On Applying Formal Techniques to the Development of Hybrid Services: Challenges and Directions

Xavier Logean, Falk Dietrich, and Jean-Pierre Hubaux, Swiss Federal Institute of Technology

Sylvain Grisouard, Alcatel

Pierre-Alain Etique, Swisscom

ABSTRACT We are primarily interested in formal techniques and how they are applied to the development of hybrid services in particular. We analyze the peculiarities of such services, look at the use of formal techniques for communication services in the industry, and highlight some of the major concerns for the application of formality in an industrial environment. It is argued that with the introduction of hybrid services, more pragmatism is required in applying formal techniques. We briefly describe an ongoing joint collaboration of Alcatel, Swisscom, and the Swiss Federal Institute of Technology in which formal techniques are applied to the specification and testing of hybrid services.

The design and implementation of communication services is a complex task. Entire communication systems can be brought to their knees by an unintended feature interaction, or a misimplemented piece of the greater puzzle. This puzzle is likely to become even more complicated in the future with the emergence of *hybrid services*. A hybrid service is defined as a service that spans many network technologies [1], especially the public switched telephone network (PSTN), cellular networks, and IP networks. 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.

With installations that will soon reach one billion wireline phones, 300 million mobile phones, and 200 million IP hosts, ensuring hybrid service reliability is one of the most challenging tasks in software engineering. Many academics see formality (formal specifications, formal methods, formal description techniques, etc.) as an inevitability for a high degree of software reliability. However, most practitioners see formal methods as irrelevant to what they do [2]. The introduction of hybrid services offers new challenges that can also be seen as new opportunities for the application of formal methods.

The contribution of this article is threefold:

- In the next section we examine a number of projects in which formal methods were applied to the industrial development of communication services. Although there are a number of excellent surveys about formal methods in general (e.g., [3]), there are currently none that address the specific area of communication services. In our survey we focus on projects with heavy industrial involvement. We indicate that, although there are several projects that applied formal methods to communication services, no attention has been paid to hybrid services yet.
- We then identify the peculiarities of communication services in general and hybrid services in particular. We analyze the impact of these characteristics on the applicability of formal techniques in practice. We illustrate that hybrid services, even though they constitute a subset of communication services, have some specific peculiarities that impose strong requirements on the applicability of formal techniques in industrial projects.
- Finally, we briefly report on an ongoing project in which we investigate the applicability of formal techniques during the development of hybrid services.

Formal Techniques for Communication Services

In this section we look at the application of formal techniques to the design of communication services in the industry and focus exclusively on a number of projects that we consider to be of substantial interest to a larger community.

Many claim to formally specify and validate communication services. The number of formalisms that purportedly facilitate the specification of communication services has exploded over the past few years. These approaches differ considerably from one another, and are difficult to compare and judge, especially since their limitations are rarely mentioned by their advocates.

Even though the extensive research on formal techniques has led to impressive academic results, far too often it does not solve the industry's problems. After more than two decades of active research into development of formal methods and their applications to distributed systems and communication protocols, there still seems to be a lack of industrial take-up. Some of the barriers to their use by developers in the industry include developers' lack of confidence in formal methods, industrial-strength tools not readily available, the lack of information on potential benefits of using formal techniques, and the small number of reports on the application of formal techniques to real-life complex examples.

Compared to traditional methods, however, formal service development can offer a number of advantages; for instance, specifications can be verified, implementation automated, and test cases generated.

Formal methods are normally applied to ensure:

- Unambiguity of specifications
- Completeness of specifications
- Consistency of specifications
- Conformance of implementations to specifications

The shaded box on the following page lists some of the most frequently used formal techniques. Almost every formal technique has been applied to communication services. How-

0163-6804/99/$10.00 © 1999 IEEE

ever, the (tele)communications community has been paying special attention to the three standardized formal description techniques (FTDs) LOTOS, Estelle, and SDL. Originally developed to unambiguously specify protocols, they were standardized about 15 years ago and have been applied successfully to 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, for example, for real time and object orientation. Consequently, there is almost more work *on* formal methods than *with* them. Unfortunately, only SDL has gained wider acceptance in industry, and the vast majority of formal methods projects are carried out with little or no industry involvement.

Table 1 lists some projects in which formal methods have been applied to the design of communications services in the industry that we consider important for a number of reasons to be discussed later. Most of these projects make use of a modeling language to specify the service behavior, a property specification language to express behavioral constraints the service should satisfy, and validation techniques like model checking. Unless otherwise stated, the formal technique listed in Table 1 indicates the modeling language.

At Lucent, several formal method projects aimed at designing and implementing software for the Lucent 5ESS (Electronic Switching System) and several formal methods have been applied over a longer time period [4, 5].

Lucent	Esterel	[5]
Dutch PTT	ACTL*	[6]
SNI	FSM	[7]
BT	Z	[8]
CSELT	Promela	[9]

Table 1. *Formal techniques in the communications domain.*

The NewCoRe project described in [4] ran over a two-year period. A specification of 7500 lines of (noncommented) 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 knowledge, this is still true.

Jagadeesan, Puchol, and Olnhausen [5] describe a technique and the supporting tools that allow automatic verification of whether Esterel programs satisfy safety properties. Safety properties state that, no matter what inputs are given and no matter how choices are resolved inside the system, the system will not misbehave (e.g., emission of undesirable outputs, undesirable modes of operation being reached).

Since neither SDL nor Esterel provides a means to make statements about the correctness of a design independent of the design itself, in both [4, 5], correctness requirements for the 5ESS switching software were expressed in temporal logic.

The weakness of SDL for expressing correctness requirements has also been noticed by Middelburg [6], who proposed using a temporal logic based on ACTL* to express properties for communication services.

Temporal logic has also been used in the project described in [7], which deserves special attention since it describes work that made the step into a commercial product. Siemens Nixdorf Informationssysteme, Munich, Germany, and the University of Passau, Germany, developed an environment for the creation of intelligent network (IN) services. Services described in this framework can be formally verified by model checking. While the service is modeled using FSMs, 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 what degree he is willing to invest in formality. 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 [8]. 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.

CSELT developed the Application Construction Environment (ACE) [9] for the specification, development, and generation of Telecommunications Information

FORMAL METHODS

FSM (finite state machine). A simple finite state machine is composed of states connected by transitions. The relative simplicity of FSMs makes them especially suitable for formal analysis.

SDL (Specification and Description Language) is based on an extended FSM 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 which 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 bidirectional channels between their communication ports.

Esterel is a synchronous programming language 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 (nonexecutable) 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 which 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 and TLA (the Temporal Logic of Actions), but they have received rather limited attention from the (tele) communications community.

Networking Architecture (TINA) services. Besides containing a family of graphical editors, it also includes a compiler which translates (intermediate code of) the specification into Promela. The SPIN tool is then used as a model checker.

Whereas there has been extensive work toward the validation of services in the IN and TINA services, there has not been much work on the application of formal methods to the development of Internet services or even hybrid services. The obvious questions are:

- Are Internet services and hybrid services any different from other (tele)communication services?
- If so, what do the differences mean for the application of formal techniques?

The Peculiarities of Hybrid Services

We investigate the characteristics of communication services. This is followed by an analysis of hybrid service peculiarities; we discuss the impact of these peculiarities on the application of formal techniques in industrial projects.

The most important characteristics of communication services are summarized in the shaded box on this page. Most of these peculiarities have been addressed in research on formal methods. For example, there is a large body of research dealing with interoperability issues.

In addition to the peculiarities of general communication services, the introduction of hybrid services renders the following characteristics important:

- Interworking of connection-oriented and connectionless services
- Integration of network-centric and terminal-centric services
- Decreased service lifetime and time to market
- Significantly increased heterogeneity

We argue that these additional characteristics change the way formal methods should be applied to the design of services, especially if industrial applicability is of major importance. In the following sections we will briefly look at the four aforementioned characteristics of hybrid services and discuss the impact these characteristics have on the applicability of formal techniques in an industrial setting.

Interworking of Connection-Oriented and Connectionless Services — Hybrid services combine connection-oriented and connectionless techniques; there is no commonly accepted call model for hybrid services. Much of the work on formal methods in the telecommunications community is based on specific call models, such as those used in the IN. As long as formal methods were applied to standardized architectures such as the IN in which all services were structured in a similar way (e.g., by using service-independent building blocks), the application and reuse of formal approaches was significantly easier.

The lack of a common call model for hybrid services implies that most of the work of applying formal techniques to (tele)communication systems has to be revised and checked to see whether and how it can be reused/adapted for hybrid services.

Integration of Network-Centric and Terminal-Centric Service Control Mechanisms — In the Internet world, services are implemented in end users' systems, whereas the telecommunications community normally has a network-centric vision where services are almost entirely implemented in the network. These two different views of network-centric and end-system-centric service control may converge to a *service-centric* vision for the deployment of hybrid services [1].

For the use of FMs in development of hybrid services, it is most probably necessary to consider software running at the user's site and in the network.

Decreased Service Lifetime and Time to Market — Until recently, introducing new services in a telephone or cellular network was a slow process, and the deployed services were offered for a rather long period. Compared to typical telecommunication services, the time to market of Internet and hybrid services is significantly reduced. The increased pace in the development of hybrid services influences the application of formal techniques on the development of hybrid services in several ways:

- As market pressure increases and time to market decreases, increased development time is hardly acceptable.
  - –Formal specifications that have

The Peculiarities of Communication Services

Concurrency, distribution, and reactivity. Communication services obviously belong to the class of concurrent, distributed, and reactive systems. A reactive system is a system that maintains an ongoing interaction with its environment, as opposed to computing some final value on termination.

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

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

Reliability, availability, and fault tolerance. Only marginal downtime 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.

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

Heterogeneous environment. Probably the most important peculiarity of (tele)communication services that singles them out from the set of other complex, real-time software (e.g., for nuclear plants and aircraft) is the environment, for both system production and system operation. The tremendous heterogeneity of today's communication systems is a result of their evolutionary development and the perpetual introduction of new services and technologies.

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

only been developed to "remove ambiguities" do not seem appealing enough to be integrated into the mainstream development process. The use of formal techniques should provide additional benefits, thereby making formality more appealing to industry.

–The use of formal techniques in practice, especially under strict deadlines, requires that they can easily be integrated into the usual service development process.

In recent years there has already been a considerable research effort 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) [10]. This trend is likely to continue.

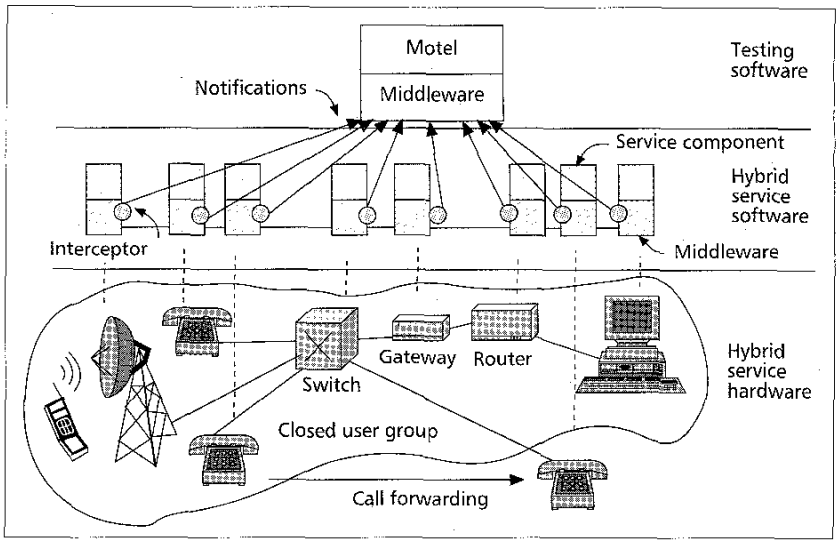


Figure 1. *MOTEL for testing a hybrid closed user group service.*

- The development of large formal specifications is rarely feasible for hybrid services. In an industrial environment, it seems to be more promising to formally express single properties with which a service should comply, rather than developing large abstract service specifications. The work described in [6, 7] provides some evidence that industry thinks this is a promising avenue to follow.
- Formal techniques should be usable as add-ons in the normal development process. Whenever more confidence is required (e.g., for safety-critical applications) and time permits, formal approaches can be added. Examples of this can be seen in the work described in [7, 8].

Significantly Increased Heterogeneity — It is frequently argued, especially in the (tele)communications industry, that proving the correctness of abstract models is not beneficial because it does not guarantee that the proven properties will be preserved in the actual implementation. 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. Another example of the impact of heterogeneity is the problem of service interactions (see the fourth section and Fig. 1). A service interaction occurs when the addition of a new feature (e.g., introducing a new service) to a system disrupts the existing services. In most cases one wants to ensure that the behavior of a service does not change when associated with other (supposedly noninteracting) services.

Whereas in homogeneous environments the assumptions are relatively easily defined and checked, this is rarely true for today's telecommunications systems and definitely not true for hybrid services. In addition, building a formal model that accurately represents the service *and the environment* is a rather tricky, and in most cases even unattainable, goal.

To account for the immense heterogeneity of the environment in which hybrid services run, the following points have to be taken into account for the applicability of formal techniques:

- As heterogeneity increases, more and more time has to be spent checking whether the implemented service behaves correctly in its environment. Our discussions with industry led us to conclude that many errors of the final service implementations are actually due to the uncertainty that arises from a heterogeneous environment.
- Rather than validating abstract models (with a large list of hidden assumptions), it seems more beneficial (cost-effective and faster) to focus on the actual implementation, or at abstraction levels that accurately represent the real implementations.
- Given a competitive market and a multiple service provider environment, there might be a limited willingness to publish or share detailed information about service specifications. It is therefore very unlikely that all service specifications would be available to the validating party. This leads us to conclude that there has to be a shift of focus to feature interaction detection, particularly at runtime.

A Monitoring and Testing Tool

In a collaborative work with the Swiss Federal Institute of Technology at Lausanne, Swisscom R&D in Bern, Switzerland, and Alcatel Corporate Research in Marcoussis, France, we are looking at the possibility of making formality more appealing to industry. This collaboration is the continuation, in the area of hybrid services, of a joint project between Swisscom (formerly Swiss Telecom PTT) and the Swiss Federal Institute of Technology at Lausanne on the specification and validation of telecommunication services in the framework of the IN [11].

Specifically, the purpose of the collaboration is to find a method and/or formalisms under the following constraints:

- As addressed in the previous section, with the peculiarities of hybrid services the use of formal techniques should, as much as possible, contribute to the quality of the actual implementation; proving the correctness of highly abstract models is not desirable.
- The time-consuming development of large formal specifications should be avoided. Formal techniques can be used as add-ons in the normal development process; if desired, they can be gradually introduced. Also, the approach should scale well.
- As developed in the analysis of the *heterogeneity* peculiarity of hybrid services, our approach should be applicable to telecommunication services, Internet services, and, above all, to hybrid services such that the investment made in the development of the method and tools is amortized.
- The proposed method has to fit in an engineering environment (e.g., proving theorems is not acceptable), as addressed in the *service lifetime and time-to-market* peculiarity.
- There should be adequate tool support.

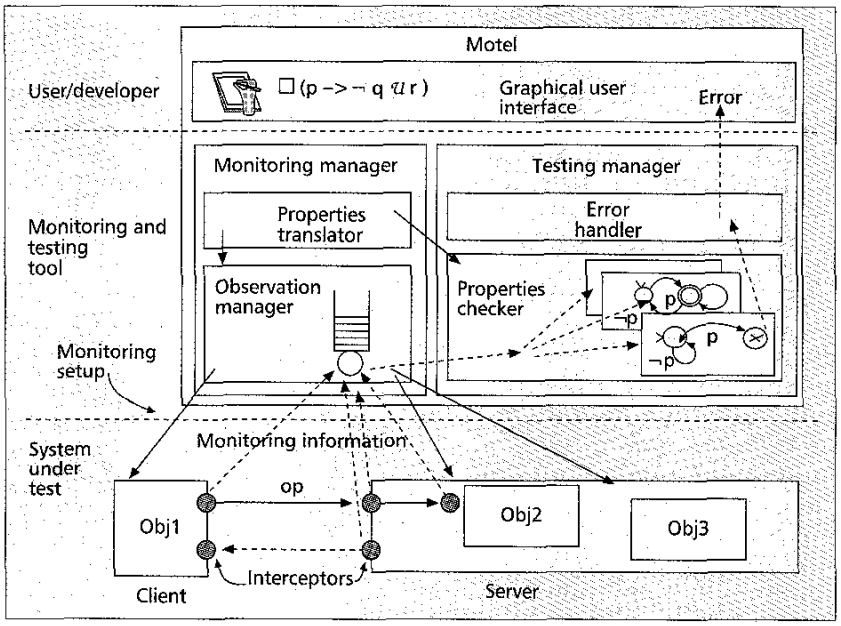
Toward the *first* goal, we consider the actual implementation that will be deployed in the network and end-user termi-

nals as the system under scrutiny. In our approach, we use formality to express behavioral constraints (properties) that the implemented service should satisfy. These formally specified properties are then constantly being observed at runtime, and all property violations are reported.

For the *second* goal, we focus on the formal specification of an arbitrary number of behavioral constraints that the system under development should have. The number of behavioral constraints can be adjusted to the needs. It is possible not to specify any properties. In such a case, the typical development process is not changed.

With regard to the *third* goal, we express behavioral constraints independent of a specific implementation language by relying completely on a set of predefined observable events. This set has been 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. The set of events is chosen very carefully, making it possible to perform source code annotation for event generation in an *automatic* manner. The abstraction level we achieve, in our proposal, through event-based system modeling makes it straightforward to link our model to a number of development platforms and implementation languages.

As for the *fourth* goal, the manual use of formal techniques is restricted to the specification of the properties that the service under construction should satisfy. Typical communications systems developers are neither trained in, nor comfortable with, the formality and mathematical sophistication of most FMs. Therefore, when formally specifying behavioral constraints, the property specifier is guided by a tool that allows him to assemble the different events and temporally relate the events to each other. For the establishment of a temporal relationship between the events, we advocate the use of linear-time temporal logic (LTL) (see the shaded box on this page). By using LTL for the specification of behavior, we can benefit from the well-known solutions for constructing test oracles.



■ **Figure 2.** *MOTEL's structure.*

Finally, concerning the *fifth* goal: we have developed the Monitoring and Testing Tool (MOTEL). MOTEL encapsulates formal methods concepts, and provides guidance and support for the specification of the properties.

Validating hybrid services with MOTEL consists of:

- Expressing the properties reflecting the behaviors the service developer wants to test specified in the service requirements.
- Applying an automatic instrumentation technique on the service implementation in order to spy on the service at runtime. The provision of a hybrid service is the result of interactions among service components that reside on service platform elements, especially user terminals, network nodes, control nodes, and gateways between different network technologies. These service components communicate via *middleware*, such as Java Remote Management Interface (RMI) [1] or Common Object

TEMPORAL LOGIC

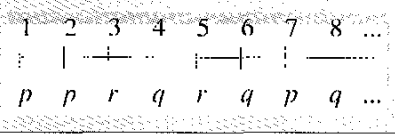
Modal logic was originally conceived as the logic of necessity and possibility, and developed by philosophers to study different "modes" of truth. *Temporal logic* is a special type of modal logic used for describing and reasoning about how the truth values of assertions change over time.

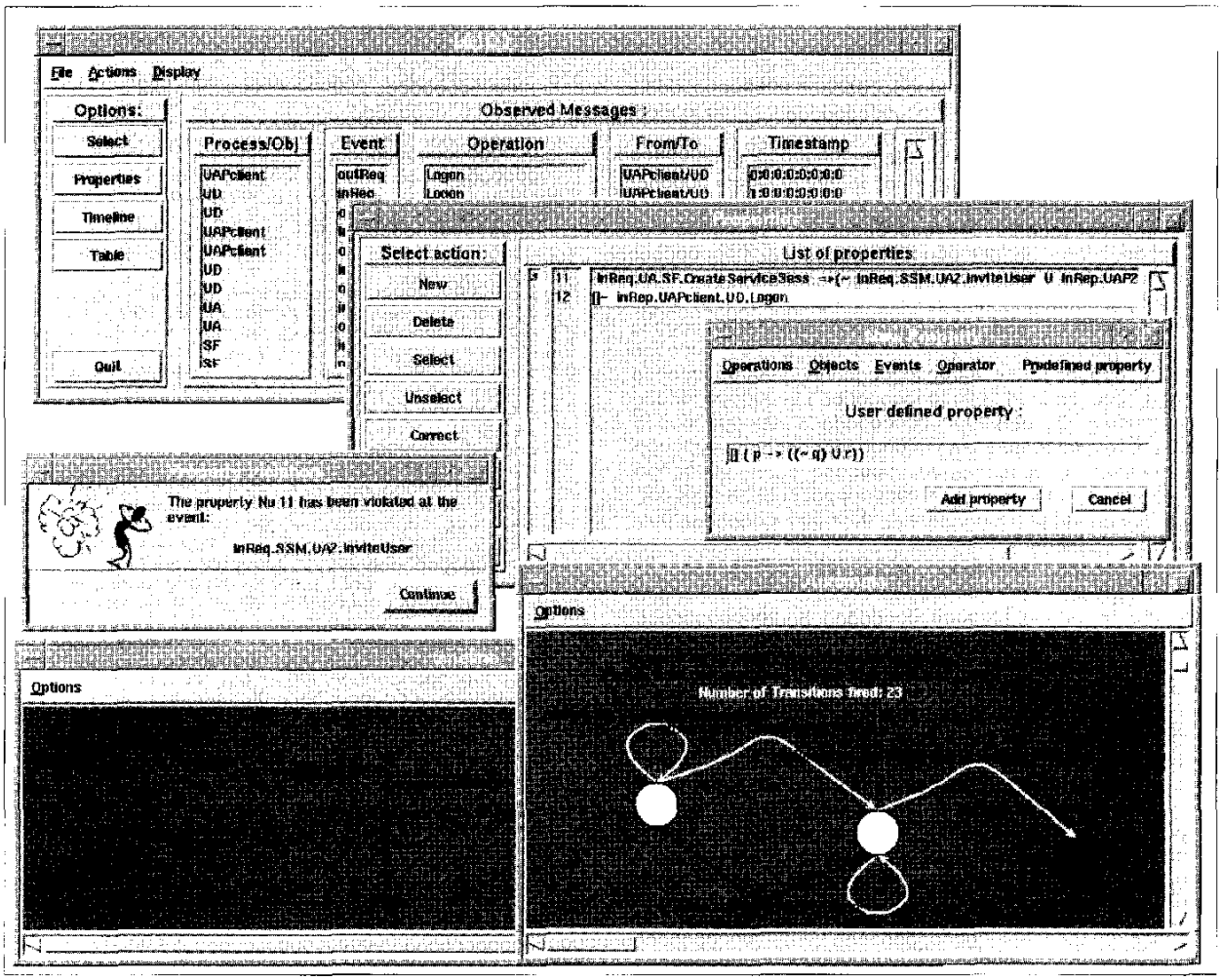
With respect to temporal logic, time can be regarded in two different ways. One is that the course of time is linear (at each moment there is only one possible future moment). The other is that time has a branching (tree-like) structure: at each moment, time may split into alternate courses representing different possible futures. Accordingly, the temporal logics are called either *linear-time temporal logic* or *branching-time temporal logic*.

Linear-time temporal logic provides a number of operators that refer to the future or to the past. For illustration consider the two future operators □ (always) and $\mathcal{U}$ (until): □p (reads always p or henceforth p) states that p is true now and that it will always be true in the future; $p\ \mathcal{U}\ q$ (reads $p\ \mathcal{U}\ q$) states the eventual occurrence of q and says that p holds continuously at least until the first occurrence of q.

Example: The formula □$(p \rightarrow \neg q\ \mathcal{U}\ r)$ states that the subformula $(p \rightarrow \neg q\ \mathcal{U}\ r)$ is always true. The subformula holds at a position i if either (1) p is false there or (2) p is true and, starting at position i, not q holds continuously until r becomes true. This formula can be used to formally state that each occurrence of p will trigger an occurrence of r before q arrives.

In the following timeline with the occurrences of p, q, and r, the property □$(p \rightarrow \neg q\ \mathcal{U}\ r)$ would be violated at position 8 with the occurrence of q (an occurrence of r should have happened before this occurrence of q).





■ **Figure 3.** *MOTEL's GUI screen dump.*

Request Broker Architecture (CORBA). For the instrumentation of the service, interceptors are put in place in the middleware layer. These *interceptors* send notifications to MOTEL when relevant events occur in the system.

- Using the observations gathered at runtime in the system under scrutiny for conformance to the requirements (expressed properties). The properties are translated into FSMs (automata), where the reported events are used to check if an error state is reached.

Let us consider Fig. 1 for an example of the use of MOTEL. In Fig. 1, we present a basic view of a hybrid closed user group (CUG) [1]: one user with an IP terminal, two users with traditional phones, and one user with a cellular phone. To simplify the example, we consider that each hardware equipment is represented as a single software component, named *service component*, which communicates by means of middleware. By introducing MOTEL in the service, we get a behavioral view of the system. Each event in the system is reported to MOTEL and used to check if the behavior of the service violates what is expressed in the properties. In the context of a CUG, MOTEL can be used to detect service interactions. For example, a basic privacy property of a CUG states that "It is always true that if the source of a message comes from a registered CUG user and is destined to the CUG, then any final destination of the message is also in the CUG." Consider now, as in Fig. 1, that one user subscribes to a supplementary service such as call forwarding, and forwards all incoming calls to another terminal that is not registered in the CUG. This would imply that the privacy property is violated, since private messages are sent to a user (terminal) not registered in the CUG. MOTEL is able to detect this kind of misbehavior by checking the privacy property during the execution (development or operational stage) of the service.

Let us look in more detail at the structure of MOTEL and the testing process. MOTEL is divided into two functional parts: a *monitoring manager* and a *testing manager* (Fig. 2).

The monitoring manager is responsible for all the activities required for observing the running implementation and for forwarding the observed information to the testing manager. The monitoring manager is divided into a *properties translator* and an *observation manager*.

The properties translator analyzes and transforms the properties into FSMs and gives them to the testing manager.

The observation manager is in charge of setting up the interceptors in order to get the needed information, and of handling the information received in order to ensure its consistency. This is performed only when necessary, to minimize the impact on the system.

The testing manager consists of a *properties checker* and an *error handler*. The properties checker handles the FSMs, translated by the properties translator. Each time the observation man-

ager reports an observed event, the properties checker updates each FSM in order to reflect the current state of the system under scrutiny. If an error state is reached, an *error report* is generated. The error report contains the last events causing the error, as well as the value of other configuration parameters that are useful to the user when diagnosing the error.

In order to improve the user-friendliness of MOTEL, a graphical user interface (GUI) is provided. This GUI is depicted in Fig. 3. Via menus, the user is guided to select the different events that can be part of a property and other menus are offered with the LTL operators in order to write the properties that have to be checked at run-time (window entitled "New Property"). The expressed properties appear in a list (window entitled "List of Properties") and can be selected for testing. The monitored information is also displayed in the GUI: in a table (window entitled "MOTEL") or in a timeline diagram (window entitled "Timeline"). The FSMs that are generated are shown in the window entitled "Automata." Property violations are reported to the user (window entitled "Property violation"). For a detailed description of MOTEL we refer the interested reader to [12, 13]. Jagadeesan *et al.* [14] developed a tool with similar concepts, but the test of properties is restricted to a single category (safety properties).

MOTEL is currently being integrated into a service design- and development platform (the PERCO Platform [15]) developed by Alcatel/Thomson. The development environment also includes a model checker, an automatic test case generator [10], and a behavior simulator. The simulation model is derived from the Unified Modeling Language (UML) description of the application. The simulation also uses a model of the executive support behavior (distributed platform, fault tolerance, remote creation, load balancing, threading).

The platform is used notably to support hybrid services. The platform supports complex applications, such as equipment supervision. Forming an essential part of the Alcatel platform, MOTEL is used to check whether a number of service properties are satisfied. The properties to validate are derived from corresponding UML specifications.

Another objective of MOTEL is to identify the differences between the behavior obtained by simulation and model checking, and the behavior on the PERCO runtime.

Conclusions

In this article we look at the validation of communication services in general and hybrid services in particular. Starting with a brief survey on the formal methods used in the industry for the development of communication services, we identify a number of peculiarities of hybrid services that are likely to trigger some changes in the focus of formal methods research. Most important, more pragmatism is needed in order to apply the recent results of research on formal methods for the development of hybrid services. We have briefly reported on a project and a tool that will be used to validate hybrid services. We believe that our method answers some of the most crucial industrial concerns about the use of formality.

Acknowledgments

The authors would like to thank the anonymous reviewers as well as H. Cogliati and S. Koppenhoefer for their comments on the article.

References

[1] C. Gbaguidi *et al.*, "A programmable architecture for the provision of hybrid services," *IEEE Commun. Mag.*, this issue.

[2] R. Glass, "Formal methods are a surrogate for a more serious software concern," *IEEE Comp.*, Apr. 1996, pp. 19–20.

[3] E. Clarke and J. Wing, "Formal methods: State of the art and future directions," *ACM Comp. Surveys*, vol. 28, no. 4, Dec. 1996, pp. 626–43.

[4] G. Holzmann, "The theory and practice of a formal method: NewCoRe," *Proc. IFIP World Comp. Cong.*, Hamburg, Germany, Aug. 1994, vol. I, Amsterdam: North Holland, pp. 35–44.

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

[6] C. Middelburg, "A simple language for expressing properties of telecommunication services and features," Tech. rep. 94 356, PTT Research, Apr. 1994.

[7] B. Steffen *et al.*, "A constraint oriented service creation environment," *PACT '96, 2nd Int'l. Conf. Practical App. of Constraint Tech.*, London, U.K., 1996.

[8] S. Aujla, T. Bryant, and L. Semmens, "Applying formal methods within structured development," *JSAC*, vol. 12, no. 2, Feb. 1994, pp. 258–64.

[9] P. Bosco *et al.*, "ACE: An environment for specifying, developing and generating TINA services," *Proc. 5th Int'l. Symp. Integrated Network Mgmt.*, San Diego, CA, May 1997.

[10] J. M. Jezequel, A. Le Guennec, and F. Pennaneac'h, "Validating distributed software modeled with UML," *Proc. UML '98 Int'l. Wksp.*, San Diego, CA, May 1997.

[11] P. A. Etique, J. P. Hubaux, and X. Logean, "Service specification and validation for the intelligent network," *Interoperable Commun. Network*, Balzer, vol. 1, no. 1, Jan. 1998, pp. 41–69.

[12] X. Logean, F. Dietrich, and J. P. Hubaux, "TINA service validation: The ErnesTINA project," *Proc. IEEE ICC*, Atlanta, GA, June 1998.

[13] F. Dietrich, X. Logean, and J. P. Hubaux, "Testing temporal logic properties in distributed systems," *Proc. IFIP Int'l. Wksp. Testing of Commun. Sys.*, Tomsk, Russia, Aug. 1998.

[14] L.J. Jagadeesan, A Porter, C Puchol, J.C. Ramming, and L.G. Votta, "Specification based testing of reactive software: Tools and experiments," *Proc. 19th Int'l. Conf. Sofware Eng.*, May 1997.

[15] Alcatel/Thomson Common Labs, "The Perco platform," *Proc. ISORC '99*, St. Malo, France, May 1999.

Biographies

XAVIER LOGEAN (xavier.logean@epfl.ch) graduated in 1995 from the Swiss Federal Institute of Technology (EPFL) with a degree in electrical engineering. He is currently a Ph.D. student in the Institute for Computer Communications and Applications (ICA) at EPFL. His main research interests are in testing and monitoring of distributed systems.

FALK DIETRICH (falk.dietrich@epfl.ch) studied computer science at the Technische Universitat Dresden, Germany, and at Rensselaer Polytechnic Institute, Troy, New York. He is currently working toward a Ph.D. at the Institute for Computer Communications and Applications (ICA) at EPFL in Switzerland.

JEAN-PIERRE HUBAUX [SM] (jean-pierre.hubaux@epfl.ch) joined the Swiss Federal Institute of Technology — Lausanne (EPFL) as an associate professor in 1990; he was promoted to full professor in 1996. He is co-founder and co-director of the Institute for Computer Communications and Applications (ICA, icawww.epfl.ch). His research activities are focused on service engineering, with a special emphasis on multimedia services; recently this focus was extended to security and mobile applications. He has authored and co-authored more than 40 publications in this area and holds several related patents. At the beginning of his activities at EPFL, he defined the innovative curriculum in communication systems (sscwww.epfl.ch). He will chair the Communication Systems Division as of October 1999. Prior to this position he spent 10 years in France with Alcatel, where he was involved in R&D activities, primarily in the area of switching systems architecture and software.

SYLVAIN GRISOUARD (grisouar@aar.alcatel-alsthom.fr) is responsible for the System Architecture Group in the Object Architecture Unit (UAO) of Alcatel-CIT Corporate Research Center. He graduated from university in applied mathematics. In 1987 he joined ONERA, where he worked in the Parallel Computing Research Group. He joined Alcatel in 1991, first in Cegelec, and then in Alcatel Business Systems/Mobile Phones where he worked on GSM software architecture design, development tools for real-time embedded applications, microkernels, and performance evaluation. He joined the Alcatel Research Division in 1996 and participated in the STREAM and SOOM ESPRIT projects. His activities focus now on the definition of PERCO, a component development environment coupled to a distributed and mission-critical CORBA platform. These activities are directly linked to the participation of Alcatel in the definition of CORBA real-time, CORBA fault tolerance, and minimal CORBA.

PIERRE-ALAIN ETIQUE (pierre-alain.etique@swisscom.com) holds a computer science degree from the Swiss Federal Institute of Technology in Zurich (ETHZ) and a Ph.D. in Communication Systems from the Swiss Federal Institute of Technology in Lausanne (EPFL). He worked with Ascom, a Swiss telecommunications company on the development of a PBX, and he is currently working for Swisscom, the biggest Swiss telecommunications operator, where he is leading an exploration program on "mobility, cards and security."