From Complex Specifications to a Working Prototype. A Protocol Engineering Case Study[$]

Manuel J. Fernández-Iglesias[@], Francisco J. González-Castaño[@], José M. Pousada-Carballo, Martín Llamas-Nistal, Alberto Romero-Feijoo.

Grupo de Ingeniería de Sistemas Telemáticos.
Departamento de Tecnologías de las Comunicaciones.
Universidad de Vigo. Campus Universitario s/n. E-36200 Vigo, Spain.
E-mail manolo@ait.uvigo.es.

Intended category: 1 Use of formal methods, industrial case study.

Abstract

We describe our experience using Formal Description Techniques (FDTs) to support the design of interception systems for GSM networks. Both the GSM protocol and the interceptor have been specified using LOTOS, an FDT standardized by the International Standardization Organization (ISO) to describe communication systems and protocols. This has permitted us to asses the feasibility of the proposed system and speed up further design phases. From the LOTOS model, a simulator has been generated automatically. The TOPO tool set was used across the process. An FTP link to a package containing the specification and the simulator is provided.

1 Introduction

This paper describes our experience using formal methods to support the design and implementation of a new product that should be integrated into a standard-based communication system, namely GSM[19,28] networks. In other words, the target product should be flawlessly integrated into an operating GSM network, it should interact with the network according to the corresponding standards, and it should offer the intended service.

Basically, this service consisted on the detection and identification of active mobile terminals in a designated, closed area, to intercept incoming or outgoing calls when desired. Selective blocking is related to applications where call establishment may be dangerous or simply a nuisance. Examples of designated areas where this system is intended to operate are social areas as concert halls or temples, health care areas[7], or restricted security areas where speech and data transfers are not permitted.

[$] This research work has been supported by Spanish project grants TIC 1FD97-1485-C02-02 and TIC 97-0669-C03-03. Implementation aspects are subject to Spanish patents P9801903 and P9901114.

[@] Manuel J. Fernández-Iglesias was affiliated to the International Computer Science Institute, Berkeley, USA, and Javier González-Castaño was affiliated to the Computer Sciences Department, University of Wisconsin-Madison, USA, when this version was written.

To be able to offer this service, GSM protocols should be analyzed to extract the information related to our problem. Moreover, the results should be verified to guarantee that the new service is consistent with the normal operation of existing GSM networks. Therefore, success depended dramatically on an initial, comprehensive analysis and understanding of GSM standards, which are scattered through more than 5,000 pages written in natural language[1].

This scenery is well suited for a formal design approach based on the concept of successive refinements[16]. On one side, this well-established design strategy will prevent designers from going on the next phase until they had a complete mastering of the system at that level of abstraction. As a consequence, design flaws will be detected at early stages, speeding up the overall design process. On the other side, the formal approach will force a methodical, comprehensive analysis of the target problem, which, as the first benefit, will permit a better understanding of the set of GSM recommendations. Furthermore, our group has some background both in Formal Methods[5,15,16] and jamming systems for mobile communications[25]. This approach fosters further collaboration between these research lines.

2 Service Description

In a few words, the proposed system can be described as a selective and range-limited device to prevent the operation of mobile GSM terminals inside a designated area. A GSM call evolves through a number of protocol messages exchanged between the mobile station (MS) and the corresponding base station (BS). Basically, our system should monitor and capture messages generated by a base station containing mobile station identifiers. These identifiers will then be used to decide if a given MS should be permitted to proceed with call initiation, or should be eventually blocked. This process takes place inside a closed, designated area.

The GSM protocol describes four types of transactions that trigger MS activity: MS location at power-up, BS-initiated MS location update, MS-terminated call, and MS-originated call. For the service we intend to offer, a relevant transaction is initiated when a message carrying ID information is detected, and captured to extract the corresponding MS identifier. Then, it should be decided, either locally or remotely, if the target MS should be blocked.

The GSM standard specifies several control procedures both on the network and subscribers. Nevertheless, none of them is intended to operate in designated closed areas. Therefore, external equipment should be provided to handle the situations described above, that is, to detect specific mobile terminals, and to gather sufficient information from the network to disable them, or to disable specific features when needed.

[1] European Telecommunication Standards Institute (ETSI) standards GSM01.10, GSM03.22, GSM04.01, GSM04.03, GSM04.04, GSM04.05, GSM04.06, GSM04.07, GSM04.08, GSM05.02, GSM05.03, GSM05.10 and GSM10.20 were relevant for this work.

Note that, as described here, MS blocking is subject to a hard real-time constraint: it should take place before user traffic flow starts or, for some scenarios, even before the call is signaled to the user. Available time to proceed is further restricted because part of the MS-BS dialog is ciphered. A comprehensive quantitative analysis was performed to estimate system feasibility[6].

3 Towards a Formal Model

LOTOS[13] has been selected as the supporting formal language. LOTOS is based on CCS[21], CSP[11] and CIRCAL[20] (behavior specification) and ACT_ONE[2] (data type specification), and supports the representation of temporal ordering of events, concurrent processes, non-determinism and process synchronization. This formal description technique has been standardized by the International Standardization Organization (ISO) for the specification of open communication protocols and systems. Note that the GSM specification is compliant to the ISO Open Systems Interconnection standard and, as stated above, our group had previous experience on the use of LOTOS, both at theoretical and practical level. The TOPO[17,18] tool set offered convenient support for syntactic and semantic analysis, symbolic execution, validation and automatic prototyping through the translation of annotated LOTOS code into C.

The final objective of the formal design process is to assess the feasibility of our target system. For this, we should evaluate the possibility of interacting with the overall GSM protocol to intercept signaling information. In case the protocol indeed permits such interaction or tampering, we should identify how and when it should be done. Then, we should develop a prototype to field test the proposed solution.

The first task was to identify and model formally the relevant aspects of the GSM transactions that lead to MS activity, as discussed in section 2. An analysis of the GSM protocol [3,4] reveals that they converge after a specific message is sent by the mobile station[2]. This particular message should trigger the interceptor, since a MS identifier is already available. Transaction will diverge again at some point after the dialog is ciphered, out of the scope of the interceptor.

Thus, we generated a model for the GSM system that only considered those aspects related to our goals, namely MS detection and identification, and call initiation and termination. The model would be compliant with GSM standards insofar the modeled functionality is concerned, and would permit the feasibility analysis described above. Furthermore, it would serve as the basis for the corresponding implementation.

In the next section we discuss the relevant aspects of this process. A package containing the full LOTOS specification, test process and results, a simulator, and a tool to generate GSM traces can be downloaded from any of the two locations below:

ftp://helia.ait.uvigo.es/pub/interceptor/interceptor.tar.gz
http://alen.ait.uvigo.es/interceptor/interceptor.tar.gz

[2] A *PagingResponse* or *CMServiceRequest* network message coded into a *Set Asynchronous Balanced Mode* (SABM) link frame.

4 The Formal Design Process

The next paragraphs will be devoted to present the overall architectural organization, to briefly outline the final LOTOS specification for the intended system, and to comment some aspects related to testing and validation.

4.1 System Architecture

The formal architecture for our system consisted on two communicating process (see figure 1). This layout has been chosen having in mind that a working prototype would eventually be constructed from the formal specification, to assess the feasibility of our interceptor.

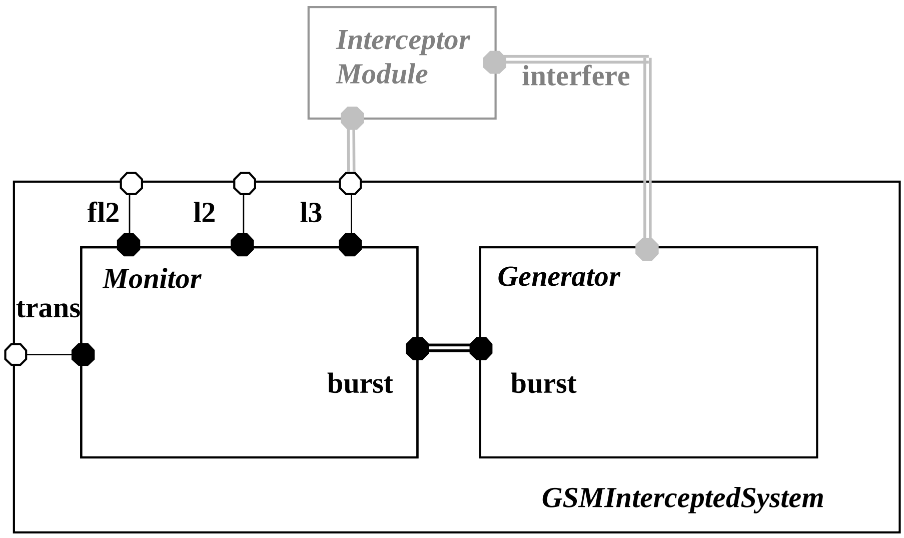

Figure 1. System Architecture

These top-level communicating processes are:

- A process that monitors the radio communication channel to detect MS activation or call initiation. This process (process *Monitor* in figure 1) is the formal specification of the required functionality, as outlined in section 2.
- A process that models physical level burst generation (i.e. the information that travels through the radio communication channel) for a base station and an undefined number of mobile stations, and therefore generates dialog sequences according to the GSM standard. This process (process *Generator* in figure 1) is in fact a LOTOS formal specification of the GSM protocol, insofar the relevant functionality is concerned.

These processes interact through gate **burst**, which models the radio channel. *Generator* produces physical-level protocol frames, and *Monitor* captures these frames for further analysis. The system interacts with the environment through gates **trans**, **fl2**, **l2** and **l3**. Through these gates, an external observer can analyze the GSM protocol evolution at the desired level of abstraction, as discussed in table 1, for individual transactions. For example, gate **trans** signals state changes for transactions between a BS and a MS providing the environment with transaction identifiers and the corresponding present state for that transaction. The external observer may be another LOTOS process synchronizing with this specification, for example, a testing process that model specific features of the GSM standard. In the case of a symbolic execution tool, the role of the external observer will be played by the system engineer interacting with the LOTOS specification through the user interface provided by the tool.

Gate	Level of abstraction
trans	Plain TDMA
fl2	Physical
l2	Data link
l3	Network

Table 1. Level of abstraction for the observation gates

An interceptor can be easily constructed from *Monitor*. For this, an *InterceptorModule* (see figure 1) synchronizes at the network level to detect a *PagingResponse* frame (see footnote in section 3) and then disables normal evolution of *Generator* through gate **interfere**. Process *Monitor* is presented in figure 2. *SynchroTDMAFrame* models slot synchronization for a Time Division Multiplex Access (TDMA) burst and acquires information about the distribution of GSM control channels. After synchronization control is passed to the *Filter* block. *L1Filter* and *L2Filter* capture frames at the corresponding level to construct network-level frames for *Transactions*. This latter process extract network-level information and assigns transaction IDs to monitored transactions.

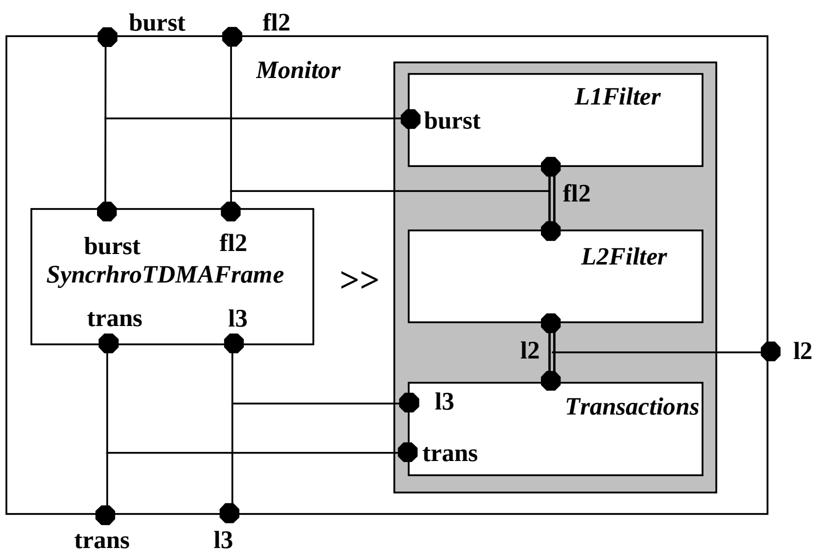

Figure 2. LOTOS specification. *Monitor* process

4.2 LOTOS Specification. A Brief Outline

The LOTOS specification includes the behavior description for the target system, abstract data type (ADT) definitions for all relevant data structures, and test processes to validate the system. The text of the specification comprises around 1,300 lines of annotated LOTOS code representing more than 50,000 different states and transitions. Apart from the LOTOS standard library providing basic data types as booleans or natural numbers, ADTs describe the syntax and (equational) semantics of data link and network messages, transaction states, logical channel references, and information fields in bursts, frames and messages. The ADT specification is organized as 16 LOTOS type definitions. These abstract definitions are translated into real C data structures by the TOPO tool set.

Behavior is specified by 16 communicating concurrent processes. Figure 3 outlines the interactions among these processes. *GSMInterceptedSystem*, *Monitor*, *Generator*, *InterceptorModule*, *SynchroTDMAFrame*, *L1Filter*, *L2Filter* and *Transactions* have been discussed above. Process *AnalyzeTDMAFrame* monitors the Broadcast Control Channel (BCCH) to acquire information about the distribution of control channels. *Transaction* monitors a specific Standalone Dedicated Control Channel (SDCCH), and checks transaction evolution to compare it with the relevant cases for the intended system. Process *Sabm* monitors the data link protocol in balanced asynchronous mode. This will be the active mode until the end of the active transaction. Other frames in the channel are ignored. Processes *NextInfo0* through *NextInfo4* implement a frame recognition machine, that is, a state machine to sort out different link and network level messages. In figure 4 we outline this state machine. Invalid frames are signaled with **tabort**, and valid protocol frames are signaled with **tsend**.

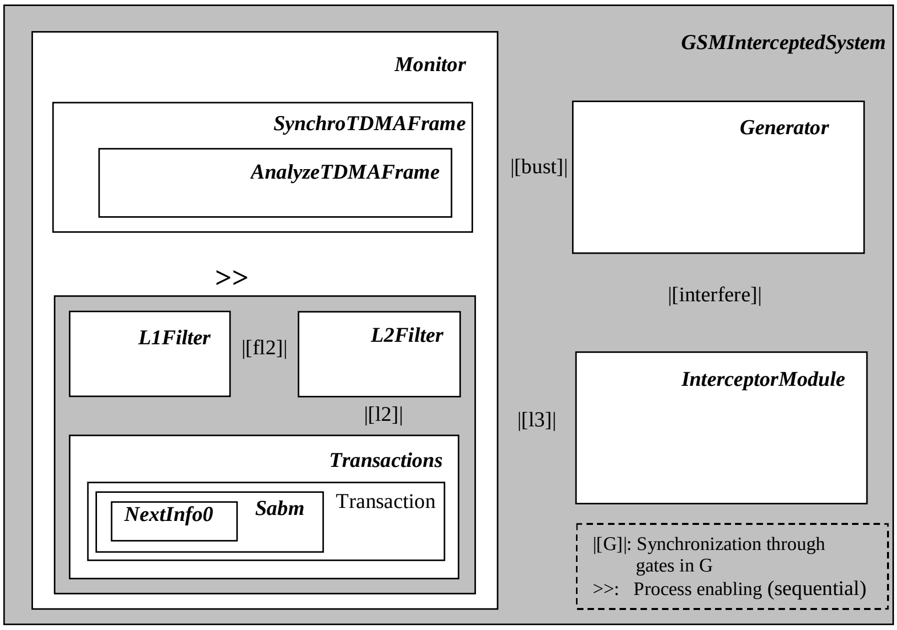

Figure 3. LOTOS Specification. Process Interaction

The rest of the processes in the LOTOS specification correspond to test sequences. Some of these test sequences are presented in table 2. For further details on testing this system refer to the discussion below and the package referenced in section 3.

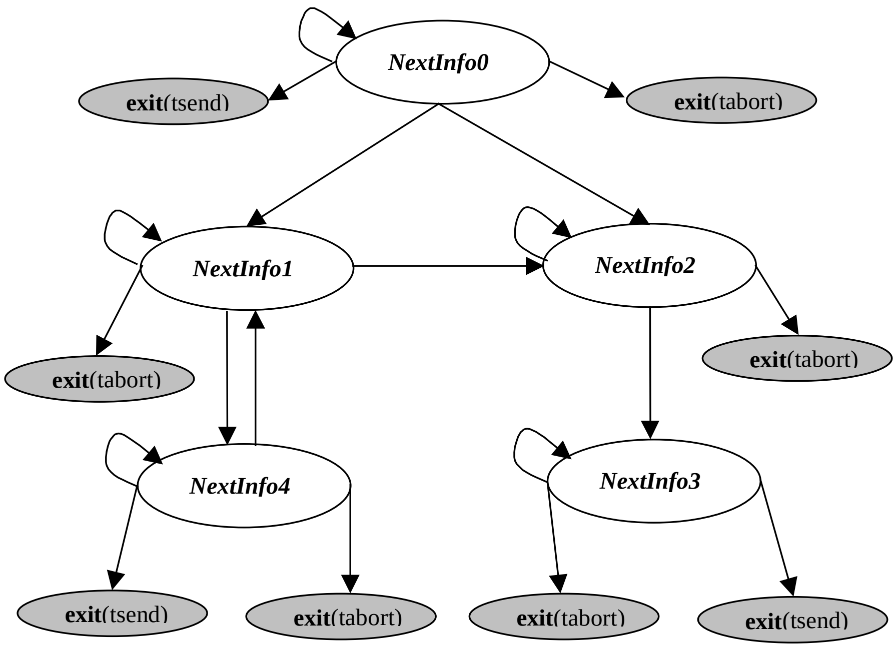

Figure 6. LOTOS Specification. Frame analysis

4.3 Protocol testing and validation

The final LOTOS specification was obtained through several top-down refinement steps, from an initial specification reflecting only overall architecture design decisions. It included all relevant protocol details needed to assess the feasibility of the target system. The TOPO tool set provided the necessary support along this process. It provided tools for syntax and static semantics checking, dynamic semantics checking and validation for each refinement step. That is, along the successive refinement process, thanks to TOPO was possible to check if each step provided a meaningful LOTOS specification, and if it was correct and consistent with the results from the previous refinement step.

For validation, two approaches were initially available. On one hand, test suites describing potential system behaviors where generated to cover all relevant aspects of the intended functionality, as pointed out by the GSM recommendations. An adequate tool (e.g. LOLA[27] from the TOPO tool set) will compose the test process with the target specification to analyze if the test passes or not. Technically, a test process reduces possible system behaviors to those specified by the test. If the resulting behavior does not include all states in the test process, the corresponding test will not pass. Table 2 presents some sample test sequences.

Process	S	T	D	R	Description
Test0000	2	2	0	MUST	Inactivity timeout if no activity at radio channel
Test0001	6	8	1	MAY	Inactivity timeout if unable to synchronize
Test0002	19	29	1	MAY	Timeout after several unsuccessful synchronization attempts

Test0100	13	17	1	MAY	Configuration captured and successful message reception
Test0101	15	21	1	MAY	Configuration captured and rejection of an incorrect frame
Test0102	15	14	5	REJ	Configuration captured and rejection of a message from an invalid channel
Test0200	124	360	2	MAY	- Simple transaction, no errors, simple authentication protocol and switching to ciphered mode. - Aborted transaction due to inactivity - Two concurrent simple transactions, one free of errors, another receives a DISConnect - Transaction with protocol errors

Legend: S= # of analyzed states; T= # of generated transitions; D= # of deadlocks found; R= Result of the test: MUST= Must pass. There is one single trace leading to the successful event in the test; MAY= May pass. Some traces lead to the successful event, other traces are deadlocks or lead to different events; REJ= Reject. No single trace lead to the defined successful event. In all cases, the results were as expected. Successful event for test *Test0102* implied acceptation of the message coming from an invalid channel.

Table 2. Sample test sequences

On the other hand, available tools permitted to check if a refinement step was equivalent to the initial specification under bisimulation[22] and testing equivalence[24]. However, this approach was feasible only at early design stages. State explosion made this solution unfeasible as the results for the successive refinement steps became more and more complex.

To sum up, the successive refinement strategy together with the formal support provided by LOTOS, prevented the designers from going on to the next phase until they had a complete understanding of the system at that level of abstraction. As a consequence, design flaws were detected at early stages, speeding up the overall design process.

4.4 Simulation and rapid prototyping

Once the successive refinement process terminates with a model of the target system that includes all the desired functionality, we must test how the system under development behaves in a real implementation. Along the formal design process we obtained an abstract model that is functionally equivalent to the final product. Anyway, some aspects will become apparent only during the implementation phase. These implementation-dependent aspects are related, for instance, to resource usage or performance.

A Unix executable version of *GSMInterceptedSystem* was generated from the LOTOS final specification using TOPO. The LOTOS code was previously annotated [18], basically to fill the gap between the abstract data types of LOTOS and the data types in C. Annotations define how to translate abstract data types into C data structures, and how to implement other minor architecture-dependent aspects (e.g. file access, output formatting, etc.). Additional code was introduced to play the role of the external environment (i.e. mobile phone subscribers). This code implements a user interface to provide the generator with call establishment events. Sequences of events can be defined and stored in trace files to be easily reproducible during prototype validation. The annotated LOTOS code was translated into around 7,600 lines of C code (around 6 automatically generated C code lines for each LOTOS line).

We found that this was a convenient approach to rapid prototyping. In it, a system prototype is generated (almost) automatically from the final specification. This prototype will permit a preliminary field test of the desired product that will help to introduce many of the implementation-dependent issues not considered during the formal design process. In our particular case, this approach had further advantages:

First, it permitted simple protocol reconfiguration. The protocol has been formally specified in LOTOS and a prototype has been automatically generated using TOPO. Any modification to the protocol can be introduced and formally tested in the LOTOS specification. Then, it can be recompiled to get a new working version of the prototype reflecting the changes.

Besides, the prototype can be tuned up for customized testing. Selected test processes can be included in the system prior to prototype generation. The resulting prototype will have only that functionality specified by the test processes. We can then concentrate in specific features of the system along the prototype evaluation phase.

Additionally, this approach provided full coverage simulation. As the formal model includes both the interceptor and the GSM protocol (process *Generator*), there is no need of an external system to generate events for the interceptor. The prototype will read event trace files for the *Generator*.

5 Concluding Remarks

We have presented in this paper our experience using formal methods for the design of interception equipment for GSM mobile terminals. LOTOS was selected as the supporting formal language because a broad selection of tools were available to support the design process, and our group had previous experience using this language. Anyway, any formal language with comparable expressive power would be suitable for this task. Among others, ESTELLE[12], SDL[14], or PROMELA[10], to name some languages the authors have some experience with, would also be good choices.

Although the appropriateness of Formal Methods for the development of industrial computer applications has been and is controversial[1,8,26,29], the results obtained using LOTOS were beyond our expectations. The generated code is compact, expressive and permits straightforward verification. Note that, for prototyping, not a single line of code concerning the core of the system (i.e. the corresponding state machine) had to be written by hand. This is particularly interesting because the core of the system is the most complex part and, eventually, the less portable. The only code additions were related to interfacing (e.g. file access and user dialog management) and to data structure translation.

Besides, the FDT approach was of paramount importance to focus the problem, that is, to separate major from minor aspects and to offer the designers a solid base to support their work.

References

[1] J. Bowen, M. Hinchey, Seven more myths of formal methods, IEEE Software, 12(4) (1995) 34-41.
[2] H. Ehrig, B. Mahr, Fundamentals of algebraic specification 1, EATCS Monographs on Computer Science, Springer Verlag, 1985.
[3] ETSI, GSM 03.22: Functions related to mobile station (MS) in idle mode and group receive mode, Digital cellular telecommunications system (Phase 2+) standards suite, ETSI Technical Report ETR 300 930, 1997.
[4] ETSI, GSM 04.07: Mobile radio interface signalling layer 3. General aspects, Digital cellular telecommunications system (Phase 2+) standards suite, ETSI Technical Report ETR 300 939, 1997.
[5] M. J. Fernández-Iglesias, M. Llamas-Nistal, Algebraic Specification through Expression Transformation, in: T. Mizuno, N. Shiratori, T. Higashino, A. Togashi, (Eds.), Procs. of the Joint Conference FORTE/PSTV'97, Chapman & Hall, 1997, pp. 355-366.
[6] F. J. González-Castaño, A. Romero-Feijoo et al, Real-Time Interceptor for the GSM Protocol, Technical Report UV/DTC/GRPILF00, Departamento de Tecnologías de las Comunicaciones. Universidad de Vigo, 2000.
[7] H. Grant, Managing electromagnetic compatibility between wireless and medical devices, Compliance Engineering, may-june (1999) 26-31.
[8] J. A. Hall, Seven myths of formal methods, IEEE Software, 7 (5) (1990) 11-19.
[9] J. Hennessey, D. Patterson, Computer Architecture: A quantitative Approach, Morgan Kaufmann Pub, 1996.
[10] G. Holzmann, Design and Validation of Computer Protocols, Prentice Hall, 1991.
[11] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.
[12] ISO, ESTELLE: A formal description technique based on the extended state transition model, International Standard 9074, International Standardization Organization, 1989.
[13] ISO, LOTOS: A formal description technique based on the temporal ordering of observational behavior, International Standard 8807, International Standardization Organization, 1988.
[14] ITU, SDL: Specification and Description Language, CCITT Recommendation Z.100, International Telecommunication Union, 1993.
[15] M. Llamas-Nistal, M. J. Fernández-Iglesias, J. C. Burguillo-Rial, J. M. Pousada-Carballo, F. J. González-Castaño, L. Anido-Rifón, LOTOOL: A Tool for Developing, Verifying and Testing Protocols using Formal Description Techniques, in: Innovations and Quality in Education for Electrical and Information Engineering, European Association for Education in Electronic and Information Engineering, 1997, pp. F1.18-F1.23.
[16] M. Llamas-Nistal, J. Quemada, M. J. Fernández-Iglesias, Direct Verification of Bisimulations, in: Procs. of the Joint Conference FORTE/PSTV'96, Chapman & Hall, 1996.
[17] J. A. Mañas, T. De Miguel, From LOTOS to C, in: K. J. Turner, (Ed), Procs. of FORTE'88, North Holland, 1988, pp. 79-84.
[18] J. A. Mañas, T. de Miguel, J. Salvachúa, A. Azcorra, Tool support to implement LOTOS specifications, Computer Networks and ISDN Systems, 25 (1993) 79-84.
[19] A. Mehrotra, GSM System Engineering, Artech House, 1997.

[20] G. Milne, CIRCAL and the representation of communication, concurrency and time, ACM Trans. on Programming Languages and Systems, 7(2) (1985) 270-298.
[21] R. Milner, Calculus of communicating systems, Lecture Notes on Computer Science 92, Springer Verlag, 1980.
[22] R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[23] M. Mouly, M.-B. Pautet, The GSM System for Mobile Communications, Cell & Sys, 1992.
[24] R. de Nicola, Extensional Equivalences for Transition System, Acta Informatica, 24 (1987) 211-237.
[25] J. M. Pousada-Carballo, F. J. González-Castaño, F. Isasi de Vicente, M. J. Fernández-Iglesias, Jamming System for Mobile Communications, IEE Electronic Letters, 34(22) (1998) 2166-2167.
[26] S. L. Pfleeger, L.Hatton, Investigating the influence of formal methods, IEEE Computer, 30(2) (1997) 33-43.
[27] J. Quemada, S. Pavón, A. Fernández, Transforming LOTOS specification with LOLA, in: K. J. Turner, (Ed), Procs. of FORTE'88, North Holland, 1988.
[28] S. M. Redl, M. K. Weber, M. W. Oliphant, An introduction to GSM, Artech House, 1995.
[29] J. Voas, Software quality's eight greatest myths, IEEE Software, 16(5) (1999) 118-120.