Formal Methods for Communication Services

F. Dietrich and J.-P. Hubaux

Institute for computer Communications and Applications (ICA)
Swiss Federal Institute of Technology, CH-1015 Lausanne

September 7, 1999

Abstract

We survey formal methods as they are applied to the development of communication services. We report on industrial and academic projects, consider different communication architectures and work related to the feature interaction problem. Based on our survey, the results reported in the literature and most importantly, on extensive discussions with industry, we investigate important industrial concerns and criticisms about the use of formal methods for the development of communication services.

We report on a collaborative project between the Swiss Federal Institute of Technology in Lausanne, Swisscom, Alcatel and Thomson in which these industrial concerns have been taken into account from the very beginning. The results of this project are currently being integrated into an industrial software development platform.

1 Introduction

With installations that will soon reach one billion wireline phones, 300 million mobile phones and 200 million IP hosts, assuring communication service reliability is one of the most challenging tasks in software engineering. Unreliable and misimplemented communication services can easily trigger huge losses and damage customer faith. To rectify the users' bad impressions, a telecom operator not only has to correct the errors, but must compensate the unsatisfied customers. In today's competitive markets, customer faith is a difficult commodity to restore. Formal methods (FMs) have been advocated as one possible approach to increase software reliability.

However, even with the most liberal meaning of formal methods (FM) it is safe to say that formal methods are rarely used in the industry [39]. While academics see formal methods as inevitable in the future of the software profession, most practitioners see formal methods as irrelevant to what they do [36]. In this paper, we are specifically interested in formal methods as they are applied to the development of communication services.

The main contributions of this paper are as follows:

1. We survey formal methods as they are applied to the development of communication services. While there are some excellent surveys about formal methods in general [70, 18], there is currently none which tackles the specific area of communication services. In [21], it is pointed out that, in practice, preference is given to those FMs that have industrial acceptance. We therefore report on industrial and academic projects separately thus explicitly identifying the formal methods that have (or have not) gained industry acceptance. We consider different

communication architectures and discuss work related to the feature interaction problem.

2. Based on our survey, the results reported in the literature and most importantly, on extensive discussions with industry, we analyze important industrial concerns and criticisms about the use of formal methods for the development of communication services; thereby reporting on some of the major barriers for the application of FMs in the communications industry.

3. We present a collaborative project between the Swiss Federal Institute of Technology in Lausanne and Alcatel in Paris, in which these industrial concerns have been taken into consideration from the very beginning. The results of this project are currently being integrated into an industrial development platform, providing strong evidence of the applicability of formal methods in the industry if their specific needs are not ignored.

The remainder of this chapter is structured as follows: In Section 2 we explain what we mean by communication services and identify their peculiarities. In Section 3 we survey the formal approaches used for communication services. In Section 4 we discuss industrial concerns about the application of formal methods and show how they can be addressed in formal methods research. In Section 5 we describe the work carried out in our collaboration with Alcatel and show how our proposal pays attention to the industrial concerns. Finally, our conclusions are presented.

2 The Peculiarities of Communication Services

In this section we explain what we mean by *communication services* and identify their peculiarities.

We follow the definition given in [67] where a *service* is defined as a meaningful set of capabilities provided by an existing or intended set of systems to all who utilize it: subscribers, end users, network providers, and service providers - each seeing a different perspective of the service. The term *communication service* as used in this paper refers to telecommunication services, Internet services and hybrid services. A *telecommunication service* is defined as a service that is provided by the Public Switched Telephone Network or mobile technology. Typical examples of telecommunication services are call forwarding or originating call screening but also video conference services. An *Internet service* is defined as a service that is provided over Internet infrastructure. A *hybrid service* [76] is defined as a service that spans many network technologies, especially the Public Switched Telephone Network (PSTN) and the Internet. An example of a simple hybrid service is Click-to-Dial, which enables a user to request, from a Web browser, a connection to be set up between telephones connected to the PSTN.

Another term frequently used in the communications domain is *protocol*. A *protocol* is the special set of rules for communicating that the end points in a communication connection use when they send signals back and forth. Protocols exist at several levels in a communication connection. Protocols are often described in an industry or international standard. For example, on the Internet, there are the TCP/IP protocols, consisting of TCP (Transmission Control Protocol), which uses a set of rules to exchange messages with other Internet points at the information packet level, and IP (Internet Protocol), which uses a set of rules to send and receive messages at the Internet address level.

In this paper we expressly refrain from looking at the application of formal methods for *communication protocols* but concentrate only on *communication services.* We acknowledge that it is not always possible to draw a clear line between communications protocols and communications services and in some cases one might be tempted to say

Concurrency, distribution, reactivity Communication services belong to the class of concurrent, distributed and reactive systems.

Real-time Communications services have to meet real-time deadlines; this means that they have to respond to a stimulus within a fixed (not necessarily in the range of milliseconds) time limit.

Code size, complexity Industrial communication software is huge in code size and very often is composed of thousands or even hundreds of thousands of lines of implementation code.

Large-scale environment Communications services operate in a large-scale environment, where large-scale refers to three major points: (i) wide geographical distribution, (ii) a high number of participating objects and processes, (iii) a high number of users interacting with the service.

Reliability, availability and fault tolerance Only marginal down-time is acceptable for communication systems. To increase reliability and availability, industrial communication systems cannot be developed without considering fault tolerance, which is quite an essential point for communication systems. One of the key characteristics of communication systems is 24h × 7d operation and they are seldomly restarted.

Streams, connectivity Communications services incorporate streams and provide some sort of connectivity, even if in some cases, the user data are conveyed by a connectionless network.

Coexistence, interoperability A communication service has to coexist and interoperate with other services.

Heterogeneous environment The authors of [73] point out that the real peculiarity of (tele-) communication services which singles them out from the set of other complex, real-time software like nuclear plants and aircraft, is the environment, both for system production and system operation. The tremendous heterogeneity of todays communication systems is a result of their historical, evolutionary development and perpetual introduction of new services and technologies. Services newly introduced or existing services that are being extended, changed or updated, have to cope with this heterogeneous environment.

Long-lived and evolving Communications systems are long-term investments; existing systems are constantly being extended. Many communication services are offered for a long time period and are thus subject to many upgrades and evolutions.

Table 1: The Peculiarities of Communication Services

that a service is just a protocol. Intuitively, we exclude all works that explicitly refer to protocols.

Some characteristics of public networks are identified in [40] and some peculiarities of telecommunication systems are briefly discussed in [73]. These characteristics partly match the characteristics of communication services. The most important characteristics of communication services are summarized in Table 1 without attributing any specific meaning to the order in which they are listed. We will come back to some of the peculiarities in Section 4.

3 Survey of Formal Methods for Communication Services

In this section we survey the application of formal methods for communication services. We first look at the work done in industry and academia, we investigate the formal methods used for specific communication architectures (Intelligent Network, TINA, Internet), and look at work related to the feature interaction problem. Work that covers more than one aspect (e.g., a paper describing an industrial project for feature interaction detection in the TINA architecture) is referenced several times but discussed only wherever it seems to be most appropriate.

It is outside the scope of this survey to discuss the advantages and disadvantages of a given formal method with respect to other formal methods. The reader interested in such a comparative study is referred to [1] where the authors provide a comparative case study for Esterel, LOTOS, Modecharts, SDL, VFMS and Z. A set of eleven fundamental and five important criteria is described and the above-mentioned FMs are evaluated according to these criteria.

FSM (Finite State Machines). A simple finite state machine is composed of states connected by transitions. The relative simplicity of finite state machines makes them especially suitable for formal analysis.
SDL (Specification and Description Language) is based on an extended finite state machine model, supplemented by features for specifying abstract data types. SDL has probably received more attention from industry than any other formal method and is well-supported by tools, some of them are commercially available.
LOTOS (Language Of Temporal Ordering Specification) is a formal specification technique for specifying concurrent and distributed systems. It consists of a language for specifying processes and an algebraic specification language called ACT ONE.
Estelle (Extended Finite State Machine Language). An Estelle specification defines a system of hierarchically-structured state machines. The state machines communicate by exchanging messages through bi-directional channels between their communication ports.
Esterel is a synchronous programming language that is based on the perfectly synchronous concurrency model in which concurrent processes are able to perform computations and exchange information in zero time.
Z is a (non-executable) formal specification notation based on set theory and first order predicate logic.
Temporal logic is a formal specification language for the description and analysis of time-dependent and behavioral aspects. Temporal logic is an extension of conventional propositional logic that incorporates special operators that cater for time. There are many temporal logics; some of them being linear time temporal logic (LTL) and ACTL* (an action based temporal logic).
Promela is a protocol validation language. The SPIN tool has been devised to validate Promela specifications.
Others. There are many more formal techniques, such as Larch, TLA (the Temporal Logic of Actions), etc., but they have received rather limited attention from the (tele-) communications community.

Table 2: Formal Techniques

Table 2 lists some of the most frequently used formal techniques. Almost every formal method has been applied to communication services: Finite State Machines (FSM) [34], Petri nets [68], Promela [29], Esterel [50] and many others. However, the (tele-)communications community has been paying special attention to the three standardized FDTs: LOTOS [45], Estelle [46] and SDL [15]. These FDTs, originally developed to unambiguously specify protocol standards, became standardized about 15 years ago and they have been successfully applied for protocol specification and validation since then: several design errors have been found in a number of protocols. The number of projects using these FDTs for service design has increased steadily as more and more tools have become available. The application of these protocol specification languages to service engineering triggered many extensions such as for real-time and object-orientation, and led to the fact that there is almost more work *on* formal methods rather than *with* them.

3.1 Formal Methods in Industry

There are few reliable statistics about faults in communication systems. Starting in 1992, however, telephone companies in the US have been required to notify the US Federal Communications Commission of outages affecting more than 30,000 customers. These reports are statistically analyzed in [58]. This analysis revealed several very interesting facts:

- Overloads accounted for 6 percent of the outages, but for 44 percent of the down-time.
- Software errors caused 14 percent of the outages, but less system down-time (two percent!) than any other source of failure except vandalism.

Even though the telecommunication network with its multitude of telecommunications software is the largest network worldwide, it is also one of the most reliable networks. It is not surprising that the telecommunications industry is often on the forefront of formal methods research.

In [21], the importance of FMs is clearly acknowledged. However, preference is given to those FMs that have industrial acceptance, tool support of production quality and that are standardized. At present, only SDL fulfils these three requirements.

Table 3 summarizes projects in which formal methods have been applied to the design of communications services in an industrial setting. It turns out that SDL, Z and Promela are used in the communications industry whereas industry has not paid much attention to other formal methods, such as LOTOS for the specification and validation of communication services. Temporal logic (TL) also seems to have attracted some interest from the industry and has been used in connection with other FDTs such as SDL [42]. TL is also integrated in commercial products [77] [79].

Company	FM	References
AT&T	Esterel	[50]
AT&T	SDL	[5] [43] [41]
AT&T	SDL, TL	[42]
AT&T	Z	[84]
AT&T	Promela, Z	[85]
Bellcore	Promela, LTL	[60]
BT	Z	[2]
BT	SDL	[54] [28]
CSELT	Promela	[9]
Deutsche Telekom et al.[1]	Petri-nets, SDL	[14]
Dutch PTT, Telia	SDL, ACTL*	[12] [66]
France Télécom	Z	[53]
Nortel	SDL	[82]
Score[2]	SDL, Z, ACTL*	[20]
SNI	FSM, TL	[77]

Table 3: Formal Methods in the Communications Industry

[1] Deutsche Telekom, France Télécom CNET, Tele Danmark A/S, Telefónica I+D

[2] Score consortium: IBM France, Broadcom, CAP SESA Telecom, CAP Gemini Innovation, France Télécom CNET, ERICSSON Telecom AB, INESC, PTT-NL, Tele Danmark, Telia AB, Telelogic, Telia Promotor Uppsala AB, Verilog, CSELT, ABB Corporate Research, NR.

At AT&T, several FM projects aimed at designing and implementing software for the AT&T 5ESS (Electronic Switching System).[3] Several formal methods have been applied over a longer time-period [5, 43, 42, 50].

Jagadeesan, Puchol and Olnhausen [50] described a technique and supporting tools that automatically verify whether Esterel programs satisfy safety properties. The verification of those properties is based on an automated translation of the temporal properties to Esterel, the compilation of the given program in parallel with the Esterel "model" of the properties; and the analysis of the resulting finite state machine for satisfaction or violations of the properties. The NewCoRe project described in [42] ran over a two year period. A specification of 7,500 lines of (non-commented) SDL code was written and about 150 correctness properties were formally specified and verified for the SDL model. As a result, a total of 112 serious design errors were detected in the design requirements. Holzmann claims that the use of automated formal verification techniques for industrial software design had never been attempted on such a scale before. To our best knowledge, this is still true. Holzmann points out that the first generation of FMs that we developed does not live up to the promise of either an engineering discipline or a scientific method. He lists three characteristics that a mature engineering discipline should minimally have:

- It discriminates between requirements and implementations, i.e., it should be possible to make explicit formal statements about the correctness requirements of a design independent of the design itself.
- It uses engineering models (prototypes) to verify design decisions.
- It can predict the essential characteristics of a product before it is built.

Since neither SDL nor Esterel provide means to make statements about the correctness of a design independent of the design itself, in both [42] and [50], correctness requirements for the 5ESS switching software were expressed using temporal logic.

The weakness of SDL for expressing correctness requirements has also been pointed out in [66] and [12]. The proposal of these authors is to use a temporal logic based on ACTL* to express properties for telecommunication services.

Temporal logic has also been used in the project described in [77], which deserves special attention since it describes work that made the step into a commercial product. For the specification of services in the Intelligent Network, Siemens Nixdorf Informationssysteme, Munich, Germany and the University of Passau, Germany, developed an environment for the creation of Intelligent Network services. Services described in this framework can be formally verified by model checking. Service properties are expressed using temporal logic. By making the use of formally specified constraints an option it is up to the service designer to decide to which degree he is willing to invest into formality; the specification of more constraints leads to a more faithful verification of the service design. This seems to be a very promising avenue to follow.

The need for an evolutionary rather than a revolutionary integration of formal methods into system development is expressed in [2]. While many projects produce complete formal specifications, the approach advocated by British Telecom and Leeds Metropolitan University leads to a gradual introduction of formal specification. Z is used as an add-on to the normal development process. While, in the beginning, the system specifications are developed as usual, those specifications are later transformed into a formal specification in Z.

CSELT developed the Application Construction Environment (ACE) [9] for the specification, development and generation of TINA services. Besides containing a family of graphical editors, it also comprises a compiler that translates (intermediate

[3] After the split of AT&T in 1996, the 5ESS is now associated with Lucent.

code of) the specification into Promela. The SPIN tool [41] is then used as model checker.

The application of the formal description language Z to the specification of TMN (Telecommunication Management Network) interfaces has been studied in [53] and it is shown how Z can be used for the specification of the OSI management information model.

In [85], Zave investigates the use of Promela and Z for the specification of telecommunication services. By using a joint semantics for Promela and Z, specifications written in Promela and Z can be composed and be reasoned about in a number of ways, e.g., by using tools that are available for either Promela or Z. The starting point for the investigation is the Distributed Feature Composition (DFC) virtual architecture which is detailed in [49].

The Score consortium [20] has investigated the use of MSCs, SDL and ACTL* and Z. To produce formal specifications, Score recommends OMT, SDL and Z with corresponding supporting tools [22]. However, it is also pointed out that yet a lot of work has to be done when one wants to introduce the methods in an industrial environment [23].

The work described in [82] starts with an identification of industrial problems by interviewing Nortel engineers. Several FMs are evaluated for their suitability and SDL is finally chosen to formally specify a multi-media messaging subsystem.

3.2 Formal Methods in Academia

The use of formal methods for the design of communications services is heavily advocated by academics. Almost every formal method in existence has been applied to communication services. Table 4 lists some of the communication service related formal methods projects carried out in an academic environment.

FM	References
FSM	[13] [34]
Petri-nets	[68]
LOTOS	[31] [57] [78] [32] [6]
Promela	[29]
SDL	[74]
TL	[8] [7] [29] [61] [6]

Table 4: Formal Methods in Academia

The application of LOTOS for the specification of telephone systems is, for example, described in [31] and [32]. These papers also address the question of how to integrate behavioral properties into LOTOS specifications.

In [6], the authors investigate the use of LOTOS and a real-time temporal logic called QTL for the specification and verification of multimedia services.

Because much of the academic work is related to the feature interaction problem and/or specific communication architectures, it will be discussed in the following sections.

3.3 Formal Methods for Feature Interactions

A significant amount of work on the application of formal methods to the design of communication services deals with the problem of service interactions. A service interaction occurs when the addition of a new feature (e.g., by introducing a new service) to a system disrupts the existing services. Sometimes service interactions are desired, and one service is explicitly designed to interact with another. In some cases, one wants to verify that the two services interact exactly as expected. In most cases, one simply wants to ensure that the behavior of a service does not alter when composed with other (supposedly non-interacting) services/features.

A yearly international workshop [33] [10] [17] [27] dedicated to the feature interaction (FI) problem and many other related publications provide strong evidence of the importance of the problem and the relevance to many researchers. Table 5 summarizes works related to the FI problem and categorizes them according to the FM used.

FM	References
FSM	[13] [69]
Petri-nets	[52] [68] [14]
Promela	[60] [29]
SDL	[54] [14]
LOTOS	[11] [30] [16]
TL	[8] [7]

Table 5: Formal Methods for Feature Interactions

The approach described in [13] proposes to detect feature interactions by describing the services as layered finite state machines. While the originating and the terminating end of a call are described by simple finite state machines, features are added on top of these as layers.

Similarly, the authors of [69] advocate the use of finite state machines to describe the system's behavior. The network is considered to be a black box and service specifications are descriptions of desired terminal behaviors. Feature interactions are modelled as unwanted properties of these descriptions. In a follow-up work [52], the authors analyze some weaknesses of their original proposal such as timing verification and propose a new method based on Petri-nets.

Nakamura et al. [68] describe an algorithm for the detection of non-deterministic features that is based on a Petri-net model. This algorithm is more efficient than typical detection algorithms with state enumeration. Experimental results have shown that, even though the proposed algorithm may theoretically detect non-determinism that actually does not occur, the non-determinism detected in five sample service specifications was detected correctly.

In [60], features are modelled as building blocks that can be combined into complete descriptions. Features have a procedural representation (in form of Promela code) and assertions are described by using linear-time temporal logic. Similar to [29], the SPIN tool is used to check whether or not the assertions expressed in temporal logic (described as never-claims in Promela) are violated.

Several papers [11] [30] [16] describe how LOTOS can be used to describe features and to detect interactions.

In [8] and [7], a temporal logic inspired by Lamport's TLA (Temporal Logic of Actions) [59] is used to describe the different features. Two features interact if they result in an inconsistent description.

The approach described in [14] incorporates several formal notations like SDL, Petri-nets and MSCs. Different formalisms and interaction detection mechanisms are proposed for the different phases of the service life-cycle. Consistency of the global approach is achieved by defining several models that form the basis for all methods.

3.4 Formal Methods for Specific Architectures

Much of the work on formal methods for communication services concentrates on specific communication architectures. In the following we consider the Intelligent Network (IN) [47], the Telecommunications Information Networking Architecture (TINA) [80] and the Internet. Table 6 summarizes the approaches discussed in this paper.

Arch.	FM	Reference
IN	LOTOS	[11]
IN	Promela	[29]
IN	SDL	[54]
IN	FSM, TL	[77]
TINA	LOTOS	[57]
TINA	ODP-DLcomp (LOTOS)	[78]
TINA	Promela	[9]
TINA	LTL	[61]
TINA	SDL	[74]

Table 6: Formal Methods for Specific Architectures

Let us first look at some formal approaches for the *Intelligent Network*: In [11], Bouma and Zuidweg present an approach to analyzing feature interactions in the IN CS-1 (Capability Set 1) Global Functional Plane. The desired behaviour of a service (feature) is represented by a property satisfied by its LOTOS specification, and the feature interaction problem appears as a satisfiability problem for the conjunction of properties. Model-checking and simulation techniques are used to identify violations of these composite properties.

Etique [29] bases his approach on the FUSION method [19], thereby taking an object-oriented development method as basis for the construction of services in the intelligent network. As the FUSION method is not formal enough to allow an automated validation of specifications, the FUSION notations are extended and formalized to obtain a language called FUS++. To validate a FUS++ specification, it is translated to Promela and, subsequently, the SPIN tool is used to validate the corresponding Promela specification.

The work described in [54] uses SDL to validate and test IN services. Two models are used: a network model and a service plane model. The network model supports several nodes with an in-built interaction detection ability in its architecture. The service plane model is an abstract view of the telephone service with mechanisms for the creation and addition of services.

In 1993, over 40 network operators, telecommunications equipment and computer equipment manufacturers formed the *Telecommunications Information Networking Architecture* Consortium (TINA-C) to define and validate a common and open software architecture for the provision of telecommunication and information services. Due to the evolving character of TINA and doubts about its future [44], the use of formal methods for the design and implementation of TINA services is still in an early stage. Since TINA is built upon ODP concepts and uses CORBA (Common Object Request Broker Architecture) as underlying platform for services, results obtained from applying FMs for ODP (Open Distributed Processing) and in CORBA-frameworks can be reused for TINA services.

In the approach described in [61], behavioral properties of TINA services are expressed using event-based behavioral abstraction and Linear-time Temporal Logic (LTL). Whether the service implementation has not violated and is not violating the formally specified properties is checked at run-time. The approach has been applied to an industrial desktop video conferencing service and is supported by a tool.

ODP-DLcomp [56] is a formal language similar to TINA ODL (Object Definition Language) [55], but offers the capability to formally specify behavior. Its formal semantics is based on LOTOS. An example of it is given in [78] where it is used to specify a plain old telephony system.

Similarly, the authors of [74] start with the interface descriptions given by CORBA IDL (Interface Definition Language) or TINA ODL that lack a behavioral description of the components and they investigate the use of SDL for behavior descriptions.

In [57], the authors use LOTOS for the design of TINA-based applications. Specifically, they specify and verify TINA service components in the ODP computational viewpoint and discuss how compliance checks with the enterprise model of ODP can be carried out.

According to International Data Corporation, revenues in the worldwide *Internet* services market grew an astounding 71% in 1998 to reach $7.8 billion. Revenues in this emerging market will earn a compound annual growth rate of nearly 60% and pass $78 billion by 2003. Malfunctions of Internet service can have an adverse affect on the core business of companies that provide services over the Internet and on their customers. Surprisingly, the emergence and rapid growth of Internet services has gone almost completely unnoticed by the formal methods community. This is even more mysterious as Internet services account for a significant part of the revenues for many emerging and well-established companies in this multi-billion market. However, as service development platforms for the Internet, e.g. [81], become mature, these platforms are likely to be extended with service validating functionality in the near future. An example can be seen in the work described in [62].

4 The Concerns of the Communications Industry

In this section we look at some of the problems that, according to the industry, prevent or hinder the application of formal methods in an industrial environment. We discuss six important industrial concerns and show how these industrial concerns can be addressed by formal methods research. These industrial concerns may or may not be justified, one may or may not agree with them and it is outside the scope of this paper to investigate *if* they are justified. However, these *are* most important industrial concerns. We argue that, by actively addressing them, formal methods are much easier to transfer into mainstream service development, as we will show in Section 5.

Very often, formal methods are applied based on simple personal preferences and historical background of the researchers. The ever-increasing number of extensions of some formal description techniques can be taken as a schoolbook example. Despite the fact some of these formal methods have been available for about 15 years and that they never attracted reasonable interest from the industry, academic research continues and industrial criticism is still largely ignored.

Even though formal methods have already been used in the communications industry as we have shown in our survey, it should be noted that the vast majority of these projects have been carried out in the R&D departments. We are still far away from the integration of formal methods into the mainstream development process. When considering existing FMs, only SDL can be said to have gained wide acceptance in the industry.

The industrial concerns discussed in the following show some of the major barriers for the application of FMs in the communications industry. They have been identified in many discussions with our industrial partners.

4.1 Focus: Implementation vs. Abstract Models

Industrial statement 1 *The only thing that counts in the end, is the final implementation and not an abstract model.*

The Communications industry, especially, frequently argues that proving the correctness of abstract models does not pay since it does not provide a guarantee that the proven properties are preserved in the actual implementation. One of the key characteristics of communication systems is 24h × 7d operation. For example, a switching system is supposed to be available for all but two hours within a 40-year period [4]. These systems are seldomly restarted which leaves them particularly vulnerable to run-time problems such as memory faults. These problems are not related to the feature set of the systems, but to the implementation. They are very likely to pass system tests and occur in the field. The absence of any kind of sophistication or optimization in the design of algorithms and data structures in formal specifications, such as data packing, optimal coding, pointers, dynamic storage allocation and interrupts already leads to a less faithful representation of industrial communications services.

Of course, the relevance of any formal model to the actual running of the program is only as good as the degree of faithfulness to which the model represents real executions of the final program [65]. In [12] it is pointed out that the construction of a model close to real-life situations is very time-consuming. With respect to the models used for the detection of feature interactions, it is stated that most of these models are of an academic nature: they are very high-level, but a lot of interactions only appear when one takes the details of a real-world system into account. A similar conclusion is drawn in [38]. The authors of [74] point out that the models produced by applying existing formal methods often bear no relation to the actual (industrial) software.

By developing more and more faithful models that account for a larger set of phenomena associated with real executions, we approach more closely the final implementation. Obviously, there is a tradeoff between the degree of detail incorporated in the model and the complexity of using it [65]. Even though almost every newly developed formal model is compared to other formal models, there is very little discussion on how the different formal models reflect the reality they are supposed to represent.

Formal and analytical methods such as formal specification, verification, and systematic development will be hard to transfer into mainstream software development unless it is shown clearly how the effort of validating the mathematical model contributes to the quality of the final implementation. The generation of correctness-preserving implementations is a possible avenue to follow. At the moment, however, the generation of correctness-preserving implementations from validated design specifications has not yet matured to a level that satisfies the industry's requirements.

So far, we have seen very little in the way of model-reality studies – experimental or otherwise – that give confidence that the proposed validation techniques for abstract models have value for industrial implementations.

4.2 Size: Very Large Code vs. Toy Examples

Industrial statement 2 *Most industrial communication systems are enormous and significantly more complex than most academic examples.*

Most academic examples are not easily extrapolated to industrial size systems. For example, the development of a portion of the software in the 5ESS switch took about 100 man-years [42]. While almost every formal approach has been shown to work well for the Dining Philosophers there are few examples in which formal methods have been applied to large industrial communication systems. Even for non-trivial examples, it should be kept in mind that, for the advocates of a given formal approach, it is quite easy to construct or select a suitable application. Evidencing the feasibility of a formal approach on an industrial service (developed in and by industry) is significantly different from showing its feasibility on a self-defined example that will most likely be chosen such that it works well in the final analysis.

Down-scaling is a frequently applied method for making the verification of large systems practically feasible. Many errors in a service that will involve thousands of users when actually deployed can be found using a small number of users during the verification process. In order to deal with systems of different sizes, formal specifications should be parameterizable.

Additionally, to make formal verification feasible it is common practice to raise the abstraction level. However, raising the abstraction level often leads to an unfaithful representation of the actual system: the more abstract the model, the further away it is from reality.

Model checking is often used for verifying that a system satisfies its specification. However, model checking requires examination of all reachable system states and therefore suffers from state space explosion. Despite the impressive progress that has been made in the model-checking community in the recent years, model checking is still computationally infeasible for systems that are represented at a lower abstraction level (with a faithful representation of the real system).

If applicability in the industry is one of the goals of a formal approach[4], showing the feasibility on an industry-provided example would help significantly in building up credibility.

4.3 Heterogeneous Environment

Industrial statement 3 *Many problems encountered are actually due to the uncertitude that arises from the heterogeneous environment.*

As the heterogeneity of the environment increases, more and more time has to be spent on checking whether or not the implemented service behaves correctly *in its environment.*

To account for the environment in which communication services are supposed to run, it would be beneficial to build formal models that faithfully represent the service and the environment. However, considering the enormous heterogeneity of today's communication systems, this is a rather tricky and, in most cases, unattainable goal.

In any case, reports about formal methods should explicitly state the assumptions and restrictions that have been imposed during the validation process so that it can be checked, whether or not a given implementation platform and the environment respect these assumptions. For example, a single assumption about the non-preemptive character of an operation that does not hold at a given platform can render the entire validation procedure useless. While, in homogeneous environments, the assumptions - if known - might be relatively easy to check, this is rarely true for the environment of today's communications systems; thereby providing a high barrier for the validation of highly abstract models in the communications industry.

4.4 Time-to-Market

Industrial statement 4 *We simply don't have the time to develop large formal specifications.*

[4] which is and should obviously not necessarily be the case for all research

Until recently, introducing new services in a telephone network was a slow process and the deployed services were offered for a rather long period. The lifetime of new communication services, especially that of Internet services and hybrid services, is much shorter than that of standard telecommunications services. Furthermore, compared to the past, the time-to-market of these services is significantly reduced. The increased pace in the development of communication services also influences the application of formal techniques to the development of communication services: As market pressure increases and time-to-market decreases, increased development time is hardly acceptable.

There are a few studies outside the domain of communication service that show that formal methods can reduce overall development time that confront reports in which formal methods projects came in over-time; thereby providing a very fuzzy idea about the actual costs of formal methods. Even though the advocates of formal methods frequently provide information about the size of the formal specifications they have developed, information about the time it took to develop these specifications is rarely included in their reports. A consequential and detailed provision of this kind of information would be a very important first step towards a more detailed investigation.

Based on the results of a collaborative project between Nortel and the University of Toronto [82], the author claims that the use of FMs can increase the quality of the software without lengthening the development cycle. Projects like the one described in [82], which investigate the usefulness of FMs by quantitatively analyzing productivity changes, are highly important for a better understanding of the real costs and benefits of FMs.

To improve the return-on-investment of formal methods, their application should provide as many benefits as possible to the users. Jackson and Wing state that the naive presumption that formalization is useful in its own right must be dropped [48]. Besides removing ambiguities, the provision of additional benefits could make formality more appealing to the industry: test case generation from formal specifications, verification of the formal model, the generation of correctness preserving implementations are just a few examples. Jackson and Wing [48] point out that the industry will have no reasons to adopt formal methods until the benefits of formalization can be obtained immediately, with an analysis that does not require further massive investment. They also note that existing formal methods, at least if used in the conventional manner, cannot achieve these goals.

Formal methods that can be used as add-ons in the normal development process might be favored by the industry. Whenever more confidence in the communication service is required (e.g., for safety-critical applications) - and if time permits - formal approaches can be added. Examples are provided by the work described in [2] and [77].

4.5 Tool Support

Industrial statement 5 *Any formal approach should be complemented with adequate tool support.*

In the NewCoRe project [42] three requirements were adopted for the integration of formal methods:

- The tools fit smoothly into the existing design environment.
- The usage of the tools requires minimal training.
- The tools deal effectively with the complexity of the application.

For the NewCoRe project, these requirements led to a decision in favor of SDL. The three identified requirements can be assumed to be general requirements for the applicability of formal methods and their tools in the industry.

Goguen [37] lists five requirements for the success of domain-specific tools:

1. a narrow, well-defined, well understood problem domain is addressed
2. there is a community of users with potential financial resources who understand the domain
3. the tool has a graphical user interface that is intuitive to the user community, embodying their own language and conventions,
4. the tool takes a large grain approach; rather than synthesizing procedures out of statements, it synthesizes systems out of modules; it may use a library of components and synthesize code for putting them together,
5. inside the tool is a powerful engine that encapsulates formal methods concepts and/or algorithms; it may be a theorem prover or a code generator; users do not have to know how it works, or even that it is there.

The first two requirements are definitely satisfied for communication services.

4.6 Industrial Development Process

Industrial statement 6 *Any formal approach has to fit well in the typical development process and has to be usable by engineers.*

Even in projects between industry and academia, formal methods are applied *by* the academic partner; very often with the simple goal of demonstrating that the application of formal methods is useful. However, we have seen little work with the goal of providing formal methods and tools that can be applied by engineers. The ultimate goal of formal methods research should be to develop approaches that will be applied by engineers in their daily work. Proving theorems and using paper and pencil are definitely not suited to catch the interest of the industry; they are almost immediately rejected by the industry and very likely doomed to fail in an industrial environment.

In any case, the use of notions and notations that engineers are used to will significantly improve the chances of a smooth integration of formal methods in industry. OMT (Object Modelling Technique), UML (Unified Modelling Language), MSC (Message Sequence Charts) are keywords software engineers are well familiar with. In recent years there has been a considerable research effort, e.g., [29] [51], into the connection between formal methods and typical development processes for software, e.g., combining the benefits of formal methods with mainstream development methods like Fusion or UML. This trend is likely to continue.

An important issue that has been addressed inadequately in FM research, is the iterative development of communication services. Formal models must be kept consistent with the changing system requirements and the code. The authors of [83] point out that this is possibly one of the biggest obstacles in applying formal modelling techniques and that it requires firm commitments from the entire development team.

5 Transferring Formality to Industry: An Example

Over the last four years several researchers at the Swiss Federal Institute of Technology in Lausanne have been working on a collaborative project with three industrial partners, Swisscom, Alcatel and Thomson. While most of today's academic formal methods research is based on "We (academics) will do it for you (industry) and you will see that it is helpful", the focus of our collaboration has been on providing an approach that can be used *by* the industry in their daily work.

Alcatel/Thomson recently adopted our proposal into one of their development platforms for object-oriented distributed applications (the PERCO platform [64]) providing strong evidence of the applicability of formal methods in the industry if their specific needs are not ignored.

In this section, we show how our proposal addresses the industrial concerns discussed in Section 4. It is outside the scope of this paper to give a detailed technical description that can be found elsewhere [25] [61] [63] [24] [62]. Technical details are only provided as far as it it necessary for our discussion of how and why our proposal caught the interest of the industry and is being integrated in an industrial platform.

In our project we focused on the validation of object-oriented distributed systems in general and CORBA (Common Object Request Broker Architecture) based systems, in particular. CORBA is the platform of choice in the TINA architecture and can also be used for the development of Internet services [71]. As extensions for fault-tolerance [35] and real-time [72] are being standardized, CORBA technology becomes even more appealing for the development of communication services.

The PERCO platform, designed and built at the Alcatel-Thomson Common Laboratory, is an industry-provided, CORBA based software platform that specifically addresses the requirements of highly available dependable autonomous systems. Its basis tenet is to use UML specifications to build as much of the application as possible, while integrating real-time and fault-tolerant properties at the architectural level. In order to check if the services developed with the PERCO platform behave correctly, they can be validated by using our proposal.

5.1 A Brief Description of our Approach to Service Validation

The general idea of our approach to service validation is as follows: We formally specify behavioral properties that the service under construction should satisfy. These properties are to be expressed in a formal model of the service. This formal service model does not contain a formal behavior description; formal model and formal property specification are thus complementary. The formal model is described in detail in [24]. For CORBA based services, a significant part of the formal model can be automatically (by means of a tool) derived from the IDL (Interface Definition Language) specification of the service. An IDL specification is written at a level of abstraction that makes it particularly suitable for providing a basis on which to express behavioral properties. The advantages of expressing properties at an IDL-like abstraction level are appealing: a property, making reference to the items of an IDL specification, inherits the implementation language independent character from IDL. The standardized mapping from IDL to implementation languages enables us to automate the process of finding all the IDL information at the implementation level. Behavioral constraints are to be specified using a formalism based on Linear-time Temporal Logic (LTL). The formally specified properties, even though expressed in an abstract model, are checked at the corresponding service implementation (written in C++, Java etc.) at service run-time.

See Figure 1 for an illustration of how our approach works: The identification and specification of important properties of the service under development are essential tasks in the development process and are carried out by software engineers at several steps in the service life cycle. Most of these properties are, explicitly or implicitly, stated in the development documents in an informal (or semi-formal) way. In our approach, the software engineer identifies the behavioral properties to be specified

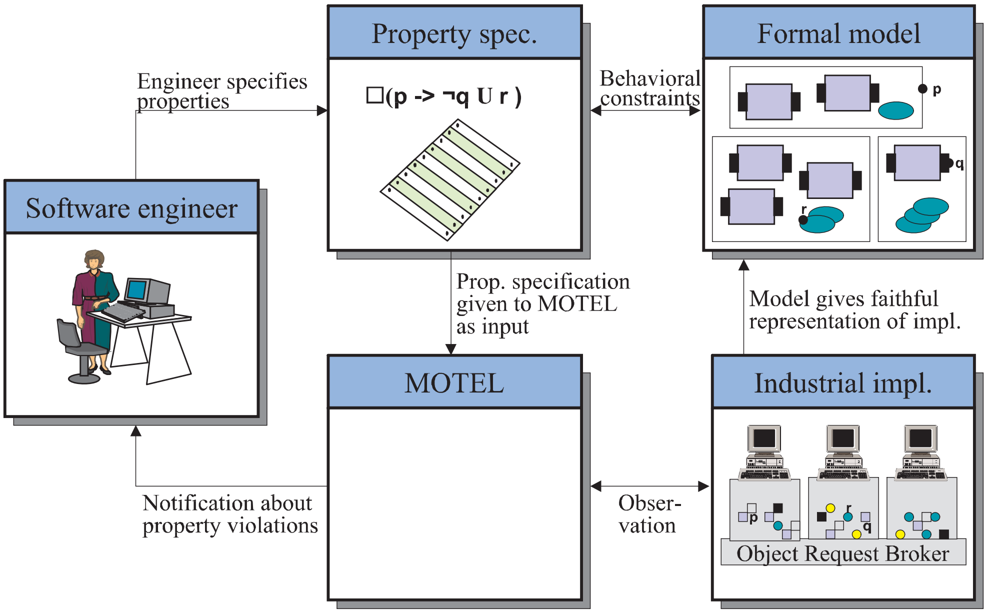
Property spec.
Formal model
Engineer specifies properties
□(p -> ¬q U r)
Behavioral constraints
p
q
r
Software engineer
Prop. specification given to MOTEL as input
Model gives faithful representation of impl.
MOTEL
Industrial impl.
Notification about property violations
Obser- vation
p
r
q
Object Request Broker

Figure 1: General Illustration of our Approach

formally according to their importance. The number of formally specified properties can be adjusted to the as needed.

To check whether or not the final implementation has violated or is violating the formally specified properties, it suffices to give the properties to a MOnitoring and TEsting tooL (MOTEL) that we have developed. MOTEL will automatically map the "model properties" to "implementation properties" and check at service run-time whether the implemented system is violating the properties expressed on our model. A description of MOTEL can be found in [63].

We express behavioral constraints independently of a specific implementation language by relying completely on a set of observable events. The idea of event-based behavioral abstraction has been frequently used (e.g., for testing [26] and debugging [3]), and is especially useful for accommodating heterogeneous platforms and multi-language programming environments. For our model, we provide a set of *predefined* events that is appropriate for modelling industrial-strength object-oriented systems. The events we use in our model, can be mapped in a straightforward manner to events as they occur during the execution of the final implementation. The set of events was determined by collaborating with several industrial players and by taking into account the trade-offs between flexibility and complexity of the model and the property language.

5.2 Relating our Proposal to the Industrial Concerns

In the following we will discuss how our approach to service validation addresses the aforementioned industrial concerns.

Industrial statement 1 The only thing that counts in the end, is the final implementation and not an abstract model.
Our solution *Formally specified properties are - even though expressed in an abstract service model - checked at the final implementation at service run-time.*

The framework for constructing formal models that is described in detail in [24], allows for the definition of formal models at an abstraction level that faithfully represents real executions of the final implementation. It has evolved through a large number of iterations with subsequent improvements and a number of discussions with our industrial partners. The starting point for the construction of our model was an investigation of today's development platforms for distributed services in industry. Rather than validating the properties in the abstract service model, we check them at the final implementation at service run-time, thus directly contributing to an increased confidence in the final implementation[5].

Industrial statement 2 Industrial communication systems are huge and significantly more complex than most academic examples.
Our solution *In a case study we have taken an industrial service as an example and we had no influence on the selection of the service.*

Our approach has been applied to a desktop video conferencing system developed by Swisscom, Dutch Telecom and Telia [25] and is currently being applied to an air traffic management system at Alcatel/Thomson. In both cases, the choice of application was outside of our control.

Industrial statement 3 Many problems encountered are actually due to the uncertitude that arises from the heterogeneous environment.
Our solution *The properties are checked during run-time of the final implementation.*

By checking whether or not the formally specified properties are respected by the final implementation at run-time, we automatically check if the service behaves correctly in its environment.

Industrial statement 4 We simply don't have the time to develop large formal specifications.
Our solution *Our approach places the focus on the formal specification of an arbitrary number of behavioral constraints.*

The development of large formal specifications is rarely feasible in industrial projects. However, engineers use mathematics to derive important properties of their proposed design [75]. The work described in [42, 50, 66] indicates that temporal logic can serve as one possible vehicle for the specification of correctness requirements. In our approach, the number of behavioral constraints can be adjusted to the needs; It is therefore possible to concentrate on a limited number of properties that are assumed to be important; It is also possible not to specify any properties. In such a case, the typical development process is not changed. The construction of the formal model in which the behavioral constraints are to be expressed is straightforward and can, to a large extend, be done automatically.

Industrial statement 5 Any formal approach should be complemented with adequate tool support.
Our solution *We have developed a MOnitoring and TEsting tooL (MOTEL).*

MOTEL encapsulates formal methods concepts and provides guidance and support for the specification of the properties. The properties specified with the guidance of MOTEL are then constantly checked while the final service implementation is observed at run-time and, if a property is violated by the service, an error message is given to

[5] Formal reasoning and the validation of the properties in the abstract model are still theoretically possible but have not been the focus of our investigation.

the user. A detailed description of MOTEL can be found in [63]. Using this prototype we were able to demonstrate the usefulness of our approach and convince our industrial partner to integrate it into his environment.

The manual use of formal techniques is restricted to the formal specification of the properties that the service under construction should satisfy. When formally specifying the properties, the property specifier is guided by MOTEL that allows him to assemble the different events and to temporally relate the events to each other. By using LTL for the specification of behavior, we can benefit from the well-known solutions for constructing test oracles. The generation of test oracles from LTL formulae is done automatically by MOTEL.

Industrial statement 6 Any formal approach has to fit well in the typical development process and has to be usable by engineers.
Our solution *The combination of our approach and UML is currently being investigated.*

One of the main characteristics of the platform currently developed by Alcatel and Thomson is the use of UML specifications to build as much of the application as possible. The automatic (or semi-automatic) generation of formal properties from UML documents is currently being investigated.

6 Conclusions

We have surveyed the formal methods used for communication services both in industry and academia and noted that few formal methods have attracted the attention of the industry. Only SDL can be said to have gained wide acceptance in the communications industry. Industry has shown considerable interest in Z, Promela and temporal logic whereas other formal methods like LOTOS are, at present, mostly academic tools with no or very little application to communication services in industry.

While formal methods have been applied to IN services and to TINA-based services, there has not been any significant investigation of formal methods for Internet services. This is very likely to change in the near future. The application of existing formal methods in the very flexible and fast-changing Internet services market seems to be extremely difficult; many formal methods are likely to fail here, at least if used in the conventional manner. Showing the benefits of formal methods for Internet services is definitely a very promising and challenging, yet untouched, research area.

We have listed and discussed major industrial concerns for the applicability of formal methods. These points, if seriously taken into consideration by the formal methods community, are likely to trigger some changes in the focus of formal methods research. While some peculiarities of communication services such as concurrency and real-time are well-understood today, others, such as the heterogeneous environment in which communication services run and the complexity of industrial communication services are inadequately addressed. We believe that the integration of the impressive results of formal methods research of the past into the industry will not be helped by yet another formal mechanism to model concurrency but by providing satisfying answers to the industrial concerns discussed in this paper.

We have briefly reported on a collaborative project between the Swiss Federal Institute of Technology, Swisscom, Alcatel and Thomson. The proposed approach to service validation is currently being integrated into an industrial development platform and will be used by engineers in their daily work. This successful transfer of formal methods to industry has been made possible by explicitly addressing the industrial concerns discussed in this paper.

Acknowledgements

The authors would like to thank the many researchers and engineers at Swisscom, Alcatel and Thomson for the interesting discussions. We thank H. Cogliati, S. Grisouard, S. Koppenhoefer, L. Logrippo, and P. Zave for their comments on the paper.

References

[1] M. Ardis, J. Chaves, L. Jagadeesan, P. Mataga, C. Puchol, M. Staskauskas, and J. von Olnhausen. A framework for evaluating specification methods for reactive systems – experience report. *IEEE Transactions on Software Engineering*, 22(6):378–389, June 1996.

[2] S. Aujla, T. Bryant, and L. Semmens. Applying formal methods within structured development. *Journal on Selected Areas in Communications*, 12(2):258–264, February 1994.

[3] P. Bates. Debugging heterogeneous distributed systems using event-based models of behavior. *ACM Transactions on Computer Systems*, 13(1):1–31, February 1995.

[4] Bellcore. LATA switching systems generic requirements (LSSGR). Bellcore, TR-TSY-000064, 1992.

[5] R. Bennett, J. Lindner, R. Michelsen, and D. Rypka. SDL in 5ESS switching system development. In *Proceedings of the 6th International Conference on Software Engineering for Telecommunication Switching Systems, Eindhoven, Netherlands*, April 1986.

[6] G. Blair, L. Blair, and J.-B. Stefani. A specification architecture for multimedia systems in open distributed processing. *Computer Networks and ISDN Systems, Special Issue on Specification Architecture*, 29:473–500, 1997.

[7] J. Blom, R. Bol, and L. Kempe. Automatic detection of feature interactions in temporal logic. In K. Cheng and T. Ohta, editors, *Feature interactions in Telecommunications Systems III*, pages 1–19. IOS Press, 1995.

[8] J. Blom, B. Jonsson, and L. Kempe. Using temporal logic for modular specification of telephone services. In L. Bouma and H. Velthuijsen, editors, *Feature Interactions in Telecommunications systems*, pages 197–213. IOS Press, 1994.

[9] P. Bosco, G. Martini, D. LoGiudice, and C. Moiso. ACE: An environment for specifying, developing and generating TINA services. In *Proceedings of the Fifth International Symposium on Integrated Network Management, San Diego, CA, USA*, May 1997.

[10] L. Bouma and Velthuijsen, editors. *Feature Interactions in Telecommunications Systems.* IOS Press, 1994.

[11] L. Bouma and J. Zuidweg. Formal analysis of feature interactions by model checking. In *Proceedings of the First International Workshop on Feature Interactions in Telecommunications Systems, St. Petersburg, FL, USA*, December 1992.

[12] W. Bouma, W. Levelt, A. Melisse, K. Middelburg, and L. Verhaard. Formalization of properties for feature interaction detection: Experience in a real-life situation. In H.-J. Kugler, A. Mullery, and N. Niebert, editors, *Towards a Pan-European Telecommunication Service Infrastructure – IS&N'94*, number 851 in Lecture Notes in Computer Science, pages 393–405. Springer-Verlag, 1994.

[13] K. Braithwaite and J. Atlee. Towards automated detection of feature interactions. In *[10]*, pages 36–59. IOS Press, 1994.

[14] C. Capellmann, P. Combes, J. Pettersson, B. Renard, and J. Ruiz. Consistent interaction detection – a comprehensive approach integrated with service creation. In *[27]*, pages 183–197. IOS Press, 1997.

[15] CCITT Recommendation Z.100. *Specification and Description Language SDL.* CCITT SG X, Contribution Com X-R15-E, 1987.

[16] K. Cheng. Towards a formal model for incremental service specification and interaction management support. In *[10]*, pages 152–166. IOS Press, 1994.

[17] K. Cheng and T. Ohta, editors. *Feature Interactions in Telecommunications Systems III.* IOS Press, 1995.

[18] E. Clarke and J. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, December 1996.

[19] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, and P. Jeremaes. *Object-oriented Development: The FUSION method.* Prentice-Hall, 1994.

[20] Score Consortium. Score deliverables. http://www.tdr.dk/public/ SoftwareEngineering/ServiceCreation/SCORE/score-deliverables.html .

[21] Score Consortium. Industry requirements on the service creation environment. Score Identifier D405, Available at [20], April 1994.

[22] Score Consortium. Score presentation (version 3). Score Identifier D408, Available at [20], December 1994.

[23] Score Consortium. Report on methods and tools for service creation (third version), volume i: Service interaction. Score Indentifier D206A, Available at [20], December 1995.

[24] F. Dietrich, X. Logean, and J.-P. Hubaux. Modelling and testing object-oriented distributed systems with linear-time temporal logic. Submitted to ”Theory and Practice of Object Systems”.

[25] F. Dietrich, X. Logean, and J.-P. Hubaux. Testing temporal logic properties in distributed systems. In A. Petrenko and N. Yevtushenko, editors, *IFIP International Workshop on Testing of Communicating Systems (IWTCS)*, pages 247–262, Tomsk, Russia, August 1998. Kluwer Academic Publishers.

[26] L. Dillon and Q. Yu. Oracles for checking temporal properties of concurrent systems. In *Proceedings of the 2nd ACM SIGSOFT Symposium on Foundations of Software Engineering*, volume 19, pages 140–153, December 1994.

[27] P. Dini, R. Boutaba, and L. Logrippo, editors. *Feature Interactions in Telecommunication Networks IV.* IOS Press, 1997.

[28] A. Eberlein, M. Crowther, and F. Halsall. Development of new telecommunications services using an expert system. *BT Technology Journal*, 15(1):217–222, January 1997.

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

[30] M. Faci and L. Logrippo. Specifying features and analysing their interactions in a LOTOS environment. In *[10]*, pages 136–151. IOS Press, 1994.

[31] M. Faci, L. Logrippo, and B. Stépien. Formal specification of telephone systems in LOTOS: The constraint-oriented approach. *Computer Networks and ISDN Systems*, pages 53–67, 1991.

[32] M. Faci, L. Logrippo, and B. Stépien. Structural models for specifying telephone systems. *Computer Networks and ISDN Systems*, pages 501–528, 1997.

[33] *Proceedings International Workshop on Feature Interactions in Telecommunications Software Systems*, St. Petersburg, FL, December 1992.

[34] A. Flora-Holmquist, J. O'Grady, and M. Staskauskas. Telecommunications software design using virtual finite state machines. In *Proceedings of the International Switching Symposium (ISS95), Berlin, Germany*, pages 103–107, April 1995.

[35] Fault-Tolerant CORBA, Draft Document, OMG, 1999.

[36] R. Glass. Formal methods are a surrogate for a more serious software concern. *IEEE Computer*, pages 19–20, April 1996.

[37] J. Goguen and Luqi. Formal methods and social context in software development. In *TAP-SOFT'95: 6th International Conference on Theory and Practice of Software Development*, number 915 in Lecture Notes in Computer Science, pages 62–81. Springer-Verlag, May 1995.

[38] J.-Ch. Grégoire and M. Ferguson. Neglected topics of feature interactions: Mechanisms, architectures, requirements. In P. Dini, R. Boutaba, and L. Logrippo, editors, *Feature Interactions in Telecommunication Networks IV*, pages 3–12. IOS Press, 1997.

[39] D. Gries. The need for eduation in useful formal logic. *IEEE Computer*, April 1996.

[40] N. Griffeth and Y.-J. Lin. Extending telecommunications systems: The feature-interaction problem. *IEEE Computer*, 26(8):14–18, August 1993.

[41] G. Holzmann. *Design and Validation of Computer Protocols.* Prentice-Hall, 1991.

[42] G. Holzmann. The theory and practice of a formal method: NewCoRe. In *Proceedings of the IFIP World Computer Congress*, volume I, pages 35–44, Hamburg, Germany, August 1994. North-Holland Publ., Amsterdam, The Netherlands.

[43] G. Holzmann and J. Patti. Validating SDL specifications: An experiment. In *Proceedings of the Ninth International Workshop on Protocol Specification, Testing and Verification*, 1989.

[44] J.-P. Hubaux, C. Gbaguidi, S. Koppenhöfer, and J.-Y. Le Boudec. The impact of the internet on telecommunications architectures. *Computer Networks and ISDN Systems*, 1998. To appear.

[45] ISO IS 8807. *LOTOS – A formal description technique based on the temporal ordering of observational behavior.* ISO/TC97/SC21, November 1988.

[46] ISO IS 9074. *Estelle - A formal description technique based on an extended state transition model.* ISO/TC97/SC21, 1989.

[47] ITU-T. General recommendations on telephone switching and signalling, Intelligent Network – Q-Series Intelligent Network Q.1200 - Q.1290, 1993.

[48] D. Jackson and J. Wing. Lightweight formal methods. *IEEE Computer*, pages 21–22, April 1996.

[49] M. Jackson and P. Zave. Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering*, pages 831–847, October 1998.

[50] L. Jagadeesan, C. Puchol, and J. Olnhausen. A formal approach to reactive systems software: A telecommunications application in ESTEREL. *Journal of Formal Methods in System Design*, 1995.

[51] Jean-Marc Jézéquel, Alain Le Guennec, and François Pennaneac'h. Validating distributed software modeled with UML. In Pierre-Alain Muller and Jean Bézivin, editors, *Proceedings of UML'98 International Workshop, Mulhouse, France, June 3 - 4, 1998*, pages 331–340. ESSAIM, Mulhouse, France, 1998.

[52] Y. Kawarasaki and T. Ohta. A new proposal for feature interaction detection and elimination. In *[17]*, pages 127–139. IOS Press, 1995.

[53] J. Keller and O. Dubuisson. Formal description of OSI management information structure as a prerequisite for formal specifications of TMN interfaces. In *Towards a Pan-European Telecommunication Service Infrastructure – IS&N'94*, number 851 in Lecture Notes in Computer Science, pages 433–442. Springer-Verlag, 1994.

[54] B. Kelly, M. Crowther, J. King, R. Masson, and J. DeLapeyre. Service validation and testing. In *[17]*. IOS Press, 1994.

[55] B. Kitson, P. Leydekkers, N. Mercouroff, and F. Ruano. *TINA Object Definition Language (TINA-ODL) Manual 1.3.* TINA-C, June 1995.

[56] F. Koch. Spezifizierung offener verteilter Systeme aus Sicht des ODP Computational Viewpoint. GMD-Studien Nr. 243, Gesellschaft für Mathematik und Datenverarbeitung, October 1994.

[57] E. Koerner and L. Strick. Applying LOTOS to the design of TINA applications. In H. Bowman and J. Derrick, editors, *Proceedings of Second IFIP International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS97)*, pages 455–466. Chapman & Hall, 1997.

[58] D. Kuhn. Sources of failure in the public switched telephone network. *Computer*, 30(4):31–36, April 1997.

[59] L. Lamport. TLA in pictures. *IEEE Transactions on Software Engineering*, pages 768–775, September 1995.

[60] F. Lin and Y.-J. Lin. A building block approach to detecting and resolving feature interactions. In *[10]*. IOS Press, 1994.

[61] X. Logean, F. Dietrich, and J.-P. Hubaux. TINA service validation: The ErnesTINA project. In *IEEE ICC Conference*, Atlanta, June 1998.

[62] X. Logean, F. Dietrich, J.-P. Hubaux, S. Grisouard, and P.-A. Etique. On applying formal techniques to the development of hybrid services: Challenges and directions. *IEEE Communications Magazine*, 37(7):132–138, July 1999.

[63] X. Logean, F. Dietrich, and S. Koppenhoefer. Run-time monitoring of distributed applications. In *Proceedings of Middleware'98, Lake District, England*, September 1998.

[64] J. Maisonneuve, S. Chabridon, and P. Leveillé. The PERCO platform. In *ISORC'99, St. Malo, The 2nd IEEE International Symposium on Object-oriented Real-time distributed Computing*, May 1999.

[65] Z. Manna and A. Pnueli. On the faithfulness of formal models. In *Mathematical Foundations of Computer Science*, number 520 in Lecture Notes in Computer Science, pages 28–42. Springer-Verlag, 1991.

[66] C. Middelburg. A simple language for expressing properties of telecommunication services and features. Technical Report 94-PU-356, PTT Research, April 1994.

[67] H. Mulder. *TINA-C Glossary of Terms, Version 2.0.* TINA-C, January 1997.

[68] M. Nakamura, Y. Kakuda, and T. Kikuno. Petri-net based detection method for non-deterministic feature interactions and its experimental evaluation. In P. Dini, R. Boutaba, and L. Logrippo, editors, *Feature Interactions in Telecommunication Networks IV*, pages 138–152. IOS Press, 1997.

[69] T. Ohta and Y. Harada. Classification, detection and resolution of service interactions in telecommunication services. In *[10]*. IOS Press, 1994.

[70] G. Parkin and S. Austin. Overview: Survey of formal methods in industry. Technical report, National Physical Laboratory, Teddington, Middlesex, U.K., May 1993.

[71] IONA Technologies PLC. *OrbixWeb Programmer's Guide*, September 1998.

[72] Real-Time CORBA, Joint Revised Submission. Available at ftp://ftp.omg.org/pub/docs/orbos/98-12-10.pdf, December 1998.

[73] R. Saracco, J. Smith, and R. Reed. *Telecommunications System Engineering using SDL*. North-Holland, 1989.

[74] R. Sinnott and M. Kolberg. Engineering telecommunication services with SDL. In *Proceedings of the Conference on Formal Methods for Open,Object-based Distributed Systems (FMOODS'99)*, 1999.

[75] IEEE Computer Society. Computer magazine, April 1996.

[76] IEEE Computer Society. Communications magazine, July 1999.

[77] B. Steffen, T. Margaria, A. Claßen, V. Braun, and M. Reitenspieß. A constraint-oriented service creation environment. In *PACT'96, 2nd International Conference on Practical Application of Constraint Technology, London, UK*, 1996.

[78] B. Stepien, K. Farooqui, and L. Logrippo. An experience modelling telecommunications systems using ODP-DLcomp. In E. Najm and J.-B. Stefani, editors, *Formal Methods for Open Object-based Distributed Systems*, pages 221–228. Chapman & Hall, 1997.

[79] Time-Rover. http://www.time-rover.com.

[80] TINA-C. http://www.tinac.com.

[81] G. Vanecek, N. Mihai, N. Vidovic, and D. Vrsalovic. Enabling hybrid services in emerging data networks. *IEEE Communications Magazine*, 37(7):102–109, July 1999.

[82] A. Wong. Formalizing requirements in a commercial setting: A case study. Master's thesis, University of Toronto, Graduate Department of Computer Science, 1999.

[83] A. Wong and M. Chechik. Applying formal methods to a telecommunications system in a commercial setting. In *11th International Conference on Software Engineering & its Applications, Paris, France*, November 1998.

[84] P. Zave. Secrets of call forwarding: A specification case study. In G. Bochmann, R. Dssouli, and O. Rafiq, editors, *Formal Description Techniques VIII*, pages 169–184. Chapman & Hall, 1996.

[85] P. Zave. Formal description of telecommunication services in Promela and Z. In *Proceedings of the Nineteenth International NATO Summer School*, 1999.