FORMAL METHODS FOR MOBILITY STANDARDS

Daniel Amyot, Rossana Andrade, Luigi Logrippo, Jacques Sincennes, Zhimey Yi

TSERG, School of Information Technology and Engineering, University of Ottawa
150 Louis-Pasteur, Ottawa, Ontario, Canada K1N 6N5
e-mail: {damyot | randrade | luigi | jack | zyi}@site.uottawa.ca

Abstract - Precise specification and exacting verification and validation of protocol standards are essential for their successful development and implementation. Currently, several languages (called FDTs for Formal Description Techniques) are available to address this issue. FDTs have reached various degrees of acceptance, but their use in standard development in North America has been limited. This paper represents an attempt towards divulging the knowledge of what exists and how it can be used effectively based on the methods that were found most useful in our work on mobility protocols. The FDTs considered are, in alphabetical order: the *Abstract Syntax Notation 1* (ASN.1), the *Language of Temporal Ordering Specifications* (LOTOS), *Message Sequence Charts* (MSCs), the *Specification and Description Language* (SDL) and the *Tree and Tabular Combined Notation* (TTCN). Also considered is a new, emerging non-formal technique, called *Use Case Maps* (UCMs).

Keywords**:** Mobility Standards, Formal Methods, Validation and Verification, Wireless Intelligent Network

I. STANDARDIZATION CHALLENGES

Numerous industries and standardization bodies (ITU, ISO, ANSI, ETSI, TIA, etc.) are constantly at work to design new telecommunications products and new standards for such products, involving increasingly complex functionalities. These services also require increasingly complex architectures and protocols, especially in a context of mobile communication. In the early stages of these processes, many features, services, and functionalities are described using informal operational descriptions, tables and visual notations such as *Message Sequence Charts* (MSCs) (16). As these descriptions evolve, they quickly become error-prone and difficult to manage. The need of precisely documenting all stages of the design process, which is very important in the industrial environment, becomes critical in the standardization process, where there is an international scrutiny process for which the stages are formalized and must undergo formal review and approval.

Inadequate descriptions are likely to hide ambiguities, inconsistencies or undesirable interactions inside or between services, or between levels of abstraction of a given service. These remain difficult to detect with conventional inspection methods, and often remain hidden until errors are discovered after implementation, at which point corrections can be very costly. A related issue is the one of interoperability of implementations. Unless standard documentation is very precise and stringently validated, chances that it may be interpreted differently by different implementors are high.

Following the practice in several standard groups, the development of each phase of a telecommunication standard is divided in three *stages*: 1) service descriptions, 2) functional entities and message sequence information, and 3) protocol and procedure specification.

Development of Formal Techniques

Several of the techniques discussed in this document have become known as Formal Description Techniques (FDTs) because they were formally defined, and in turn they allow to formally define distributed systems. This is the case for ASN.1, LOTOS, SDL, and TTCN.

These techniques have different histories. ASN.1, LOTOS, and TTCN were developed in relation to the standardization effort, within CCITT and ISO, of the Open Systems Interconnection (OSI) protocols and services. At the beginning, the OSI community resolved that formal methods were needed: they would be developed quickly and used in the standardization work. History unfolded differently. First, the development of the FDTs turned out to be a complex project that lasted longer than expected. The languages became available only in the late eighties, when much of OSI standardization had been completed. Second, several languages were developed while only one had been planned. Third, when people started looking at the languages, they found them falling short of their initial optimistic expectations. Notably, the languages and the associated tools and methodologies were incomplete, experimental, hard to learn and insufficiently documented. Implicitly, standards developers and industry questioned the wisdom of investing in something unproven (8).

Nevertheless, these languages have shown notable resiliency. They are still in use nowadays and are the subject of research and development. New tools have been developed for them and applied in areas at times quite remote from the ones envisaged by their initial designers.

FDTs still have their legitimate place in the standardization process, especially to provide precise descriptions that improve the likelihood of interoperability in a heterogeneous telecommunications environment.

In this context, the use of appropriate specification techniques for describing standards would have a profound positive impact on the overall consistency and validity of these documents, and on the effort required to draft and maintain them.

II. EVALUATION CRITERIA

Criteria were selected on the basis of our own experiences with specification techniques and requirements engineering, and on existing surveys (3)(9)(20). Thirteen criteria were grouped in four categories: usability, validation and verification, tool support, and training.

Usability

- *Readability*: descriptions need to be readable by domain experts (and not only by description technique experts).
- *Modularity*: composition operators are needed to allow large descriptions to be decomposed into smaller parts.
- *Abstraction*: is concerned with the level of detail that needs to be addressed, and with separation of concerns.
- *Scalability*: complex and simple systems should be describable in a similar way.
- *Maintenance and Evolution*: relates to reuse, modification of existing parts, and addition of new details.
- *Looseness*: when few details are available (early stages), description techniques should permit some level of incompleteness and non-determinism in a description.
- *Maturity*: a technique has a high level of maturity if it has reached a level of wide acceptance and application.

Validation and Verification (V&V)

- *Completeness and Consistency*: techniques should offer ways of checking completeness and consistency of partial functionalities, scenarios and levels of abstraction.
- *Testing and Simulation*: V&V is improved when descriptions can be executed, animated, simulated and tested.
- *Verifiability and Correctness*: verification of a model against requirement properties. Verification approaches are usually stronger than testing and simulation, but they are also harder and more costly to perform.

Tool Support

The techniques should be supported by tools for the capture, documentation, maintenance, animation, testing and verification of descriptions; with a special interest in multi-platform, industrial-strength and quality tools, where support and training are available.

Training

- *Learning Curve*: how quickly a new user can learn the concepts, theories, techniques, and tools to make useful application of the description technique.
- *Tutorials and Documentation*: good tutorials and documentation are necessary for a good training. Courses, case studies, and other technology transfer activities are important as well.

III. SELECTED FORMAL TECHNIQUES AND EVALUATION

In this section, a short overview of six specification techniques (UCM, LOTOS, MSC, ASN.1, SDL, and TTCN) particularly relevant to the documentation of telecommunication standards is given. Each of the six selected techniques is evaluated according to the criteria defined in Section II.

Other well-known specification techniques are not discussed in this document. Although many of them have reached good levels of recognition in different areas and have been standardized in some cases, it seems that at present they have limited potential for application in the North American telecommunications environment.

Use Case Maps (UCMs)

UCM is a visual notation developed at Carleton University (Ottawa) and utilized for capturing the requirements of reactive systems (6)(7). Unlike all other languages discussed in this report, this notation is not an FDT, although research has been done to map it into several formalisms (2). UCMs describe scenarios in terms of *causal relationships* between *responsibilities*. They can have internal activities as well as external ones. Usually, UCMs are abstract and include multiple traces. Scenarios are expressed above the level of messages exchanged between components; hence they are not necessarily bound to a specific structure. A causal relationship can be refined in many ways in terms of exchanges of messages, depending on the components structure, the available communication

channels and on the chosen protocols. UCMs suggest a concept of *architecting behaviour* instead of specifying behaviour as in most FDTs. UCMs have a precise graphical grammar, but the semantics is not formally defined.

Usability: UCMs are based on a graphical and intuitive notation which is highly *readable*. One characteristic of UCMs is to allow to specify systems at different levels of *abstraction*. For example, UCMs allow the designer to work with the amount of detail available, hence *looseness* is high. In many real-life examples that have been specified by UCMs, *scalability* appears to be excellent, especially when using the stub/plug-in mechanism for recursive definitions. Given its relative novelty and continuous evolution, this technique is still *not mature*.

V&V: Because of the notation's informality and looseness, *completeness and consistency* checking, as well as *verifiability*, become difficult issues and are hard to support. V&V of UCMs is possible by the intermediary of other techniques. For example, UCMs translate into LOTOS, so V&V can be done by using LOTOS tools (1)(2).

Tool support: Only one prototype editing tool is available for UCMs. Although further developments are planned, current support is still weak.

Training: Because of UCM's intuitive nature, the *learning curve* is excellent and the technique is easily accepted by many practitioners. It is possible to use the notation at different levels of competence. A book (6) and some tutorials are available (7).

Language Of Temporal Ordering Specifications (LOTOS)

LOTOS is an algebraic specification language and a FDT standardized by ISO for the formal specification of open distributed systems (10). It has formally defined syntax, static semantics, and dynamic semantics. Using LOTOS, the specifier describes a system by defining the temporal relations along the interactions that constitute the system's externally observable behaviour.

A LOTOS specification consists of behavior expressions and abstract data types. LOTOS behavior expressions consist of actions and processes combined by means of a number of operators. LOTOS is capable of describing and prototyping communicating systems at many levels of abstraction through the use of *processes, hiding, parallel composition*, multiway and nondeterministic *synchronization*. LOTOS is executable, and LOTOS models allow the use of a number of validation and verification techniques such as step-by-step execution (simulation), random walks, testing, expansion, model checking, and goal-oriented execution (4). Several tools can be utilized for the automation of these techniques, and several development cycles based on scenarios and stepwise refinement are available (1)(4).

Usability: The specification style in LOTOS can adapt itself to different expressive needs and LOTOS can be used at many different *abstraction* levels.

LOTOS specifications have been written for very complex systems. There is a record of over ten years of using this standardized language in many different application areas, and a forthcoming Enhanced LOTOS is currently under study (18). Thus, the language is *mature*.

LOTOS requires precision and each action sequences should be specified exactly. Although several nondeterministic alternatives can be specified, *looseness* is low.

V&V: LOTOS was designed for V&V and, by using it, many types of design errors can be detected. Many *inconsistency and incompleteness* problems have to be resolved at the time the LOTOS specification is written.

Testing and simulation are well supported within LOTOS methodology, since LOTOS is an executable language. Many techniques exist for *verification and correctness* checking in LOTOS. One of the most successful is model checking. There is plenty of experience in using them in specifications of real-life systems.

An additional asset of LOTOS is the *algebraic transformations*, leading to better implementations. Obtaining test cases from specifications leads to more accurate testing and LOTOS' full formal semantics enables a theory of verification.

Tool support: A number of software tools have been developed for supporting the V&V of LOTOS specifications, for example, LOLA (4), TOPO (4), ELUDO (developed at the University of Ottawa) and CADP, the Caesar/Aldebaran Development Package. These tools are routinely used in research environments, so they are fairly robust, although they are not industrially supported.

Training: LOTOS and its related methodology are very well documented in many books, tutorials, and papers (4). The language is not easy to *learn*, although its order of difficulty does not exceed the one of many 'unconventional' programming languages.

Message Sequence Charts (MSCs)

MSC, standardized by ITU-T (16), is a graphical and tex-

tual language for the description and specification of the interaction scenarios between system components. Its main area of application is as an overview specification of the communication behaviour of real-time systems, in particular telecommunication switching systems. MSCs may be used for requirement specification, simulation and validation, test-case specification and documentation.

MSCs focus on the communication behaviour between system components and their environment by means of message exchange. The main focus of MSCs is on the specification of special system properties or functions. For example, a set of MSCs usually covers only a partial system behaviour since each MSC represents one scenario.

MSCs complement SDL and can be used for the automatic generation of test cases. The MSC language has graphical (MSC/GR) and textual (MSC/PR) syntax forms. A recent enhancement, High-level MSCs, includes control structures that can combine several MSCs.

Usability: Being simple and intuitive in nature, MSCs are a very *readable* notation. Because of the fact that MSCs basically provide disjoint scenarios, *modularity, abstraction, scalability, maintenance and evolution*, are all fairly weak. This has been improved somewhat by the introduction of High-level MSCs. Although their formal definition is recent, MSCs have been in wide use in various forms for decades, hence they are a very *mature* notation.

V&V: MSCs present disjoint scenarios that are not executable, they are poor for *completeness and consistency* checking, *testing, simulation, verifiability* and *correctness*, yet widely used to represent the results of V&V activities.

Tool support: Various MSC-based tools have been implemented. Some of them are available commercially.

Training: MSCs are natural and easy to learn; the *learning curve* is excellent. Good *tutorials* and *documentation* are available.

Abstract Syntax Notation One (ASN.1)

ASN.1 is a language for describing structured information, typically, information intended to be conveyed across some interface or communication medium. In its essence, it is an extension of the well-known Backus Normal Form. ASN.1 is an ISO standard (13).

ASN.1 is widely used in the specification of communication protocols, cryptography, and electronic commerce, In particular, it is employed in virtually all of the emerging standards for the application layer of OSI (19).

Usability: Since ASN.1 is a notation for specifying precisely data structures and encodings, its place is at a very low level of *abstraction*, and *looseness* is very low. This is a very well established and *mature* technique, often used in combination with SDL(15), MSCs, and TTCN.

V&V: Because of the declarative nature of ASN.1, V&V activities do not particularly apply to it. ASN.1 tools include facilities for checking the syntax and consistency of the expressions.

Tool support: ASN.1 is well supported.

Training: Although not difficult, ASN.1 needs *learning.* There is at least one good book on ASN.1 (19). *Tutorial* materials are included in the supporting tools.

Specification and Description Language (SDL)

SDL (14) is another FDT designed for reactive, concurrent, real-time, distributed, and heterogeneous systems. The basic SDL model consists of extended finite state machines communicating by means of queues. Notions of types and inheritance make SDL an object-oriented language. In this context, SDL is suitable for international standards in the telecommunication area, for systems in development, and for verification and validation of the system behaviour. In short, SDL is a language to support human understanding of system descriptions, formal analysis and comparison of behaviors, in an implementation independent way.

SDL has two concrete syntaxes: the graphic representation called SDL/GR and the textual representation called SDL/PR. The graphic form is more intuitive and displays relationships more clearly than the textual form.

Usability: Concerning *modularity*, SDL includes now object-orientation, although some tools do not support it. Like all comparable techniques, SDL demands full precision and, thus, does not support *looseness*. SDL is very *mature*, since it has been in industrial use and regularly enhanced since the early 70s.

V&V: *Completeness and consistency checks, testing and simulation, verifiability and correctness* are all very well supported by the SDL tools. There is very considerable experience on using these V&V techniques with SDL.

Tool support: Well-known SDL toolsets include Telelogic's SDT and Verilog's (Object) Geode. Both offer integrated tools for editing, analyzing, and compiling SDL specifications and for report generation. Most tools also include facilities for editing, managing and generating

related descriptions in MSC, ASN.1, and TTCN. Extensive support for syntactical and semantic analysis, simulation and formal verification (usually based on state space exploration) are also provided for most of these tools. In addition, most of them generate code that will work in different target environments without modification. The code generation together with the high-level nature of SDL make development using SDL very cost efficient in all situations where the communicating, extended finite state machine paradigm fits the application.

Training: To learn and use SDL effectively, books, papers, technical reports, and a number of courses as well as commercial toolsets are available (5).

Tree and Tabular Combined Notation (TTCN)

The third part of ISO/IEC 9646 (11) has defined a test notation, called TTCN (12), for use in the specification of OSI abstract conformance test suites. In short, in constructing a standardized abstract test suite, a test notation is used to describe abstract test cases. Thus, the notation is independent of test methods. It is informal but with clearly defined semantics.

Two forms of the notation are provided: a human-readable tabular form, called TTCN.GR, for use in OSI conformance test suite standards, and a machine processable form, called TTCN.MP, for use in representing TTCN in a canonical form within computer systems or when transferring TTCN test cases between computer systems.

Usability: TTCN is a very clear and *readable* notation, mainly because of its graphical representation, and it also includes good *modularity* features for creating complex test suites from elementary test cases. This technique has been used for real systems, hence it is quite *scalable*. Since TTCN has been designed to describe precise test scenarios, it is *not loose*. TTCN is a very well-established and widely used technique, hence it is very *mature.*

Tool support: TTCN is supported by a number of excellent tools.

Training: TTCN is a natural technique to use for telecommunications specialists. Thus, the learning curve is very good and tutorial materials exist (17).

IV. CONCLUSIONS

Summary of the Evaluation

Table 1 summarizes the strengths and weaknesses of the specification techniques discussed in the previous section, according to the evaluation criteria of Section II.

Table 1: Evaluation of the Selected Techniques

Technique / Criteria	UCM	LOTOS	MSC	ASN.1	SDL	TTCN
Readability	+	0	+	0	0	+
Modularity	0	0	–	0	+	+
Abstraction	+	+	–	–	0	0
Scalability	+	+	–	0	0	+
Maint. & Evol.	0	0	–	0	0	0
Looseness	+	–	0	–	–	–
Maturity	–	+	+	+	+	+
Comp. & Consis.	–	+	–	0	+	0
Testing & Simul.	–	+	–	NA	+	+
Verif. & Correct.	–	+	–	NA	+	+
Tool Support	–	0	+	+	+	+
Learning Curve	+	–	+	0	0	+
Tutorials & Doc.	0	0	+	+	+	+

Legend: +:Strength; 0:Adequate; –:Weakness; NA: Not Applicable

Combined Use of Formal Methods

UCM is an intuitive notation. Its low learning curve, high level of abstraction and readability make it especially appropriate for stage 1 of the standardization process. UCMs offer an excellent medium for the initial exposures and discussions of features, services and functionalities. They can be used loosely and at several levels of detail. As such, they efficiently support the development and refinement process, while facilitating communication among teams.

MSCs are used to define stage 2 message sequences between system components. Since they can be generated from UCMs, and can guide the prototyping of LOTOS, SDL and TTCN specifications, they provide a transition between the different stages.

The purpose of ASN.1 is to accurately define data structures. Its use becomes beneficial as early as the need for detailed specification of data values, parameters and encodings emerges.

LOTOS' capability of specifying at high levels of abstraction brings it into play early in stage 2. The generation of LOTOS specifications can be guided by UCMs. LOTOS being also capable of specifying details, its use can carry on to the final stage. MSCs can be used to guide the generation of LOTOS specifications, as well as their validation and verification.

SDL requires precise identification of each state and knowledge of the underlying architecture. It is more implementation-oriented than LOTOS and hence comes into play at a later time, at the end of stage 2 and into stage 3. A SDL specification can be designed in conformance to UCMs, MSCs, ASN.1 and LOTOS specifications.

TTCN, being a test notation, comes into play when the design of the system is finalized. Tests cases for implementations can be derived from UCMs, MSCs, LOTOS and SDL specifications.

The top part of Table 2 presents the relevance of each evaluation criterion in the three stages of standard development. In general, most criteria are relevant for the three stages and this is especially true for scalability, maturity, tool support, learning curve, and documentation. Modularity, maintenance & evolution and V&V criteria are considered to be merely useful for stage 1, where only a general description of a service is required. Abstraction and looseness become less necessary for stage 2, and perhaps even less for stage 3 where detail and precision are required. Readability in stage 3 is less of an issue as readable but less detailed descriptions are available for stages 1 and 2.

Since none of the techniques discussed can cover all stages effectively, this table can help weighting their overall relevance in terms of stages. The bottom part of Table 2 presents the perceived relevance of each specification technique in each stage.

Table 2: Relevance of Criteria and Techniques

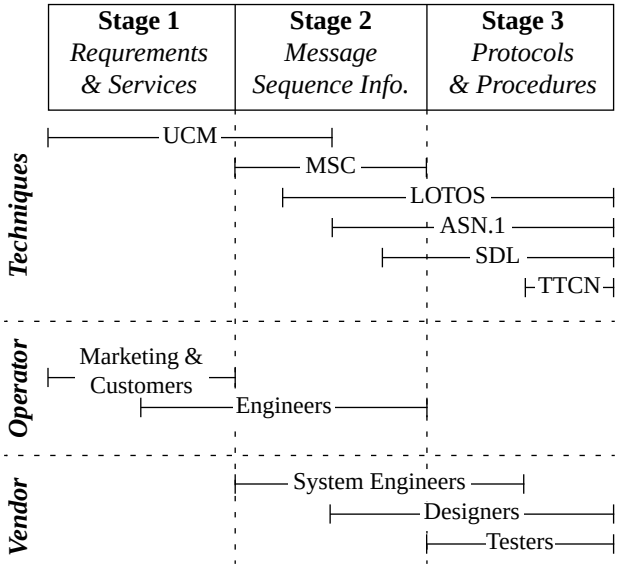
	Stages	1	2	3
Evaluation Criteria	**Readability**	+	+	0
	Modularity	0	+	+
	Abstraction	+	0	–
	Scalability	+	+	+
	Maintenance&Evolution	0	+	+
	Looseness	+	0	–
	Maturity	+	+	+
	Completeness&Consistency	0	+	+
	Testing&Simulation	0	+	+
	Verification&Correctness	0	+	+
	Tool Support	+	+	+
	Learning Curve	+	+	+
	Tutorials&Documentation	+	+	+
Techniques	**UCM**	X	X	
	LOTOS		X	X
	MSC		X	
	ASN.1		X	X
	SDL		X	X
	TTCN			X

Legend: +:Necessary; 0:Useful; –:Not Needed; X:Relevant

Suggestions on the Standard Lifecycle

The top part of Figure 1 details our conclusions on the use of specification techniques in the three stages of the standard development process.

Figure 1: Relevant Methods and Teams Involved for the Three Drafting Stages.



These techniques should be introduced in phases in the standard development process. All phases shall be defined and completed prior to presentation to potential users.

UCMs are not as mature as the other techniques, and adequate tool support will not be available until the end of this summer. Nonetheless, they are the most likely candidate for a first contact with new specification techniques. Their graphical and loose nature provides the best chance of acceptance in a large group of people with heterogeneous background.

MSCs are already well introduced to the target community. The next step might be to introduce SDL or LOTOS concepts at the appropriate stages and in the appropriate roles. For example, one could first experiment with the use of UCMs, LOTOS, and MSCs. A LOTOS specification can be generated and tested via UCMs, possibly refined as MSCs. Then, more illustrative and automatically validated message-oriented scenarios (MSCs) can be generated from the specification.

Although each technique has a role, not everyone would need to learn them all. As in most engineering areas, various specialized teams can come into play. Figure 1 presents such a scenario where different teams on the operator side and on the vendor side are involved in the three development stages.

The great complexity of distributed systems makes it advisable to analyze them with several complementary techniques. It is unfortunate that most teams specialize in a technique only. Much can be learned by competitive and complementary use of the various techniques, and this process deserves further study and experimentation.

Acknowledgment. The authors are grateful to Nortel, CITO and NSERC for their support of this research project. John Visser and Jim Hodges of Nortel provided many valuable insights in the nature and constraints of the standardization process.

REFERENCES

(1) Amyot, D., Hart, N., Logrippo, L., and Forhan, P., "Formal Specification and Validation using a Scenario-Based Approach: The GPRS Group-Call Example". In: *ObjecTime Workshop on Research in OO Real-Time Modeling*, Ottawa, January 1998. http://www.csi.uottawa.ca/~damyot/wrroom98/wrroom98.pdf

(2) Amyot, D., Buhr, R.J.A., Gray, T., and Logrippo, L., "Use Case Maps for the Capture and Validation of Distributed Systems Requirements". In: *Fourth International Symposium on Requirements Engineering*, Limerick, Ireland, June 1999.

(3) Ardis, M.A., Chaves, J.A., Jagadeesan, L.J., Mataga, P., Puchol, C., Staskauskas, M.G., and Olnhausen, J.V., "A Framework for Evaluating Specification Methods for Reactive Systems — Experience Report". In: *IEEE Transactions on Software Engineering*, 22 (6), 1996, 378-389

(4) Bolognesi, T., van de Lagemaat, J., and Vissers, C., *LOTOSphere: Software Development with LOTOS*. Kluwer Academic Publishers, The Netherlands, 1995.

(5) Braek, R., "SDL Basics". In: *Computer Networks and ISDN Systems*, 28, 1996, 1585-1602.

(6) Buhr, R.J.A. and Casselman, R.S., *Use Case Maps for Object-Oriented Systems*, Prentice-Hall, USA, 1995. http://www.UseCaseMaps.org/UseCaseMaps/pub/UCM_book95.pdf

(7) Buhr, R.J.A. (1997) "Use Case Maps as Architectural Entities for Complex Systems". In: *Transactions on Software Engineering, Special Issue on Scenario Management*, 24 (12), December 1998. http://www.UseCaseMaps.org/UseCaseMaps/pub/tse98final.pdf

(8) Courtiat, J.-P., Dembinski, P., Holzmann, G., Logrippo, L., Rudin, H., and Zave, P., "Formal methods after 15 years: Status and trends — A paper based on contributions of the panelists at the FORmal Technique '95 Conference, Montreal, October 1995". In: *Computer Networks and ISDN Systems*, 28, 1996, 1845-1855.

(9) Craigen, D., Gerhart, S., and Ralston, T., *Industrial applications of formal methods to model, design, and analyze computer systems: an international survey.* Noyes Data Corporation, USA, 1994.

(10) ISO, Information Processing Systems, Open Systems Interconnection, "LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", IS 8807, Geneva, 1989.

(11) ISO/EIC, Information Technology, Open Systems Interconnection, "Conformance Testing Methodology and Framework (CTMF)", IS 9646, Geneva, 1991. Also: CCITT X.290-X.294.

(12) ISO/EIC, "OSI CTMF Part 3: The Tree and Tabular Combined Notation", IS 9646-3: 1992, Geneva.

(13) ITU, "Recommendation X.680-683, Abstract Syntax Notation One (ASN.1)". Geneva, 1994.

(14) ITU, "Recommendation Z.100, Specification and Description Language (SDL)". Geneva, 1994.

(15) ITU, "Recommendation Z.105, SDL Combined with ASN.1 (SDL/ASN.1)". Geneva, 1995.

(16) ITU, "Recommendation Z. 120: Message Sequence Chart (MSC)". Geneva, 1996.

(17) Probert, R.L. and Monkewich, O., "TTCN: the international notation for specifying tests of communications systems". In: *Computer Networks and ISDN Systems*, 23 (05), 1992, 417-438.

(18) Quemada, J., *Working Draft on Enhancements to LOTOS*. ISO/IEC JTC1/SC21/WG1, "Enhancement to LOTOS" (1.21.20.2.3), January 1997.

(19) Steedman, D, *Abstract Syntax Notation One ASN.1: The Tutorial & Reference*. Technology Appraisals, Twickenham, UK, 1990.

(20) Weidenhaupt, K., Pohl, K., Jarke, Matthias, and Haumer, P., "Scenarios in System Development: Current Practice". In: *IEEE Software*, March/April 1998, 34-45.