Computer Networks and ISDN Systems 22 (1991) 1–6
North-Holland

Guest Editorial

Introduction to the PSTV-IX

Ed Brinksma, Giuseppe Scollo and Chris A. Vissers
Computer Science Department, University of Twente, 7500 AE Enschede, The Netherlands

1. The Ninth PSTV Symposium

The Ninth IFIP TC-6 International Symposium on Protocol Specification, Testing and Verification was held at the University of Twente in Enschede, The Netherlands, in June 1989. Approximately 170 participants attended the Symposium, testifying to the growth of interest in its goal, the application of formal methods to the design, description, analysis, implementation and testing of open systems. In this edition of the Symposium, the following themes are covered in particular:

- formal specification of switching systems using Z, Estelle and LOTOS;
- application of formal models: process algebras, transition systems, temporal logic;
- conformance testing: test generation, design and implementation of test systems;
- compilation and transformation of formal description techniques;
- tool environments; and
- verification by state space exploration.

A record number of 94 papers were submitted, out of which 26 were selected for presentation at the regular sessions. Three invited presentations introduced the audience to new topics: one on compositional methods in design and verification of real-time distributed systems, by W.P. de Roever and J.J.M. Hooman, one on open distributed processing, by J.J. van Griethuysen, and one on EEC research policies, by J.-J. Lauture. Both the regular papers and two of the invited papers (the technical ones) appeared in the Symposium Proceedings.

Besides the regular sessions and the invited presentations, the Symposium featured one day of tutorial presentations and two short workshop sessions. A selection of these events has given birth to the collection presented in this issue, consisting of a tutorial on formal methods in protocol conformance testing, two papers on experience with formal methods in protocol implementation and a paper on a meta-tool assisted design of a graphical G-LOTOS editor.

Before introducing the contents of this collection in more detail, it seems appropriate to recall the Symposium events which gave birth to this special issue. By giving not just outlines of what the papers of this selection are about, but also a recollection of the lively arena in which they were proposed in their first form, we hope to succeed in stimulating the reader's curiosity towards what these papers tell, the answers they give to relevant questions and the questions they put forward.

2. Tutorial presentations

The organization of a tutorial day was in the tradition of the Symposium since its seventh edition (Zürich, 1987). Three tutorials were given:

- F.W. Vaandrager introduced ACP, the Algebra of Communicating Processes, as an algebraic framework for the specification and verification of concurrent, communicating processes.
- G. Kahn and D. Clément presented the CENTAUR meta-tool system, which takes the formal syntax and semantics of a language as input and produces a collection of language-

specific tools with graphic man-machine interfaces.

- B.S. Bosik and M.Ü. Uyar surveyed formal methods in protocol conformance testing, from theory to implementation, discussing, on the basis of extensive experience, the impact of a number of such methods on practical aspects of conformance testing, such as testbed architecture implementation and automatic generation of test scripts.

3. Workshop sessions

The workshop sessions were a novel feature of this edition of the Symposium. They were organized to stimulate professional interaction among the participants and to promote cross-fertilization between fields that are in the tradition of the Symposium, on the one hand, and emerging topics of interest, on the other hand.

The first workshop session, chaired by G. von Bochmann, focussed on approaches to protocol implementation using formal methods.

The second workshop session, chaired by B. Pehrson, addressed graphics and interactive interfaces for formal description techniques.

Speakers of workshop sessions gave short, informal presentations of work in progress; a large part of each session was taken by lively discussion. The interaction that was established at the workshop sessions, and further co-ordinated by the workshop Chairmen, resulted in refined work and, in one case, in the integration of two formerly distinct contributions to the Symposium. Three papers were eventually selected for this collection.

Ed Brinksma (M.Sc. in Mathematics, University of Groningen, Ph.D. Computer Science, University of Twente, The Netherlands) joined the Computer Science Department of the University of Twente in 1982. Since 1988 he holds an associate professorship at the same department. From 1982 to 1988 Dr. Brinksma chaired the ISO committee that defined the formal specification language LOTOS, of which he is one of the principal designers. Dr. Brinksma has been and is involved in many international research projects such as COST 11 and ESPRIT projects (SEDOS, LOTOSPHERE). His main research interests are the theory of transformation, verification and testing of (descriptions) of distributed systems, and its applications, in particular in the context of open systems.

Giuseppe Scollo (Laurea in Electrical Engineering, University of Catania, Italy, 1977) joined the Computer Science Department of the University of Twente in 1985, where, since 1986, he is Associate Professor with the TIOS Group. His previous experience includes measurement systems and performance analysis of computer networks, protocol design, design of laboratory data acquisition systems, algebraic specification of OSI services and protocols. He is lecturer in abstract data types, architecture of interaction systems, and protocol design. His main research interest is in algebraic methods and design principles, that he develops in collaboration with the algebraic logic group of the University of Pisa, and contributes to several European projects within the COST and ESPRIT programmes. Since 1985, he is IEC/ISO Project Editor for the Formal Description of OSI Transport Standards in LOTOS.

Chris A. Vissers is a full professor in the faculties of Computer Science and Electrical engineering at the University of Twente, The Netherlands where he leads the "Tele-informatics and Open Systems" group. This group works on several aspects of telecommunication and open distributed systems, such as telematics applications, communication network protocols, transmission technology, distributed system architecture, formal methods, design support tools, performance analysis, and implementation techniques. His personal interests are in the area of distributed system architecture and the formal approach of design methodologies for open distributed systems.

The following reports summarize presentations and discussion at each workshop session.

3.1. Protocol implementation using formal methods

The session started with short presentations of four papers:

(a) *Incremental Development of an HDLC Protocol in ESTEREL*, by G. Berry and G. Gonthier, discussed stepwise refinements which lead from a protocol specification to an implementation-oriented specification which can be automatically translated into executable code.

(b) *Distributed Implementation of LOTOS Multi-Rendezvous*, by Q. Gao and G. von Bochmann, addressed the problem of implementing the LOTOS rendezvous synchronization in a distributed environment. An algorithm based on virtual rings is used for this purpose, and it is shown how the dynamically changing configurations of LOTOS processes and gates can be co-ordinated within a distributed environment.

(c) *A Distributed Algorithm for Synchronous Process Communication at Ports*, by P. Sjödin, addressed the same problem as the previous paper. An algorithm is proposed for a fixed number of synchronizers, each representing a LOTOS process, and a fixed number of ports, each representing a LOTOS gate. The algorithm provides the necessary co-ordination between synchronizers and ports in order to schedule the LOTOS interactions.

(d) *Implementing Protocols with Multiple Specifications*, by S.C. Murphy, P. Gunningberg and J.P.J. Kelly, reported on experience with the use of specifications in Estelle, LOTOS and SDL, for the implementation of a simple sliding window protocol and of the OSI class 0 Transport protocol.

Among other questions, the discussion addressed the following ones:

1. Feasibility and usefulness of automatically deriving an implementation from a given specification.

The automatic implementation goal was stressed by the first speaker, who observed that the efficiency of the obtained implementation will in general depend not only on the specification compiler and on the specification language, which may allow for more or less efficient implementation strategies, but also on the structure of the specification—which may be optimized through transformations. The problem of automatic implementation from a given specification was touched also by several other presentations at the Symposium.

2. Whether the automatic, distributed implementation of LOTOS specifications, as aimed at in the second and third paper, is of any practical value.

The proponents of those implementation approaches argued that, first, the proposed algorithms address an important problem of distributed systems in general and that, second, those algorithms could be the basis for automated prototyping in a distributed environment.

The opponents remarked that most LOTOS system specifications would have to be transformed into a more implementation-oriented style, before an acceptably efficient implementation could be obtained. Much research in software engineering in general, and applied to LOTOS in particular, is centered on the question of specification and program transformations in the context of the software development cycle. These transformations would usually require human intervention and could include those implementation aspects that relate to the distribution of the system and communication between the different system components residing on different sites.

It was concluded that both views have their merits: automatic implementation strategies could be used for prototypes, where efficiency is less important; they may lead to more efficient implementation strategies for specific system structures. On the other hand, methods developed for specification and program transformations may be applied to obtain implementations that better fit particular environments or meet given performance requirements.

3. Whether formal specifications should be taken alone, or accompanied with some informal overview, describing the purpose of the specified system and its general structure and/or mechanisms.

The question was raised by the observation, made in the fourth paper, that implementers had difficulties in understanding the overall sense of the formal specifications which were the basis of their implementation projects. Therefore, they generally adopted an implementation approach where each 'module' of the specification was translated into a 'module' in the implementation, even if the structure of the specification was not

necessarily intended as the structure of the implementation.

The reported experience showed that the intended behaviour of each 'module', separately considered, was well understood by the implementers, since the resulting implementations were largely correct.

The discussion concluded that a well-engineered combination of informal outline and formal description was likely to yield the best results.

3.2. Graphics and interactive interfaces for FDTs

The session started with three short presentations:

(a) *Generation of Graphic Language-Oriented Design Environments*, by R. Backlund, P. Forslund, O. Hagsand and B. Pehrson, presented an extension, to include graphic syntaxes, of the meta-technique approach to interactive programming environment generation that is based on attribute grammars—an approach originally defined by the Cornell Synthesizer Generator. The meta-tool LOGGIE, based on their approach, was also presented, and afterwards demonstrated together with a first application: the Graphical front-end to the Concurrency Workbench, that is an editor and simulator for Milner's CCS.

(b) *Adding Graphics to Estelle*, by D. New and P.D. Amer, presented GROPE, a tool for graphically animating the dynamic execution of an Estelle formal specification. The syntax included interconnected modules with finite state machine behaviours. The speaker argued that a graphically animated presentation of the simulated specification is much more effective than a textual presentation; especially, in that it helps to focus the observer's attention on the most pertinent activities of the system being observed. A large amount of information that is present in a complete specification is so suppressed, e.g., redundant information in modules that is needed for strict type checking, comments, etc.

(c) *Techniques for the Formal Definition of the G-LOTOS Syntax*, by T. Bolognesi and D. Latella, introduced a number of such techniques as the title says. A subset of LOTOS was considered. Examples of pictures suitable for presenting this subset graphically were described in Pictor, an algebraic language for picture definition. An extended grammar approach to the definition of G-LOTOS was contrasted with an approach based on an abstract syntax for LOTOS. The usefulness of definite clause grammars was considered as well.

The discussion addressed two questions:

1. Actual need or use of graphic user interfaces.

This subject got the largest part of the discussion. Main conclusions were:

- A graphic syntax makes a specification more comprehensible, especially when you are exposed to it for the first time, e.g., when reading a protocol standard document, in learning, etc. However, complex specifications are incomprehensible, be they graphical or not. They have to be broken down into manageable parts.
- Unexperienced designers prefer a graphic syntax, stressing comprehensibility, while experts at work want compact, expressive notations, stressing rapidity of man–machine communication. Textual notations seem so far mostly superior from the latter viewpoint.
- It seems important to create a feeling of direct manipulation of the basic semantic concepts in the tool user interface. If the semantics of the specification language is graph-oriented in some sense (e.g., transition systems), a graph-oriented syntax is useful. The abstract terminal symbols could be states, state transitions, labels, etc., and the concrete syntax would then describe two-dimensional graphs with circles as nodes, arcs as transitions, and text strings as labels. If it is natural to think about the semantic concepts as strings (e.g., logical formulas and derivation rules), then a textual syntax could be preferable. The abstract terminal symbols are in this case atomic propositions, logical operators, etc., while the concrete syntax is based on text strings.

2. Maturity of meta-techniques for their adoption in interactive design of realistic tool products.

A rather limited discussion reached the following conclusions:

- Meta-techniques are mature, as far as interactive languages based on a textual syntax are concerned. A good example of this was actually presented at the Symposium, in the tutorial day.
- Meta-techniques are not yet mature in the more general case, viz. including languages that have a graphic syntax. However, work is in progress, as exemplified in two of the presentations given at this session. Eventually, such techniques will reach maturity.

4. Contents of this Special Issue

Formal methods are profitable not only for specifying network protocols and entities but also, perhaps even more so, for testing the conformance of network products to their formal specifications. This is not an academic statement anymore, it is a fact. Industrial awareness of this is a fact as well, as testified for example by the effort that the major standards organizations in the field, ISO/IEC and the CCITT, put nowadays into the definition of a formal methodology for conformance testing.

The question then arises, to what extent are existing theories and formal methods viable for practical implementation. The tutorial by Barry S. Bosik and M. Ümit Uyar, *Finite State Machine Based Formal Methods in Protocol Conformance Testing: From Theory to Implementation*, which opens the present collection, is the result of several years of experience in looking for answers to this question.

Not surprisingly, the authors limit their survey to methods that are based on Finite State Machines. These form not only the most developed branch of relevant theory, but also the basis of the methods that have received more extensive implementation attempts in industry. The tutorial offers a comparison of four major formal methods for conformance testing generation that are reported in the literature: transition tours, distinguishing sequences, characterizing sequences and unique input/output sequences. Moreover, it illustrates the implementation of each of these methods.

The tutorial also addresses the impact of formal methods on other practical aspects of conformance testing, including implementation of the test system and automatic generation of test scripts, reports on experience with testing several protocol implementations from several manufacturers, and explores the relationship between formal methods for test generation and standardized methods and principles relating to conformance testing.

Not less relevant to industrial application of formal methods seems to be the problem that Gérard Berry and Georges Gonthier tackle in their *Incremental Development of an HDLC Entity in Esterel*, and that essentially is the protocol engineering problem, but addressed within a novel implementation framework. The most intriguing aspect of their work comes indeed from the novelty of the implementation technique, which is a parallel programming language that supports synchronous communication.

Engineering formally specified protocols in such a framework has a number of veritably interesting consequences. First, features such as parallelism and synchronous communication in formal specification techniques are no implementation problem. Second, efficiency of the implementation becomes insensitive to degree of modularity, parallelism and internal communication, because the related scheduling and synchronization are dealt with at compile-time. Third, perhaps most interesting, a new programming style becomes available. The paper illustrates this style in a tutorial fashion, with emphasis on architecture, design and maintenance of the implementation.

Conversely, the interest of *Experiences with Estelle, LOTOS and SDL: A Protocol Implementation Experiment* by Susan C. Murphy, Per Gunningberg and John P.J. Kelly, is mainly at the other, more abstract side of the protocol implementation problem, that is, the specification side. The paper thus presents form and results of an experiment on implementing protocols from formal specifications written in standard FDTs.

Six implementation projects were defined on a matrix of two protocols (a simple sliding window protocol and the class 0 transport protocol) and the three standard FDTs. In each project, a group of graduate students implemented the given protocol based only on the knowledge of its formal specification written in the given FDT, and of additional information (encoding rules) common to all groups.

To the best of our knowledge, this is the first experiment of that kind, thus neither form nor results should be taken as definitive. Yet, both form and results prove highly informative. Moreover, besides the inherent, informative interest, the perhaps most interesting fact about this paper is that it *cannot* be definitive, and thus invites repeating the experiment—following genuine canons of scientific investigation.

Such canons of formal engineering are obeyed by *The Definition of a Graphical G-LOTOS Editor using the Meta-Tool LOGGIE*, by Tommaso Bolognesi, Olof Hagsand, Diego Latella and Björn Pehrson. The subject of the paper is a prototype implementation of a graphical editor for a subset

of G-LOTOS. Formal is the implementation object, that is a formal description technique, albeit in the graphical form the standardization of which is underway by ISO/IEC and CCITT. Formal is the definition of the G-LOTOS subset that is chosen for the implementation target. Formal is the specification of the graphical editor that is derived from that definition. Mechanical, that is formal in the most strict sense of the word, finally is the generation of the graphical editor from its specification—thanks to the meta-power of the tool under consideration in the paper. What more formal usage of the marvels of electricity could ever be shown?

The reader might object here: all that may be nice, but why should such a language implementation story deserve the interest of protocol engineers or network managers? Just because (the use of) the particular language (perhaps) deserves it?

No, certainly not for such a dubious syllogism. However, it nowadays happens that protocol engineers find themselves more and more often involved in tool design, e.g., to let a program carry out a task that a human mind finds boring after all; or that network managers find themselves more and more often forced to promote or evaluate tool design projects, whose degree of formality and mechanizability directly affects feasibility, cost and chances of success. Thus we feel obliged to conclude this presentation with the following 'editorial' remark. It is not by chance that such a paper closes the present collection. This editorial choice, concerning relative ordering, is intended to signify the open-endedness of the area of interests that are relevant to protocol specification, testing and verification.

5. Related literature

Besides the obvious reference to the Symposium Proceedings [2], a few other papers are closely related to presentations given at the Symposium. These are:

- the addition of graphics to Estelle, that is addressed in [4];
- the tutorial on the CENTAUR system, an introduction to which appears in [1];
- the experiment with multiple specifications, other aspects of which are presented in [3].

Acknowledgements

We gladly thank Gregor v. Bochmann and Björn Pehrson, who contributed the reports on the workshop sessions they respectively chaired.

All of the four papers that form the present collection were improved by acute, constructive criticism by the referees, to whom, also on behalf of the readers, we extend a hearty thanks for their outstanding contribution to the quality of this selection.

References

[1] P. Borras, D. Clément, Th. Despeyroux, J. Incerpi, G. Kahn, B. Lang and V. Pascual, CENTAUR: the system, in: Henderson, P., ed., *Proc. SIGSOFT'88, 3rd Annual Symposium on Software Development Environments*, Boston, Nov. 28–30, 1988 (ACM Press, Boston, 1988) 14–24.

[2] E. Brinksma, G. Scollo and C.A. Vissers, eds., *Protocol Specification, Testing, and Verification, IX* (North-Holland, Amsterdam, 1990).

[3] J.P.J. Kelly and S.C. Murphy, Achieving dependability throughout the development process: A distributed software experiment, *IEEE Trans. Software Eng.* **16**(2) (1990) 153–165.

[4] D. New and P.D. Amer, Adding graphics and animation to Estelle, *Inform. Software Technol.* **32**(2) (1990) 149–161.