 42–55, Universität Passau, Fakultät für Mathematik und Informatik.

[136] R. B. Blumenthal, M. P. O'Reilly, and P. Russo, "Addressing Feature Interaction During Service Creation", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 364–370, IOS Press.

[137] K. Kimbler, "Addressing the Interaction Problem at the Enterprise Level", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 13–22, IOS Press.

[138] P. Georgatsos, T. Nauta, and H. Velthuijsen, "Role of Service Management in Service Interaction Handling in an IN environment", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 213–225, IOS Press.

[139] J. Cameron, K. Cheng, F. J. Lin, H. Liu, and B. Pinheiro, "A Formal AIN Service Creation, Feature Interactions Analysis and Management Environment: An Industrial Application", in *Feature Interactions in Telecommunication Networks IV*, P. Dini, R. Boutaba, and L. Logrippo, Eds., Amsterdam, June 1997, pp. 342–346, IOS Press.

[140] K. Kant and L. Ong, "Signaling in Emerging Telecommunications and Data Networks", *Proceedings of the IEEE*, vol. 85, no. 10, pp. 1612–1621, October 1997.

[141] F. Dupuy, G. Nilsson, and Y. Inoue, "The TINA Consortium: Towards Networking Telecommunications Information Services", *IEEE Communications Magazine*, vol. 33, no. 11, pp. 78–83, November 1995.

Biographies of the Authors

Dirk O. Keck was born in Tuebingen, Germany, in 1968. He has received his Dipl.-Ing. degree in electrical engineering from the University of Stuttgart, Germany, in 1994. Until 1995, he has been a scientific staff member at the Institute for Microelectronics Stuttgart (Professor B. Hoefflinger). Since then, he is a member of the scientific staff at the Insitute of Communication Networks and Computer Engineering at the University of Stuttgart. His research interests include private and public communication networks (ISDN and IN) and the application of formal methods to telecommunications system design. Mr. Keck is a member of ITG (German Communications Society).

Paul J. Kuehn (F'89) was born in Gruessau, Germany, in 1940. He received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from the University of Stuttgart, Germany, in 1967 and 1972, respectively.

From 1973 to 1977, he was head of a research group for traffic research in computer and communications systems at the University of Stuttgart. In 1977, he joined Bell Laboratories, Holmdel, NJ, where he worked in the field of computer communications. In 1978, he was appointed Professor of Communications Switching and Transmission at the University of Siegen, Germany. Since 1982, he has held the Chair of Communication Networks and Computer Engineering at the University of Stuttgart. His areas of interest include communications switching, computer and communications system performance evaluation, and computer networks.

Dr. Kuehn is a member of the ACM, ITG (German Communications Society), and GI (Geman Informatics Society). In 1977, 1979, and 1983, he became a member of the Communications Switching Committee of the NTG, the International Council of the International Teletraffic Congress (ITC), and W.G. 7.3 of the IFIP, respectively. He was appointed Vice President of the ITC in 1985 and Governor of the International Conference on Computer Communication (ICCC) in 1987. He was also Program Chairman of the ICCC in 1986 and of various national conferences on computer communication and performance modeling. In 1990, he was appointed Professor Associé at the Télécom Paris/ENST, and in 1991 was elected Chairman of the International Advisory Council (IAC) of the ITC. In 1993, he was appointed Member of the Academy of Sciences of Heidelberg, Germany, and in 1995, Member of the Academy Leopoldina, Halle, Germany. In 1996, he was awarded the Dr. h. c. for Technology by the Lund Institute of Technology, Sweden, and in 1998 the Dr.-Ing. E.h. by the University of Technology of Dresden, Germany.