Some Notes on the History of Protocol Engineering

Gregor v. Bochmann[1], Dave Rayner[2] and Colin H. West[3]

During the 1970ies and '80ies, the first computer communication networks were designed and implemented in the research and commercial sectors. Many of the protocols developed during that time are still in use today. This paper starts by giving an overview of these developments. Then it concentrates on the development of protocol engineering, that is, the methods for the specification of communication protocols and services, the verification of protocols and their implementation and testing. After personal views of the developments in the 1970ies, the basic concepts developed at that time are explained. The standardization of Formal Description Techniques in the 1980ies is discussed in the following section, as well as the standardization of conformance testing. The purpose of the paper is to show the long way we have come and to suggest that many of the basic concepts have not changed too much during these years, although more detailed aspects have evolved and given rise to new technological developments.

1. Introduction

The FORTE conference, together with its ancestor - the PSTV (Protocol Specification, Testing and Verification) conference, is now in its 30th year, and is now combined with the FMOODS conference which was established in 1996. The authors feel that this is a good occasion to look back to the early times of communication protocols. The purpose of this paper is to provide some of our personal reflections about the early times of protocol engineering in the 1970ies and '80ies.

While many new communication protocols were developed during these early times of computer and data communications, some people also became concerned with the question of what methods are best suited for defining protocols, for verifying that they have the desired properties, for developing an implementation (which could be in different hardware/software environments), and for testing a given protocol implementation for conformance with the protocol definition. This preoccupation is reflected in the name of the PSTV conference and has been called "protocol engineering" [Boch 93c].

In this paper, we first give an overview of the development of computer communications protocols in the 1970ies and '80ies. In Section 3, we give then some personal views of the 1970ies and explain the basic concepts of protocol engineering that were explored and defined by the research community during those years. In Section 4, we give a short overview of the development of Formal Description Techniques (FDTs) for communication protocols and services, which took place during the 1980ies in the context of the standardization of protocols for Open Systems Interworking (OSI). The section also gives an overview of the standardization of protocol conformance testing. The conclusions discuss the impact of these developments on the current technological trends.

[1] Gregor v. Bochmann <bochmann@site.uottawa.ca> is with the School of Information Technology and Engineering (SITE) at the University of Ottawa, Canada.

[2] Dave Rayner <dave-rayner@ntlworld.com> was at the UK's National Physical Laboratory from 1975 until 2007

[3] Colin West <cw@ieee.org> retired from the IBM Zurich Laboratory in 1999.

2. Computer communications in the 1970-80ies

2.1. Initial developments

The 1970ies and '80ies were stimulating times for people working in the area of computer communications. In the late '60ies, the first link-level protocols were introduced for remote access, over leased lines, from terminals to host computers. In 1968, Lynch [Lynch 68] described a protocol with positive and negative acknowledgments which was not fully reliable, as pointed out by Bartlett et al. in 1969 [Bart 69]. During the same time, IBM had introduced the Bisync protocol which was used for a long time (a superficial look at its definition seems to indicate that it had similar flaws to the protocol of Lynch).

Then in the early '70ies, a new generation of link protocols was introduced with bit-oriented framing and sequence numbering (initially represented by 3 bits with values from 0 to 7). IBM's SDLC strongly influenced the international standard HDLC developed within the International Organisation for Standardization (ISO). This protocol was used in 1976 by the International Telegraph and Telephone Consultative Committee (CCITT) for their standard network access protocol, called X.25. Note: IBM, at that time, had a dominating market position as computer and software manufacturer, similar to Microsoft in the software domain nowadays.

In the late 1960ies, several research projects had been established to experiment with the concept of packet switching. The first two operational packet switched networks were the ARPANET in the USA and the NPL network at the National Physical Laboratory in the UK. Larry Roberts led the ARPANET team that had the first nodes of a wide area network operating by December 1969. Donald Davies led the NPL team that had the world's first local area network operational by 1970. Donald Davies coined the term "packet switching" in 1966 and co-invented the concept along with Paul Baran of the original ARPANET team. ARPANET is seen as being the forerunner of today's Internet. The French Cyclade network project, led by Louis Pouzin, introduced the concept of an unreliable datagram service at the Network layer over which an end-to-end Transport layer protocol would build a reliable connection-oriented service, sometimes called "virtual circuits" [Pouzin 1973]. These concepts were later adopted for the IP/TCP protocol hierarchy of the Internet as we know it today.

During this time, it also became apparent that there needed to be standards for interconnecting different packet-switched networks. Protocols that could be used for this purpose were called Internet protocols. In international group of experts was formed in 1972 under the name INWG[4] which became affiliated with the Technical Committee for Communications Systems (TC-6) of the International Federation of Information Processing (IFIP), as Subgroup 6.1. This group met quite frequently and elaborated a set of Internet protocols which included precursors of IP and TCP, a virtual terminal protocol and others. Gregor Bochmann participated in some of these meetings and had the impression that Vint Cerf from ARPANET and Louis Pouzin from Cyclade were the principal leaders of the group. Towards the end of the 1970ies, these IP/TCP protocols were implemented for the interconnection of Ethernets and satellite networks on an international basis, and the original protocols of the ARPANET disappeared.

2.2. Standards for computer networks

At the same time, the common carrier companies and governmental PTT ("post-telephone-telegraph") organizations had realized that standards for commercial packet-switched networks were required and started to discuss possible approaches within CCITT (which later became ITU-T). Following their tradition of telephone network technology, they opted for a network service of

4 The International Network Working Group (INWG) was created at the ICCC in 1972 and elaborated proposals of protocol standards for network interconnection. This led to the Internet protocols. An account of these developments and the emergence of the Internet as the general networking infrastructure in the 1990ies is given in [Hauben]. A few years later, the INWG group became Subgroup 6.1 of IFIP TC6 and as such could participate in the discussion on X.25 within the CCITT.

reliable virtual circuits. This meant that at the interface between a host computer and the network, there was the requirement of supporting a large number of such circuits with separate flow control. Therefore the interface standard (strictly a "CCITT Recommendation") X.25 that was adopted in 1976 was much more complex than what would be required for providing a datagram service.

During the mid-70ies, there was much polemics concerning the question whether a datagram or a virtual circuit network service would be more appropriate for packet-switched networks. The INWG group had submitted their proposal for datagram and end-to-end Transport protocols to CCITT and tried to convince the PTT experts that their approach was preferable over the virtual circuit approach. However, CCITT stuck to the virtual circuit approach and produced in X.25 a standard network access protocol supporting several multiplexed virtual circuits. Later, a similar standard, called X.75, was defined for the interconnection of different packet-switched networks.

Although the datagram approach had not been adopted by CCITT for the network access protocol standard, discussion continued on whether this technology would be advantageous to be deployed within a network, internally. The Datapac network developed in Canada by Bell-Northern-Research did use internally datagrams with alternative routing. On the other extreme was the French Transpac network which used virtual circuits internally - where each circuit, when established, was allocated to a fixed route. Arguments were related to resilience against different types of hardware faults and resource requirements in terms of memory requirements in the switches (in case of Transpac) and transmission overhead due to lengthy packet headers (in case of datagrams, plus the Transport protocol required for reliability).

2.3. Proprietary computer networking protocols, such as SNA

In the meantime, different computer manufacturers had also started developing proprietary protocols for computer communications. The previous decade had seen batch-oriented computer systems evolve towards interactive systems. A systematic approach to providing shared communications facilities that permitted large numbers of terminals to access a variety of applications running on large host machines was needed.

In 1974 IBM announced Systems Network Architecture (SNA) as its general methodology to enable users to access data processing services and devices in a computer network. SNA was defined as a layered architecture including data link control, routing, flow control, management of shared resources and presentation services.

SNA was initially documented largely by means of informal prose and descriptions of sample message sequences. It was soon realized that this lacked the precision necessary to describe a system of this complexity. By 1976 SNA was described by defining the functional structure of a network node whose components were defined in terms of states, state machines and combinatorial functions [Schultz 75]. The methodology was defined by T.F. Piatkowski [Piat 75]. The SNA definition was unique at the time in terms of its scope and precision. It had significant graphical content which made it easy to read and understand.

Product implementers and testers were however anxious to have a machine interpretable definition of the SNA architecture and by 1978 a third definition was developed that preserved the finite state machine structure but represented it in terms of a programming language FAPL [Schultz 75] which was an extension of PL/1 augmented with tabular finite state machines and constructs for managing messages.

2.4. Application protocol standards

Given that proprietary computer networking protocols had been developed by different manufacturers – besides SNA, similar protocols had been developed by Digital Equipment Corporation, Honeywell and others – interworking between systems provided by different manufacturers was difficult. In order to simplify the interworking between such systems, the OSI standardization project was initiated within ISO in 1977. It was mainly aimed at standards for distributed applications that would use the X.25 standard as the basic communication service over

which the application protocols would run. A collection of articles describing the situation around 1980 was assembled by Paul Green [5].

The work on OSI extended over more than 15 years until the mid to late 1990ies. As usual, not all parties participating in the standardization meetings were really interested in successful standards, and there was a variety of different interests, which led to relatively complex standard proposals that had options for all kinds of situations to replicate or accommodate pre-existing protocols. For instance, much discussion took place about the Transport protocol, which is not really required over X.25, but which is required if the datagram service is used in the network layer. Because of the input from the Internet community, a Transport protocol Class 4 was introduced, quite similar to TCP. In fact, there were altogether 4 classes of Transport protocols.

We note that the ARPANET (later evolving into the Internet) has been continuously in service for the university and research sector within the USA and in some other countries since the mid 70ies. Later these protocols were supported by the Berkeley Unix software which was available free of charge within the research community and became widely used in this context. This continuous use of the Internet is the main reason why it finally became the de-facto standard for the current global Internet – much helped by the World Wide Web application (originally developed at CERN, Geneva) in the early 90ies.

To finish this section, let us mention just two other interesting protocol developments in the 1980ies. In 1980, CCITT had defined the Teletext standard. It may be called the Web standard of the ‘80ies. Like the current Web service over the Internet, Teletext was based on the Hypertext concept [Nelson 65] with its pages, links and anchors. The service was implemented, for instance, in the UK, France and Germany, and the Canadian version included graphic images. The main problem was the lack of suitable terminals to display the pages (since desktop computers were not very common at that time). Special Videotext terminals were manufactured and could be rented. In the UK and elsewhere the service was broadcast to TV sets, which some organizations like NPL connected up to computer networks to make the service accessible from any computer terminal.

Within the OSI framework, many application-layer protocols were developed. For the description of these protocols, aspects related to the data structures of the parameters of service primitives and protocol messages are usually much more important than the temporal ordering of interactions specified by state machine models. To describe these data structures, the ASN.1 notation was developed and was used for most OSI application protocols. Also a Remote Operations standard was developed to invoke operations on remote servers. (Note: this was probably influenced by an earlier protocol for Remote Procedure Calls developed at the Xerox PARC research center in California). ASN.1 also includes standards for encoding the structured information. Several encoding standards were developed, based on octet-oriented encoding of tags and lengths indicators. In the mid-‘80ies, several tools (e.g. [Boch 90t]) were developed for automatically generating encoding and decoding routines for protocol messages defined using this description method. More recently, CORBA was developed for the same purposes. Nowadays, XML is often used for this purpose (e.g. for Web Services), and associated encoding and decoding tools are available from various sources.

---

5 The special issue of the IEEE Transactions on “Communications on Computer Network Architectures and Protocols” (April 1980) edited by Paul Green. It included also several papers on protocol engineering, such as “Formal Methods in Communication Protocol Design” by Bochmann and Sunshine, “Protocol Representation with Finite-State Models” by André Danthine, “Towards Analyzing and Synthesizing Protocols” by Zafiropulo, West, Rudin, Cowan and Brand, “Executable Description and Validation of SNA” by Schultz, Rose, West and Gray, “A General Transition Model for Protocols and Communication Services” by Bochmann, and “OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection” by Hubert Zimmermann.

3. Development of protocol engineering methods in the 1970ies and early '80ies

3.1. A personal view of early development

Bochmann had been working in the early 1970ies on compiler design, semantic attributes and methods for defining the semantics of programming languages. Now he was looking for a younger field of investigation with more open problems. In 1973 or '74, he attended a conference on computer communications where he met Louis Pouzin who was talking about the need for standard communication protocols to allow the interworking between different computer networks developed in different parts of the world. Pouzin pointed out that good methods for precisely defining the communication protocols where needed and also methods for verifying that they provide the communication service that is expected. The question of how these protocols and services could be formally defined was not known.

This encouraged Bochmann to experiment with different specification paradigms. On the one hand, he explored the use of finite state machines [Boch 75d], inspired by the paper by Bartlett et al. [Bart 69] on the alternating bit protocol which used a state machine formalism, and he developed the method of protocol verification by reachability analysis which was later applied to the X.25 protocol standard [Boch 78i]. On the other hand, he tried a specification style using a programming language and assertions such that properties of specifications could be proven using program proving techniques for concurrent systems [Boch 75c]. It was clear that certain protocol features, such as sequence numbering, cannot be adequately described by state machines, although the state machine approach appears to be natural for many aspects of protocols. Therefore he proposed an approach combining both methods in the form of an extended FSM model [Boch 77c]. When he presented this paper 1977 at the IFIP Congress in Toronto, he met Pitro Zafiropulo who, independently, had done similar work with Colin West at the IBM Research Lab in Zurich (see below).

The Computer Network Protocols Symposium [Danthine 1978] organized in February 1978 by André Danthine in Liège was the first international conference dedicated to the design of communication protocols. It was a very interesting meeting place for industrial and academic researchers working in this field. There were many papers on methods for defining and verifying protocols, and also many papers related to particular protocols, in particular to the newly adopted CCITT recommendation X.25 for computer communications. Several of the papers presented at this conference were later published in enhanced versions in Computer Networks[6]. Carl Sunshine presented a paper surveying different methods for protocol specification and verification, which was later expanded into a journal version [Boch 80a].

3.2. An industrial perspective

In any developing scientific or technological field, industrial companies have a variety of interests that define the nature of their contributions. They benefit from and contribute to general advances in the science and technology. Company success often depends on proprietary technology that distinguishes them from their competitors. They also have to satisfy their customers and provide timely solutions to customer problems.

In IBM, the development of SNA was required to satisfy the needs of the customers for improved communications services and was seen as an important technology that represented a competitive advantage. During he development of SNA, it became clear that the methods used to describe and design such systems had yet to be developed. It was also clear that validating the

---

6 A special issue of Computer Networks (Vol. 2, No. 4/5) included many enhanced versions of papers from the Liège conference including several on protocol engineering, such as "Finite state description of communication protocols" by Bochmann, "Automated protocol validation: One chain of development" by Rudin, West and Zafiropulo, "Verfication of protocols using symbolic execution" by Brand and Joyner, and "Modeling and verification of end-to-end transport protocols" by Danthine and Bremer.

correctness of communications systems was a difficult task. This resulted in a research project being started in the IBM Zürich Laboratory.

In this context, Pitro Zafiropulo developed a theory for validating communications protocols [Zafi 78] which considered a communications protocol as an interaction between two communicating finite state machines, whose state transitions could be accompanied by the sending or receiving of a message. The protocol was validated by enumerating all of the possible message sequences that each machine could execute between an initial and final state and then analyzing all possible duologues, that is, pair-wise combinations of message sequences executed by the two machines. The analysis assigned each duologue as belonging to one of three categories. Firstly, the duologue could be completely executed with each of the state machines having received all of the messages transmitted on the other. Secondly, not executable because an attempt to duologue would inevitably result in the execution of another duologue, because for example complete execution would imply the reception of messages that had not been sent. The final classification was erroneous which occurred when execution of a duologue would imply the reception of an incoming message in a state when there was no provision for it, a situation later called “unspecified reception”.

Colin West developed an automated tool that incorporated the general principles of Zafiropulo’s Duologue Matrix Theory, which was formulated in terms of predicate calculus, not a suitable starting point for the development of a validation tool. He started out “playing” with the APL programming language representing each duologue to be analyzed as a pair of vectors of positive and negative integers representing the messages sent and received by each state machine. He came up with a sequence of APL operations on matrices derived by outer products form the vectors, such that the result contained all possible states of the message queues and the corresponding states of the two state machines [West 78c].

During the development of this Duologue Matrix tool, he realized that an alternative validation approach based on the reachable states would be more effective. Here therefore built a second tool that performed what is now known as reachability analysis. Starting from an initial system state, the global system states reachable by executing all possible state transitions were computed. New system states reached were added to a database of states if they had not previously been found. Deadlocks were detected if a given state contained no executable transitions that lead to other states, and reception errors were detected when here was no provision for processing an incoming message. The analysis terminated when all transitions from all reachable systems states had been analyzed.

The Duologue Matrix tool was used to validate the X.21 protocol [West 78b], found several errors and so demonstrated for the first time that significant results could be obtained from automated protocol validation. The reachability analysis tool duplicated the X.21 results and required an order of magnitude less computing resources for this particular problem.

The tool had other significant advantages. It could validate protocols between more than two state machines and also protocols that contained arbitrary numbers of cycles. It could also be easily extended to other queuing disciplines and network topologies. This meant that it could be used to validate the IBM Token Ring protocols [Rudin 82].

The work a the IBM Zurich Laboratory in the mid 1970's resulted in a theory of protocol validation, demonstrated that validation techniques could be automated and usefully applied to significant problems and set the stage for subsequent developments in the field.

SNA was further developed in the 1980s and remained the focus of the protocol validation work in the IBM Zurich Laboratory. Early SNA Networks were largely static with a single System Services Control Point to control the network. End-user terminals, such as printers and display terminals, incorporated minimal processing capabilities. Subsequent enhancements to SNA included program-to-program communications (LU 6.2) [Gray 83] and Advanced Peer-to-Peer Networking (APPN) [Green 87] that provided SNA support for more general dynamic networks of smaller computers. These extensions of SNA were accompanied by a change in the nature of the architecture definition. The early single thread execution model was replaced by a concurrent process model that reflected more closely the nature of the implementations being produced.

This had a significant impact on the nature of the validation that could be performed because much state information defining the model execution was incorporated in its runtime support.

The disjoint execution required for reachability analysis was no longer possible. Instead, validation was performed by testing a model of the SNA network. For this purpose, a model of an APPN network was created in a simulated environment and the application programming interfaces in the network were driven with random inputs that generated multiple concurrent sessions between simulated network users. Although conceptually simple, this technique proved to be extremely successful and resulted in a very significant improvement in the quality of the architecture, although it could in no sense prove correctness.

This validation work using a testing approach contrasts with the work of Holzmann who had at the time developed the SPIN system [Holz 91] which was capable of analyzing extremely large numbers of system states. The APPN architecture definition consisted of 40 000 lines of code and would have been impossible to transform into a form that allowed it to be subjected to the type of analysis performed by SPIN. Furthermore, the number of systems states that could have been generated would have been literally astronomical so that a complete analysis would not have been possible.

Examination of the errors found during the APPN validation showed that all of the errors could be expressed in terms of a few of the parameters that collectively defined the system state and so could occur in many different states of the system. This explains why the random analysis that did not completely explore the state space was able to discover so many errors [West 92].

Work on the executable SNA model was discontinued in the early 1990s for several reasons. The architecture had reached a relatively mature state, had been implemented in many products and was well understood. Hardware and software technology had reached the stage where it was often simpler to port code from an existing product rather than develop new code from the architecture definition.

The explosive growth of the Internet and TCP/IP also meant that the communications environment was evolving away from proprietary systems. However, these changes had little impact on the higher layers of SNA that had become important components of many applications.

In 1991 Network World published a quote by a Microsoft consultant who said "The U.S. banking system would probably collapse if a bug were found in (IBM's) LU 6.2". LU 6.2 was the SNA component that embodied the transaction processing support that was crucial to many industries, particularly banking, and is still widely used today. The validation of the SNA architecture that contained it has contributed significantly to the reliability of today's transaction processing systems.

3.3. Some important concepts

Many of the important concepts for protocol engineering were elaborated during the 1970ies and early 80ies. In the following, we will highlight some of these developments.

3.3.1. Protocol layering – communication service

The concept of protocol layering became well-known through the OSI standardization effort which defined seven protocol layers. These ideas were first developed within the French Cyclade network in the early 1970ies, distinguishing between an unreliable datagram network service and a reliable end-to-end transport service with corresponding protocols (see for instance [Pouzin 1973]). A protocol layer is characterized by its protocol and the communication service it provides. The importance of both, protocol and service descriptions, was clearly explained in our survey with Sunshine in 1980 [Boch 80a] [7]. The first formal models of communication services

[7] The importance of the specification of the communication services was further elaborated by Luigi Logrippo and Chris Vissers in 1985 [Viss 85]. [Boch 80e], makes the distinction between local properties (related to the interactions at a single service access point – and therefore locally observed by each protocol entity) and global properties (relating interactions at different service access points – the properties of the communication service "proper"). This concept was generalized in a paper with Michel Raynal [Boch 83a] to arbitrary multi-component architectures.

were perhaps the (very simple) service of the alternating bit protocol shown in Figure 7 of Bochmann's paper on finite state protocol description [Boch 78i] and the queue model of a reliable communication service in [Boch 75c] and [Stenning]. The formalized description of more complex communication services was discussed in [Boch 80e].

3.3.2. Interworking between heterogeneous networks

With the development of experimental packet-switched networks in several countries during the first half of the 1970ies, and the development of commercial packet-switched networks in the second half, the interconnection of different networks and their interworking became a major issue (see for instance [Green 86]). The first paper we remember on the principles of interworking between heterogeneous networks was written by Gien and Zimmermann in 1979 [Gien 79]. They pointed out that the interconnection must be realized at the service level; it requires compatible services within the two interconnected networks. Much of the later work did not take this into account. Bochmann's papers from 1990 [Boch 90b, Boch 90i] tried to put these principles into a formal setting.

3.3.3. The role of protocol specifications

Towards the end of the 1970ies, it became clear that a precise protocol specification is important because it should be (a) the starting point for the verification of the correctness of the protocol design (against the service specification), (b) the basis for the development of implementations on different computers that would be compatible with one another, and (c) the reference for developing test cases by which a given implementation could be checked for conformance with the protocol specification. This fact is reflected in the title of the IFIP conference series on Protocol Specification, Testing and Verification (PSTV) which started in 1981, and also led to the formation of a working group on Formal Description Techniques (FDT) for OSI Protocols and Services within the OSI standardization effort of ISO (see below). At the 1978 workshop in Liège, John Day talked about the need for precise protocol specifications in the context of the OSI standardization, which had just started the year before.

One important aspect of protocol specifications is their relative abstractness. In particular, the interfaces of a protocol entity with the layer below and with the user in the layer above must be specified in an abstract manner in order to allow different realizations of these interfaces for different implementations. This led to the concept of "service primitives" which was used to define the OSI layer services. Abstractness is also required for the description of the dynamic behavior of protocol entities; for instance, often different acknowledgement schemes are allowed for a protocol and could be implemented in different manners.

3.3.4. Protocol verification and reachability analysis

It soon was apparent that one has to be very careful when designing a new communication protocol. Already the paper describing the alternating bit protocol in 1969 [Bart 69] pointed out a bug in the design of an earlier bit-based protocol. Sunshine's surveys on protocol description and verification methods[8] (and his collection of articles [Sunshine 1981]) show the many different approaches that were used to specify the properties of the dynamic behavior of protocol entities and related methods to show that their communications would lead to an overall desired behavior.

This work led to the distinction of general properties that each protocol should satisfy and specific properties that are required by the communication service that is intended to be provided by the protocol [Boch 80a]. The general properties state the absence of certain flaws, including well-known concepts such as deadlocks, but also the concept of "unspecified reception", first described by the IBM Zürich team. The first proofs that some (very simple) specific properties of the communication service are satisfied were included in Bochmann's papers of 1975.

---

[8] Carl Sunshine published surveys on protocol engineering in various conferences [Suns 76, Suns 78] and also the journal paper [Boch 80a]. A bibliography assembled by John Day was also included in the proceedings of the Liège conference. The most important collection of articles in this area is probably the book "Communication Protocol Modeling" [Suns 81] edited by Sunshine in 1981.

The term “reachability analysis” has been used to describe a verification approach where one considers a global system consisting of the protocol entities and the underlying communication medium and where the protocol specification is used to determine the possible state transitions that may occur in the global system. Reachability analysis then consists of exploring all possible global states that can be reached from the initial state of the system. By analyzing this global reachability graph, one can determine whether the general properties and certain service requirements are satisfied. We note that similar ideas had been proposed in a more general context[9]. In the context of communication protocols, the state transitions are usually associated with the sending or receiving of protocol messages or with the loss or error in the transmission medium. This led to certain models for the communication medium that were incorporated in certain verification tools, and specific types of errors, such as “unspecified reception”.

While the reachability analysis for certain simple protocols have been done “by hand”, any real-life application of this method requires automated tools because of the large number of reachable global states. As mentioned above, the first tools for this purpose were developed within IBM Zurich. Later many other tools have been developed, not only for finite state machine models, but also for many other specification formalisms. Holzmann’s SPIN tool [Holz 91], initially developed in the early 1980ies and continuously improved, introduced the so-called bit-state representation where, through hashing, a global state is represented by a single bit; this allows the analysis of very large state spaces. Many of these tools have been extended to the realm of model checking, where in addition to the detection of deadlocks and unspecified receptions in the model, the designer can specify specific properties in temporal logic that will then be checked by the tool.

3.3.5. Constructive methods for protocol design

While protocol verification is intended for checking whether a given specification of a protocol has all the desired general properties and satisfies the requirements of the desired communication service, some researchers explored constructive methods for protocol design. The objective was to find methods for constructing automatically, or semi-automatically, a protocol specification that has the given desired properties. The obtained protocol designs would then be “correct by construction”.

The first results in this direction were partial results. For instance, the protocol synthesis approach described by Zafiropulo et al. [Zafi 80] was a semi-automatic approach; the user would define the transitions that send a protocol message, while the automated system would derived all necessary transitions for receiving these messages in the different states of the destination process. The automated design tool would assure that the resulting protocol specification had no unspecified receptions, but all aspects related to the service provided were ignored.

A method developed by Philip Merlin and Bochmann [Boch 80d], later called “submodule construction”, considered that the desired communication service was given; but also here, there was no automatic protocol construction. In fact, it was assumed that the specification of one of the protocol entities would already be given. The construction method would then automatically find the most general behavior for the other protocol entity such that it would be compatible with the first one (without any unspecified receptions) and that both entities together would provide the given communication service (if that is possible). Later it was noticed that this method is useful for finding the behavior of a protocol converter, for finding a controller in discrete-event control applications, for embedded testing, and possibly for component re-use (for more details, see for instance [Boch 09a]).

The following two methods proposed in the 1980ies are probably more useful for protocol construction. At the first PSTV workshop in 1981, Gouda and Yu presented an interesting method

---

[9] The notion of state transitions in the context of the verification of general concurrent systems had for instance been described in [Gilb 72] and [Kell 76]. One issue is the choice of the size of transitions; larger transitions simplify the verification process. For this reason, Bochmann combined a sending transition of one component with the corresponding receiving transition of the other component (correct or erroneous) into a single “message transmission” transition in his early work [Boch 78i].

for designing communication protocols without deadlocks and with bounded communication channels (see journal version [Gouda 84]). The method starts with the specification of one protocol entity, and ignores the communication service. Its main contribution is to show how conflicting initiatives from the different protocol entities can be resolved.

Another protocol derivation approach was proposed by Reinhard Gotzhein and Bochmann in 1986 [Boch 86g]. Here it is assumed that the specification of the communication service is given, including two or more service access points. The method allows the derivation of the behavior of the protocol entities for each of the service access points, including the exchange of protocol messages between the different entities for coordinating the temporal order of service interactions at the different service access points. A reliable underlying message transmission service is assumed. Over the years, this method has been adapted to various specification paradigms, and it can be used for deriving component behaviors for distributed applications concerned with business processes, workflows and other systems where the dynamic behavior may be described by collaborations performed in some specific temporal order [Boch 08b].

3.3.6. Tools for protocol implementation

A verified protocol specification is clearly the starting point for protocol implementation. While finite state machine models are easy to implement in hardware and software, extended state models and the Formal Description Techniques (FDT) developed in the 1980ies require more complex implementation structures, especially related to the concurrency of the model specifications. Automated tools (compilers) for generating implementation code have been developed for these specification languages since the early 1980ies (see for instance [Blumer 82], [Boch 87] and [Vuong 88]). These developments later led to commercial modeling tools, for instance for the SDL language, which now has been integrated into the UML modeling framework.

3.3.7. Specification-based testing

In the context of protocol standardization, conformance testing is an important issue. The question is how to test a protocol implementation to determine whether it conforms to the protocol standard. It is clear, therefore, that the test cases to be applied to the protocol implementation should be based on the protocol specification that defines the standard. Since the early 1980ies, much research was aimed at test suite development, that is, developing methods for finding a set of test cases, based on a given protocol specification, that would find most faults that may exist in any implementation. The first publication on this topic was probably the paper by Sarikaya [Sari 82] adapting the so-called W-method [Chow 78], which was originally developed for testing software components, to the protocol testing based on FSM models. In the subsequent journal paper [Sari 84], he also discussed the problems related to the synchronization between different interfaces, a problem specific to protocol testing.

Since then a large body or research results have been established by researchers around the world [10]. On the one hand, tests for specific faults were considered, such as output and transition faults for state machines; on the other hand, test development methods were adapted to extended state machine models and other specification languages, such as Input/Ouput Automata and process algebras. While most methods provide a fixed set of test cases, other methods consider adaptive testing where the next test input depends on previous interactions.

In some cases, test inputs were randomly selected. For instance in IBM, a precisely defined applications programming interface for accessing the LU 6.2 protocols in SNA was developed. Known as Common Programming Interface Communications (CPI-C) [IBM 1992], it was available in a range of computer programming languages and was subsequently adopted by X/Open. An important feature of the interface was that the possible sequences of commands or function calls are defined in terms of a state machine. This made it possible to generate a random

[10] An invited paper at the 1994 International Symposium on Software Testing and Analysis, gave a review of these methods and discussed their relevance to software testing in general [Boch 94].

test driver, that exercised the interface [West 95]. This could be used for both simple, functional testing of the underlying software, or for stress testing by initiating many concurrent CPI-C programs. The driver was used to test a wide range of different CPI-C implementations and was remarkably successful.

An interesting by-product of testing one PC-based implementation was that a bug was found in the underlying TCP/IP code that was common to all 16 bit implementations of Berkeley UNIX.

3.4. Conferences on protocol specification, verification and testing

The first international conference concentrating on protocol specification, verification and testing was organized by Dave Rayner 1981 in Teddington (UK) under the title "Protocol Testing – Towards Proof ?". Sponsored by Working Group 6 of IFIP, this conference started the series of conferences under the name "Protocol Specification, Testing and Verification" (PSTV). This became the principal conference for the discussion of formal description techniques, and methods and tools for protocol verification, implementation and testing. In 1988, when the ISO FDTs had been standardized, Ken Turner started a second, parallel conference series called Formal Description Techniques (FORTE) which concentrated more on the FDTs, their tools and practical applications. Later in 1996, when the general interest in these FDTs had decreased, these two conference series were combined into a single conference under the name "Formal Techniques for Networked and Distributed Systems" (FORTE).

In 1988, also a conference series on protocol conformance testing was started by Son Vuong in Vancouver (Canada) under sponsorship from IFIP. Initially called International Workshop on Protocol Test Systems (IWPTS), it was later called International Conference on Testing Communicating Systems (TestCom) and became more recently a joint conference with the International Workshop on Formal Approaches to Testing of Software (FATES). Since that time, most of the research results on protocol testing have been presented at this specialized conference.

The bibliographical information and the tables of content for most of these conferences can be found on the Internet [conferences], and some highlights from the early PSTV conferences are given in the Annex.

4. Standardization of methods for protocol engineering

4.1. Standardization of formal description techniques (FDT) for communication protocols and services

As noted above, the people involved in the standardization of OSI protocols and services were looking for description techniques that could be used to defined protocols precisely and that would be associated with tools that would be helpful for verification, implementation and conformance testing. In 1980 an "FDT Group" was formed within ISO under the leadership of John Day. Initially, the group concentrated on writing guidelines about how to informally describe communication protocols and services, which were used by the various OSI subgroups for writing protocol and service descriptions. Then the group attacked the problem of defining formal techniques for which tool support would be possible to develop. The experts participating in the group had varying interests and therefore various approaches to protocol specification were proposed. This led finally to the formation of three subgroups A, B and C with subgroups B and C developing different techniques. Subgroup B developed an extended FSM formalism, later called Estelle, and subgroup C developed a formalism based on rendezvous communication (inspired by CCS [Miln 80]), later called LOTOS. The early stage of development of these languages is described in a paper by Vissers et al. [Vissers 83]. During the development of these FDTs, the OSI Transport protocol Class 2 was often used as an example to show the effectiveness of the FDT (e.g. [Boch 82n]). Later, in order to provide some comparison between the different FDTs,

Bochmann published example specifications of this protocol in the three FDTs (Estelle, LOTOS and SDL) [Boch 90a].

At this time, the ITU had already defined the System Description Language (SDL) which had been developed over several 4-year study periods. This was in fact an FDT based on the extended FSM model, and therefore the question came up whether Estelle and SDL could be developed jointly into a single language. This gave rise to the first joint working meeting between ISO and ITU experts, which was held 1984 in Ottawa and chaired by Bochmann. (Note that up to that time, the development of OSI protocols had been done by ISO and ITU in separate meetings.) It was decided that the FDT experts from OSI and ITU should get together and draft a proposal for a joint FDT as a candidate to be discussed afterwards by the respective parent organizations. Several people worked on this candidate FDT, in particular Roland Groz and Bochmann. However, the resulting document, called X.250 within the ITU, was not accepted, and the development of these two languages continued independently.

During that time, the OSI experts were waiting for an FDT that they could use for describing the new OSI protocols. However, the development of the FDT's was not complete, and there was the very sensitive question of which FDT should be used. The subgroup working on guidelines for conformance testing, led by Dave Rayner, expressed a strong need for an FDT, since they were looking for a language to define conformance test cases. Since the testing experts did not want to select one of the proposed FDTs, they finally decided to develop their own language, called TTCN (acronym of Tree-Table-Combined-Notation, not related to any FDT), which has since then been used in the area of protocol conformance testing.

Estelle and LOTOS became ISO standards in 1986, but were never much used. SDL was further developed within ITU and was used for the description of ITU protocol standards. An overview of the state of the art of protocol specifications using informal and formal techniques at that time is given in [Boch 90g]. Looking back at this time of FDT development, one has to conclude that these description methods and the related tools were not much used for the purpose they were developed (except for SDL). One can give different reasons for explaining this situation:

1. The community was not ready to use formal modeling approaches to define requirements for protocol implementations.
2. The implicit competition between the three different languages made it difficult for practitioners to make the choice.
3. The OSI protocols, for which the FDTs were originally intended, did not have much success themselves.

A further reason was that the specification techniques in use for OSI protocols were already quite good and extremely useful. For example, the OSI Session Layer was specified in terms of tabular finite state machines. In 1985 Colin West found it quite easy to transform these into an executable model that he validated using the reachability analysis techniques that he had earlier used for SNA [West 86]. A substantial number of errors were found.

When the OSI Transaction Processing standard was developed in the early 1990s, the 680 page document contained similar tabular state machines, as well as 360 pages of Lotos and Estelle definitions of the protocol.

The state machine definitions represented the working documents of the committee and were definitive. The Estelle and LOTOS representations were derived from those definitions by FDT experts after the fact. Colin West also validated this protocol during its development using a random walk approach. The transformation of the tabular state machines into an executable model was largely automated.

It proved possible to attend a meeting of the committee developing the protocol, take the ASCII document which was the work product of the committee at the end of the day, produce an executable model, validate it overnight on a laptop computer and present the committee the following morning with a series of message exchange sequences that lead to errors in the protocol. Using this technique, approximately 200 errors were found in early versions of the OSI Transaction Processing standard [West 93]. It was not possible to completely execute the Estelle and Lotus models during the development of the standard and they contributed little to its overall quality.

4.2. Standardization of conformance testing

The research community in the UK began developing work on protocol conformance testing at a workshop at NPL in January 1980, where Dave Rayner presented a paper on "Protocol Independent Testing Philosophy". This was soon followed by a paper at an IEEE Symposium in May 1980 [Rayn 80]. This early work was applied to a set of network independent protocols known as the "Coloured Book" protocols, which were a forerunner of OSI protocols. The main participants included NPL and the National Computing Centre[11]. This work evolved quickly over the next few years, as can be seen in [Rayn 83][12]. This work was complemented by other key contributors to the early PSTV and IWPTS conferences. In particular, there was soon international collaboration on protocol testing involving the UK, USA, France and Germany.

Thus, when, in 1982, the ISO OSI community raised the question of how to ensure that OSI protocol implementations really conformed to the specifications, it was not surprising that there was an immediate response from this research community. In particular, there was a strong response from the UK's BSI committee on OSI which resulted in Dave Rayner taking the lead on this issue in ISO from 1983[13], when work began to create a standard methodology and framework for OSI conformance testing.

In the early days of this standardization, there was strong opposition from major suppliers, particularly IBM and DEC, to the creation of such a standard. They wanted acceptance of suppliers' declarations of conformity and they did not want to see the rise of independent test houses or conformance testing imposed on them by telecom suppliers. However, it was not long before attitudes changed and the group led by Dave Rayner was getting constructive contributions from all categories of interested parties: large suppliers (particularly IBM and DEC), telecom companies (then called the PTTs), software houses, test houses, consultants, and academic researchers.

After several years of discussion and refinement of ideas, the result was the development of a comprehensive conformance testing methodology and framework, standardized initially as the 5-part ISO/IEC 9646-1 to 5, published in 1991 and 1992. These parts provided

1. General concepts – giving an overview including a comprehensive terminology
2. Abstract test suite specification – giving a description of how to standardize tests, independent of the test notation
3. TTCN (Tree and Tabular Combined Notation) – a detailed test definition language
4. Test realization – giving requirements on test systems that would run the standardized tests
5. Requirements on test laboratories and clients for the conformance assessment process

This multi-part standard applied to testing individual OSI protocols in any OSI layer except the Physical Layer. However, by the time the standards were published, the OSI standardization community had moved on to develop the concept of protocol profiles, collecting together requirements across multiple layers. Furthermore, people were beginning to develop tools to support TTCN and there were many ideas for extension of TTCN to make it more comprehensive. Thus, there was a demand for a second edition of the multi-part standard to address these new issues. The result was that a 7-part second edition was eventually published between 1994 and 1997. The two additional parts provided

6. Protocol profile test specification
7. Implementation conformance statements – standardizing how suppliers should make their declarations of conformity

Furthermore, these standards were also published by ITU-T as X.290-6 and were endorsed by the European Telecommunications Standards Institute (ETSI) and the European Workshop on Open Systems (EWOS), which built upon these standards. Within ISO itself, several standards were produced for test suites for particular protocols, each test suite being constructed in

[11] Others were the South West Regional Computing Centre and the Computer Aided Design Centre.

[12] This was a revision of the paper first presented by Dave Rayner at the second PSTV workshop held in May 1982

[13] Dave Rayner was ISO rapporteur for OSI conformance testing from 1983 to 1997.

accordance with ISO/IEC 9646. It is fair to say that this multi-part standard represents the most comprehensive testing methodology ever standardized. What is more, the usefulness of this methodology did not die with OSI, but rather lives on. For example, conformance testing of the protocols involved in 3G mobile phone services was conducted using this methodology.

There were, of course, many people who contributed to these developments but one who should be specially acknowledged was Anthony Wiles (editor of TTCN and a major presence in ETSI).

Although there was always dialogue between those involved in developing the testing methodology and those working on formal description techniques, the testing community essentially sought a pragmatic solution to the problem of conformance testing for protocols whose definitions were informal. Many in the testing community argued that formal specifications would always arrive on the scene too late, be unintelligible for many involved in defining and implementing the protocols, and as a consequence would never be accepted as definitive. On reflection, the same criticisms could be leveled at test specifications. The test notation, TTCN, was designed to be easier to understand than formal techniques, but as it evolved to be ever more expressive and implementable in test tools, it actually became just as unintelligible to most people as any of the formal techniques. Indeed, it came close to being another formal technique. It is interesting to note that probably the most successful TTCN tool came from Telelogic along with an SDL tool, and indeed there were moves in ITU-T and ETSI to bring TTCN and SDL closer together, allowing protocols to be defined in SDL with closely related test specifications written in TTCN.

5. Conclusions

This historical retrospective indicates that the basic concepts related to the specification of communication protocols and services, the verification of protocols and their implementation and testing originally developed in the 1970ies and ‘80ies are still valid today. Many things have remained the same, although over the years, the more detailed aspects have evolved and given rise to new technological developments. The IFIP conferences in the field of protocol engineering, namely PSTV, FORTE and TestCom, have played an important role in maintaining the tradition in this field and providing a forum for the discussions of new developments in these areas.

During the first half of the 1980ies, there was some form of hype on Formal Description Techniques (FDT). Industry had high hopes that formal specifications of protocol standards, especially in the context of OSI, would lead to less ambiguous definitions of the protocols, and would lead, through the use of automated tools, to simpler methods for verification, implementation and conformance testing. However, these hopes did not materialize, and in addition, the push for OSI protocols slowed down during the second half of the decade. This was a big disappointment for the FDT community.

We note, however, that the results of the work on protocol engineering, formal description techniques and the related tools for protocol verification, implementation and testing have had an impact, have been carried over to the current state of the art, and have been used for many practical applications in different fields, as explained in the following paragraphs.

Layered protocol architecture: The conceptual layered architecture for communication protocols, as developed during the 1970ies, and standardized as the OSI Reference Model in the early ‘80ies, are nowadays part of each good text book on computer communications.

Model checking: In a sense, the reachability analysis tool of Colin West was the first model checker – able to check a model of several protocol entities for general types of flaws, such as deadlocks and unspecified receptions. The invention of “model checking” by Clark, Emmerson and Sifakis in 1981 provided, in addition, for checking specific properties specified in temporal logic. During a visit of Bochmann at Havard University in 1982, Edmund Clark was excited about the fact that his model checking algorithm could not only check properties of a given state machine model, but also properties of protocols and provided communication services – by applying his algorithm to the global state model obtained by reachability analysis. The SPIN tool developed by Gerald Holzmann since the early 1980ies is an example of a successful model checking tool that started out as a tool for reachability analysis and, over the years, incorporated

many advances in the field, including checking temporal logic properties [Holz 91]. This tool has been used for many practical applications of hardware and software verification. The community of people using the tool has been growing over the years and meets at yearly workshops [SPIN 2009].

UML tools: As mentioned above, the ITU had developed over the years since the late '70ies an standard FDT, called System Description Language (SDL), which was used for describing many communication protocol standards and other industrial systems, and its commercial tools developed in the 1990ies have been used for the development of commercial protocol implementations, for instance in the wireless telephony sector. The most successful tool was produced by the Swedish company Telelogic (recently bought by IBM); it includes code generation for model simulation and implementation, as well as advanced reachability analysis functions. Recently, SDL has been integrated into UML-2 as a profile, and the tools are adapted to this new context.

Model-driven development: This term has recently become a fashionable concept. Protocol engineering has used this approach since its beginning. The reason for this is the fact that a protocol specification should be relatively abstract (in order to allow for different implementations and APIs) and precise (in order to guarantee compatibility of different implementations). The protocol specification is therefore an abstract model of the final implementation, and protocol verification is done at the model level. In fact, the FDTs SDL and Estelle, as well as Harel's State Charts or 1987 [Hare 87] are languages based on the principle of extended finite state machines from the 1970ies, and they can be considered to be ancestors of the State Diagrams notation now now part of UML.

Conformance testing methodology: The methodology defined so comprehensively in ISO/IEC 9646-1 to 7 and ITU-T X.290 to 296 remains an important reference work for anyone needing to apply conformance testing to telecommunications today. This was seen when it was used as the approach to test 3G mobile phone protocols.

References

[Ball 80a] A. Ball, G. v. Bochmann and J. Gecsei, Videotex Networks, IEEE Computer, Vol. 13, No. 12 (December 1980), pp. 8-14.

[Bart 69] K. A. Bartlett, R. A. Scantlebury and P. T. Wilkinson, A note on reliable full-duplex transmission over half-duplex links, ACM Communications, Vol.12, No.5, May 1969, pp.260-265.

[Blumer 82] T. P. Blumer, R. L. Tenney, A formal specification technique and implementation method for protocols, Computer Networks 6(3): 201-217 (1982)

[Boch 08b] G. v. Bochmann, Deriving component designs from global requirements, Proc. Intern. Workshop on Model Based Architecting and Construction of Embedded Systems (ACES), Toulouse, Sept. 2008.

[Boch 09a] G.v. Bochmann, Using first-order logic to reason about submodule construction, submitted to FORTE 2009.

[Boch 75c] G. v. Bochmann, Logical verification and implementation of protocols, Proc. 4th Data Communication Symposium (ACM-IEEE), pp. 7-15 to 7-20, Oct. 1975, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 75d] G. v. Bochmann, Communication protocols and error recovery procedures, Proceedings ACM Symposium on Interprocess Communication, SIGOPS Review, 9, No.3, 45-50 (1975).

[Boch 77c] G. v. Bochmann and J. Gecsei, A unified method for the specification and verification of protocols, Proc. IFIP Congress 1977, pp. 229-234.

[Boch 77j] G. v. Bochmann and R. J. Chung, A Formalized Specification of HDLC Classes of Procedures, (invited paper) National Telecomm. Conf., Dec. 1977, proc. pp. 03A..2-1 to 2-11. Reprinted in Advances in Computer Comm. and Networking, ed. W. Chu, Artech, 79.

[Boch 78e] G. v. Bochmann, Combining assertions and states for the validation of process commmunication, IFIP Working Conference on Constructing Quality Software, Proc. ed. by P.G. Hibbard and S.A. Shuman, North Holland, 1978. Translated into russian.

[Boch 78i] G. v. Bochmann, Finite State Description of Communication Protocols, Computer Networks, Vol. 2 (1978), pp. 361-372.

[Boch 80a] G. v. Bochmann and C. A. Sunshine, Formal methods in communication protocol design, (invited paper) IEEE Tr. COM-28, No. 4 (April 1980), pp. 624-631, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 80d] G. v. Bochmann and P. M. Merlin, On the construction of communication protocols, ICCC, 1980, pp.371-378, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 80e] G. v. Bochmann, A General Transition Model for Protocols and Communication Services, IEEE Trans. Comm., COM-28, 4 (April 1980), pp. 643-650, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 82e] G. v. Bochmann, J. Gecsei and E. Lin, Keyword access in Telidon: An experiment, Proc. Videotex 82, New York, June 1982.

[Boch 82n] G. v. Bochmann, E. Cerny, M. Gagne, C. Jard, A. Leveille, C. Lacaille, M. Maksud, K. S. Raghunathan and B. Sarikaya, Experience with Formal Specifications Using an Extended State Transition Model, IEEE Trans. COM-30, No.12 (Dec. l982), pp. 2506-2513.

[Boch 83a] G. v. Bochmann and M. Raynal, Structured specification of communicating systems, IEEE Trans. Computers C-32, 2(Febr. 1983), pp. 120-133.

[Boch 86g] G. v. Bochmann and R. Gotzhein, Deriving protocol specifications from service specifications, Proc. ACM SIGCOMM Symposium, 1986, pp. 148-156.

[Boch 87h] G. v. Bochmann, G. Gerber and J.-M. Serre, Semiautomatic implementation of communication protocols, IEEE Tr. on SE, Vol. SE-13, No. 9, September 1987, pp. 989-1000, (reprinted in "Automatic Implementation and Conformance Testing of OSI Protocols", IEEE, edited by D.P.Sidhu, 1989).

[Boch 90a] G. v. Bochmann, Specifications of a simplified Transport protocol using different formal description techniques, Computer Networks and ISDN Systems, Vol. 18, no.5, June 1990, pp. 335-377.

[Boch 90b] G. v. Bochmann and P. Mondain-Monval, Design principles for communication gateways, IEEE Tr. on Selected Areas in Communications, Vol.8, 1 (Jan. 1990), pp. 12-21; russian translation: Express Information (overview of western publications), Information Transfer, 1991.

[Boch 90i] G. v. Bochmann, Deriving protocol converters for communication gateways, IEEE Trans. on Comm., Vol. 38, 9 (Sept. 1990), pp. 1298-1300.

[Boch 90t] G. v. Bochmann, D. Ouimet and G. Neufeld, ASN.1 and Estelle implementation support tools, Proc. Third Int. Conf. on Formal Description Techniques, IFIP (FORTE'90), Nov. 1990, Madrid, pp.531-534.

[Boch 93c] G. v. Bochmann, Protocol Engineering, contribution to Concise Encyclopedia of Software Engineering, Derrick Morris and Boris Tamm eds., Pergamon Press, 1993, pp. 266-271.

[Boch 94d] G. v. Bochmann and A. Petrenko, Protocol testing: Review of methods and relevance for software testing (invited paper), ACM International Symposium on Software Testing and Analysis (ISSTA'94), Seattle, USA, 1994, pp 109-124.

[Cerf 93] V. Cerf, How the Internet came to be: The birth of the ARPANET, http://www.netvalley.com/archives/mirrors/cerf-how-inet.txt

[Chow 78] T. S. Chow, Testing software design modelled by finite state machines, IEEE Trans. on Soft. Eng., Vol.4, no.3, 1978.

[Conferences] Computer Science Conferences and Workshops, see http://www.informatik.uni-trier.de/~ley/db/conf/index-f.html

[Danthine 1978] Computer Network Protocols Symposium, 13-15 Febr. 1978, organized by André Danthine, Université de Liège, Belgium.

[Gilb 72] P. Gilbert and Q.J. Chandler, Interference between communicating parallel processes, Comm. ACM 15, pp. 427-437 (1972).

[Goud 84] M. G. Gouda and Y.-T. Yu, Synthesis of communicating Finite State Machines with guaranteed progress, IEEE Trans. on Comm., vol. Com-32, No. 7, July 1984, pp. 779-788.

[Gray 83] J. P. Gray, P. Haman, P. J. Hansen, M. A. Lerner and M. Pozef.sky, Advanced program-to-program communication in SNA. IBM Systems J. 22 (4) (1983) 298-318.

[Green 86] P. E. Green, Protocol conversion, IEEE Truns. Commun.. vol. COM-34. pp. 257-268. Mar. 1986.

[Green 87] P. E. Green, R. J. Chappuis, J. D. Fischer, P. S. Frosch and C. E. Wood, A perspective on advanced peer-to-peer networking, IBM Systems J. 26 (4) (1987) 414-428.

[Hare 87] D. Harel, Statecharts: A visual formalism for complext systems, Science of Computer Programming, Vol. 8, 3 (June 1987), pp. 231-274.

[Hauben] Ronda Hauben, Communicating across the boundaries of dissimilar networks, http://www.columbia.edu/~rh120/other/misc/finland416.txt

[Hoar 78] C. A. R. Hoare, Communicating sequential processes, Comm. ACM 21, 8 (Aug. 1978), pp. 666-677.

[Holz 91] G. J. Holzmann, Design and Validation of Computer Protocols, Prentice Hall, 1991.

[IBM 92] IBM Systems Application Architecture Common Programming Interface Communications Reference, IBM SC26-4399-0.5 (June 1992). SC26-4399-05 (June 1992).

[Kell 76] R. M. Keller, Formal verification of parallel programs, Comm. ACM 19, 7 (July 1976), pp. 371-384.

[Lynch 68] LYNCH, W. C. Reliable full-duplex file transmission over halfduplex telephone lines, Comm. ACM 11, 6 (June 1968), 407-410.

[Miln 80] R. Milner, A calculus of communicating systems, Lecture Notes in Computer Science, No. 92, Springer Verlag, 1980.

[Nelson 65] T.H. Nelson, Complex information processing: a file structure for the complex, the changing and the indeterminate, ACM/CSC-ER Proceedings of the 1965 20th national conference.

[Neuf 90b] G. Neufeld and Y. Yang, The design and implementation of an ASN..1-C compiler, IEEE Transactions on Software Engineering, October 1990, Vol 16 Number 10, pages 1209-1220.

[Piat 75] T. F. Piatkowski, Finite-state architecture, Proc. 7th Annual Southeastern. Symp. System Theory, March 1975, Auburn University, Auburn, AL, and Tuskegee Institute, Tuskegee, AL, IEEE Cat. No. 75 CH0968-8C.

[Pnueli 77] A. Pnueli. The Temporal Logic of Programs. Proc. 18th IEEE Symposium Foundations of Computer Science (FOCS 1977), pages 46-57, 1977.

[Pouzin 73] L. Pouzin, Presentation and Major Design Aspects of the Cyclades Computer Network, Proc. Third Data Communications Symposium, St. Petersburg, Florida

[Rayn 80] D. Rayner and K.A. Bartlett, The Certification of Data Communication Protocols, Proc. IEEE Symposium, Trends and Applications: 1980, Computer Network Protocols, held at the National Bureau of Standards on 29 May 1980, pp 12-17, 1980.

[Rayn 83] D. Rayner, A System for Testing Protocol Implementations, Computer Networks 6, pp 383-395, 1983.

[Rudin 78] H. Rudin, C.H. West, P. Zafiropulo, Automated protocol validation: One chain of development, Computer Networks, Vol. 2, No 4/5, pp. 373-380, 1978.

[Rudin 82] H. Rudin, C. H. West, A Validation Technique for Tightly Coupled Protocols IEEE Transactions on Computers, C-31, 7, July 1982, 630 - 636.

[Sari 82a] B. Sarikaya and G. v. Bochmann, Some experience with test sequence generation for protocols, Proc. 2-nd Int. Workshop on Protocol Specification, Testing and Verification, North Holland, 1982, pp. 555-567.

[Sari 84a] B. Sarikaya and G. v. Bochmann, Synchronization and specification issues in protocol testing, IEEE Trans. on Comm., COM-32, No.4 (April 1984), pp. 389-395.

[Schultz 75] G. D. Schultz, D. B. Rose, C. H. West and J. P. Gray, Executable representation and validation of SNA, in; P. E. Green Jr., ed., Computer Network Architectures and Protocols (Plenum. New York, NY. 1982), 671-705.

[SPIN 2009] http://ti.arc.nasa.gov/event/spin09/

[Sten 76] N. V. Stenning, A data transfer protocol, Computer Networks 1(1976), pp. 99-110.

[Suns 76] C.A. Sunshine, Survey of communication protocol verification techniques, Proc. Symp. Computer Networks, NBS, Maryland, November 1976, pp. 24--26. (IEEE)

[Suns 78] C.A. Sunshine, Survey of protocol definition and verification techniques, ACM SIGCOMM Computer Communication Review, Vol. 8, 3 (July 1978), pp. 35 – 41. Also included in the proceedings of the Liège conference [Dant 78].

[Sunshine 81] C. Sunshine (ed.), Communication Protocol Modeling, Artech House, 1981.

[Tomp 81b] F. W. Tompa, J. Gecsei and G. v. Bochmann, Data structuring facilities for interactive videotex systems, IEEE Computer, Vol.14, No.8, August 1981, pp.72-81.

[West 78] C. H. West, An automated technique of communications protocol validation, IEEE Trans. COM-26 (1978), pp. 1271-1275.

[West 78b] C. H. West and P. Zafiropulo, Automated validation of a communications protocol: the CCITT X.21 recommendation, IBM J.Res. Develop. 22 (1978), pp. 60-71.

[West 78c] C. West, General technique for communications protocol validation, IBM Journal for Reseach and Development, Vol. 22, No. 4, 1978, pp. 393-404.

[West 86] C. H. West, A validation of the OSI session layer protocol, Computer Networks ISDN Systems 11 (1986) 173-182.

[West 92] C. H. West, Protocol validation - principles and applications, Computer Networks and ISDN Systems, 24 (1992) 219-242.

[West 93] C. H. West, The challenges facing formal definition techniques, FORTE'93, Boston, MA, Oct. 1993.

[West 95] C. H. West, A. Tosi, Experiences with a random test driver, Computer Networks and ISDN Systems 27 (1995) 1163-1174.

[Viss 83a] C. A. Vissers, G. v. Bochmann and R. L. Tenney, Formal description techniques, Proceedings of the IEEE, vol. 71, 12, pp. 1356-1364, Dec. 1983; translated into russian.

[Viss 85] C. Vissers and L. Logrippo, The importance of the concept of service in the design of data communications protocols, IFIP Workshop on Protocol Specification, Verification and Testing (Toulouse, 1985), North Holland.

[Vuon 88] S. T. Vuong, A. C. Lau and R. I. Chan, Semiautomatic Implementation of Protocol using an Estelle-C Compiler, IEEE Transaction on Software Engineering, vol. 13, no. 3, pp. 384-393, March 1988.

[Zafi 78]] P. Zafiropulo. Protocol Validalion by Duologue Matrix analysis, IEEE Trans. Comm. 26 (8) (1978) 1187-1194.

[Zafi 80] P. Zafiropulo, C. H. West, H. Rudin and D. D. Cowan, Towards analyzing and synthesizing protocols, IEEE Tr. Comm. COM-28, 4 (April 1980), pp. 651-660.

Annex: Some highlights from early PSTV conferences

In the following, certain developments in the state of the art of protocol engineering will be pointed out in relation with presentations at the early PSTV conferences sponsered by IFIP. The selection of these points is necessarily biased, and we hope that we do not offend anybody by not mentioning his/her contributions. We note that the bibliographical information and the tables of content for most of PSTV, FORTE and TestCom conferences can be found on the Internet [conferences].

- The first PSTV conference had some emphasis on testing methods because the organizer, Dave Rayner, was involved in conformance testing. There were many papers on conformance testing and testing tools, but also many papers on protocol specification and verification. There was for instance an early paper on code generation from EFSM models by Blumer and Tenney (see journal version [Blumer 82]), and the paper by Gouda and Yu on protocol construction mentioned earlier.
- The second PSTV was organized by Carl Sunshine in Idyllwild (near Los Angeles, USA). Here we noticed several papers proposing the use of temporal logic for describing protocol properties and verification. Temporal logic was a new topic in computer science at that time, introduced by a paper by Pnueli [Pnueli 77] in 1977. Some experience with a dialect of the Subgroup-B FDT was presented by my research group, describing the development of the specification of the Transport protocol and its use for implementation and conformance testing (see journal version [Boch 82n]). Gerald Holzmann presented a paper comparing three algebraic protocol validation methods. Subsequently, he developed the well-known Spin tool [SPIN] using a standard "reachability" approach. One of his important innovations for protocol verification, the so-called bit-state representation, was presented at the PSTV in 1987. It was also at the second PSTV, that Behcet Sarikaya presented a paper on the derivation of conformance test cases from a formal FSM model of the protocol, based on the so-called W-method [Chow 78] which was originally developed for testing software components. This was probably the first paper about using a formal protocol specification for the derivation of test cases. In the subsequent journal paper [Sari 84], he also discussed the problems related to the synchronization between different interfaces, a problem specific to protocol testing.
- The third PSTV was organized by Harry Rudin and Colin West in Rüschlikon (near Zurich). As in previous years, various methods for protocol specification and verification were presented. This year, a specific session on Petri nets was included with contributions from France, Finland, Germany and Australia. There was also a session on "Integrated Systems" in which many tools for protocol implementation and/or verification were presented.
- The fourth PSTV was organized by Yechiam Yemini at Sky Top, PA (USA). During this year, two papers suggest the use of the general description techniques CCS [Milner 1980] and CSP [Hoare 1978] that had been defined a few years earlier. These techniques, based on rendezvous communication, influenced strongly the development of the FDT LOTOS. There was also a paper by Brinksma and Karjoth on a specification of the Transport protocol in LOTOS. This year also had a session on performance modeling, a topic that has always been active at PSTV and FORTE, but never very strongly.
- The next two PSTV (1985 organized near Toulouse (France) by Michel Diaz, and 1986 organized by Bochmann and Sarikaya near Montreal (Canada)) are characterized by the presence of many papers presenting automated implementation and verification tools for the FDT Estelle (including the X.250 version) and other languages. A paper by Logrippo et al. also described the interpretive execution of LOTOS specifications.