Experience in validating protocol integration using Estelle

Jean-Marc Jézéquel

Experience in Validating Protocol Integration using Estelle

Jean-Marc JÉZÉQUEL

TRANSPAC
F-35512 CESSON SEVIGNE CEDEX, FRANCE

Abstract

This paper presents a ten months long experiment led at TRANSPAC to check the interest of a Formal Description Technique like Estelle for industrial purposes.

Through the *Intelligent Network* new service introduction, Estelle —and associated tools, e.g. VEDA and *ECHIDNA*— has been used for different purposes: validation of a brand new protocol and test of its implementation, modelisation of an already existing one, and validation of the integration of both protocols.

After a brief introduction to the TRANSPAC framework, we present an outline of these experiments. Then we draw some conclusion on the suitability of both Estelle and the tools used in this context, and on the economical balance of such an approach to deal with concrete problems.

1 TRANSPAC general framework

1.1 The first packet switching network in the world

TRANSPAC has been created in 1978 to commercialize, manage and expand the french public packet switching network. There are now 70000 users directly connected to TRANSPAC, using 210 Packet Switching Systems (PSS) located in 124 different sites and exchanging 2.2G bytes per month.

The basic service proposed is the Virtual Circuit. User can access to the TRANSPAC network with X25, X32 or X28 standardized protocols[5], from dedicated links, telephonic or telex networks, or ISDN B channels.

Higher level services are also proposed, as Atlas 400 —an X400 mailer with 2500 subscribers and 150K messages exchanged/month— EDI servers, private network supervisions services...

1.2 Structure of the network

The TRANSPAC network is made of first and second generation PSS (Mitra 125 and Sesa DPS-25) to connect users, and of Transit-PSS (Alcatel DPS-8300) to interconnect PSS using high throughput links ($\geq 256Kb/s$), see figure 1. Users can be connected directly (with synchronous or asynchronous links) or through User Public Concentrators or Enterprise Local Concentrators... In the near future, users will also be able to connect themselves using ISDN D channel permanent links.

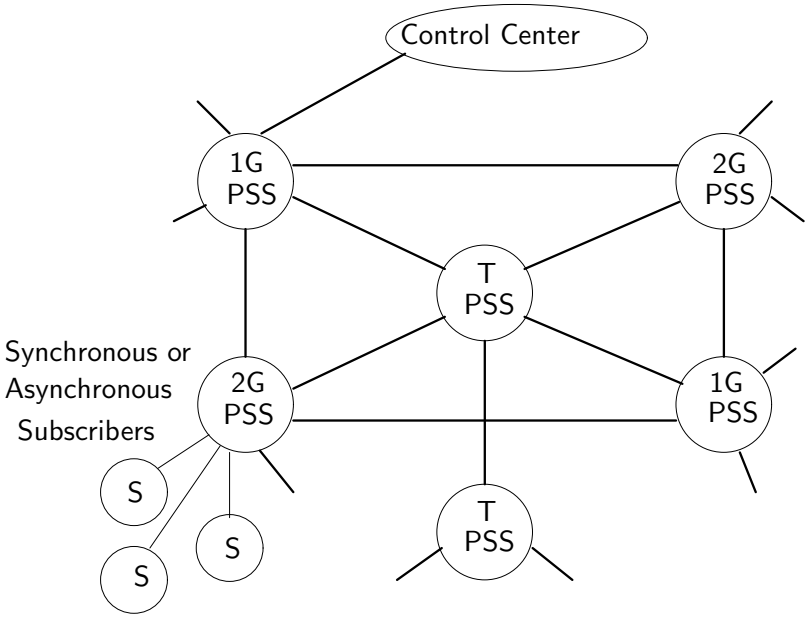


Figure 1: Structure of the TRANSPAC Network

Within the TRANSPAC network, inter-PSS communications are performed using the XCM protocol, which is an evolution of X25.

1.3 The Intelligent Network project

In order to fit users' needs, TRANSPAC has to offer new services above the basic network. Until now, each new service introduced in the network was costly because we needed to modify the software of each kind of PSS.

The Intelligent Network (IN) project aims at lowering these costs. Its principle is to remote process the signalling (e.g. Call-Request, Call-Confirmation and Clear packets) in a so-called *Packet Service Command Point* (P-SCP) specialized machine.

Access to this "intelligent" P-SCP is granted to *Packet Service Access Systems* (P-SAS), which are in a first step located in TRANSPAC T-PSS. At the request of the P-SCP, the P-SAS have to trap relevant transiting signalling packets, and send them to the P-SCP for processing (see figure 2). The P-SCP can also spontaneously request a P-SAS to send signalling packets.

The dialogue between P-SAS and P-SCP is made on a Virtual Circuit established by the P-SCP, and the P-SAS knows the P-SCP as a standard subscriber.

This architecture can be used to provide administration functions, like:

- Call Transfer Service
- *dynamic configurable multiple subscriber* service
- access supervision service

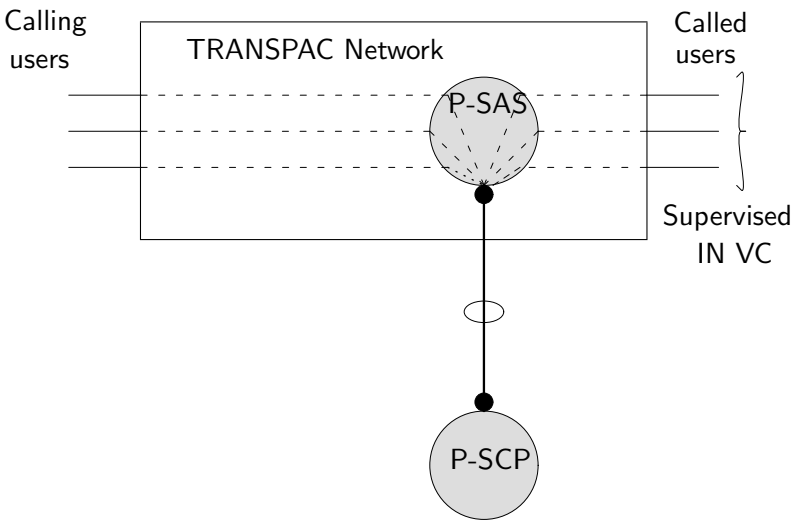


Figure 2: The Intelligent Network Architecture

- access control service
- client specific services

1.4 Context of the study

In order to introduce the IN service, TRANSPAC had to develop a new protocol between P-SAS and P-SCP. As the P-SAS should be implemented in the T-PSS DPS-8300 —which is a multi-processor machine with internal protocols— the rather complex interactions of this new IN protocol with the existing ones have to be studied carefully.

Having a growing interest for FDTs, TRANSPAC wanted to test the usefulness of those technics on this IN project.

TRANSPAC needed a concrete, commercially available set of tools to model and validate both new and already existing protocols.

After a rapid enquiry, TRANSPAC choose the Estelle language[2] and the VEDA environment[3] designed at CNET and sold by VERILOG SA, mainly because it was the only one environment —available in France— allowing some validation of real-size protocols.

Furthermore, through a contract with IRISA, TRANSPAC has been granted the right to use *ECHIDNA*[4] as a second Estelle compiler, featuring a windowed interactive visual debugger and a multi-processor kernel for experimentation purposes. For IRISA, the interest of this contract was to check how *ECHIDNA* was suited for industrial purposes, and how to fit them better.

2 Using Estelle to model and validate proprietary protocols

2.1 Specification and Validation of the IN protocol

This first part of the study was rather classical: given a brand new protocol, we had to make a formal specification and a validation.

The communication between the two protocol entities P-SCP and P-SAS is performed above an X25 VC, which can be *Reset*ed or broken at any time by anyone of the P-SCP, P-SAS or the TRANSPAC network. Each time this communication is broken, the P-SCP tries to establish it again. However, the effects of such a failure on currently supervised VC should be as low as possible, so a special mechanism has been designed to deal with message losses.

In a first step, the VC has been modeled as a bidirectional, reliable, unbounded FIFO channel.

Starting with an informal description in French, we made a formal model with two Communicating Finite State Automata (CFSA), one for a supervised VC at the P-SAS, the other one for its image at the P-SCP.

We did a coverage graph analysis with an experimental tool (lent by IRISA) which led to some design error detection and showed the too much complexity of the protocol.

After various iterations and some simplification, we got an interesting validation, providing informations on the maximum size of channels and the reuse of supervised VC identifiers.

Then we did a full model using Estelle (about 700 lines of source code). We took into account the network sending *Reset* and *Clear*, and the resynchronization procedure upon reconnection.

However, modeling the interfacing of this protocol with X25 was rather difficult, because full modeling of X25 was not the point, but any simplified model led to interface problems.

We did intensive simulations using VEDA and *ECHIDNA*, but none of them led to new error detection. During this phase, we appreciate the VEDA feature called *observers*[1], which made it possible to check easily for the consistency between the supervised VC view at the P-SAS and at the P-SCP.

Using the *experimentation on real distributed systems* features offered by *ECHIDNA*, we made a very demonstrative exhibition of this IN protocol using one Sun workstation playing the role of the P-SAS and an other one playing the P-SCP role.

Another interest of this Estelle formal specification is its usefulness as a reference for implementation testing. In this context, we built —at a relative low cost: 2 months.man— an Estelle testing environment for the IN protocols using *ECHIDNA* above an X25 package. It allows the TRANSPAC test team to specify test cases in Estelle, and then to compile and run them in front of an actual protocol implementation.

[1] Kind of almighty deamons which can spy Estelle modules, see [1]

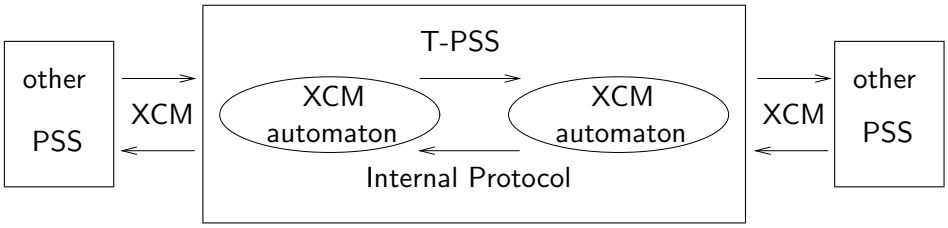


Figure 3: The T-PSS abstract specification for one Virtual Circuit

2.2 Modelisation of the T-PSS internal protocols

In order to study the integration of the new IN protocol within the T-PSS, we had to model first the existing T-PSS internal protocols. The external abstract specification of the T-PSS for a Virtual Circuit can be thought as a set of two XCM automata, each one communicating with both the other one and the external world, thus performing the switching on the VC (see figure 3).

Therefore the first step was to build a model of this specification using Estelle, in order to have an operational reference model to be compared with the implementation. The only little difficulty here was to take into account erroneous cases for which there were only informal specifications.

We got a 300 lines —60 transitions— Estelle specification, from which we could automatically produce a testing simulated environment for the implementation modelisation (see below).

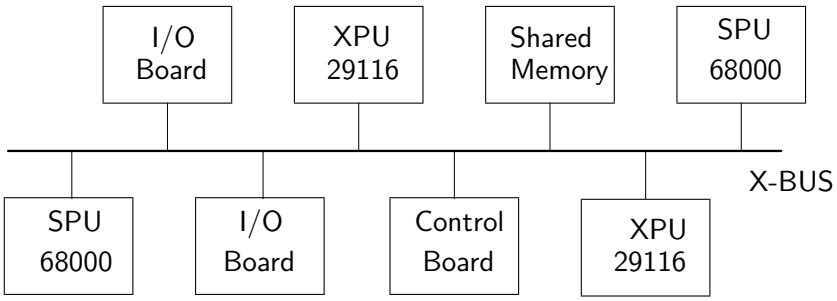
Switching and signalling functions are implemented in the DPS-8300 T-PSS on a set of boards linked through an high speed bus, and communicating with both message exchanges and shared memory, see figure 4. Micro-coded AMD 29116 boards (XPU) are used to perform fast switching of simple packets (i.e. Data packets), meanwhile signalling processing and routing is subcontracted to MC 68000 boards (SPU).

For a given VC, signalling related internal interactions between components are described in figure 5.

The typical processing of an incoming signalling packet is described in figure 6, but in reality, as there were many performance related improvements during the development and maintenance courses of the project, the real protocols are quite different from the initial specifications.

What we had to do here can really be called *reverse engineering*, because we had to build an high level description from assembly listing.

As each processing of an internal interaction could be considered *atomic* —and therefore modeled with an Estelle transition— the main problem was to model shared memory communications. We finally choose to gather in a single Estelle module both the common memory block and the transitions of the two processors accessing it. With that choice, we loose some parallelism between those two processors, but anyway, there is a price to be paid for every



SPU: Signalisation processing unit, XPU: Switching processing unit

Figure 4: The DPS-8300 T-PSS architecture

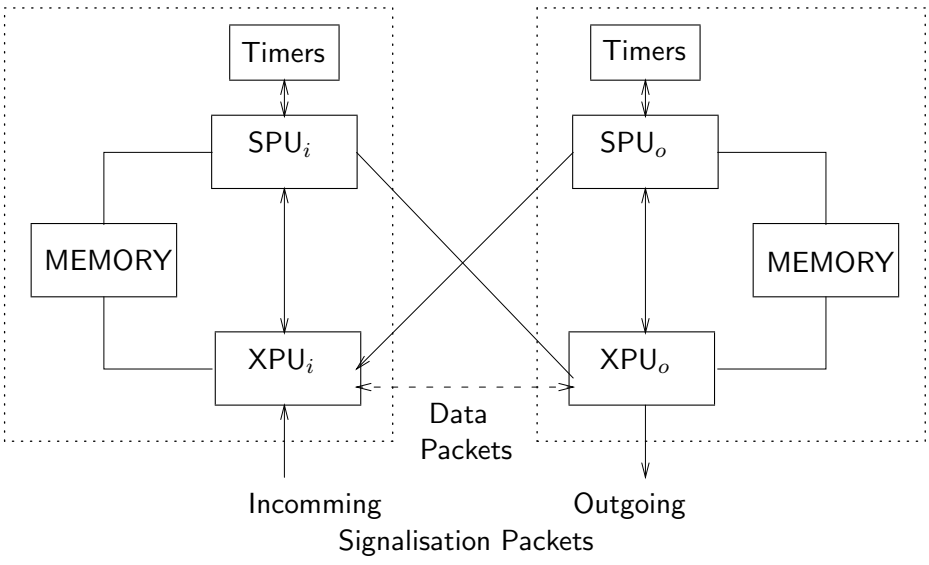


Figure 5: Relations between components during signalling processing

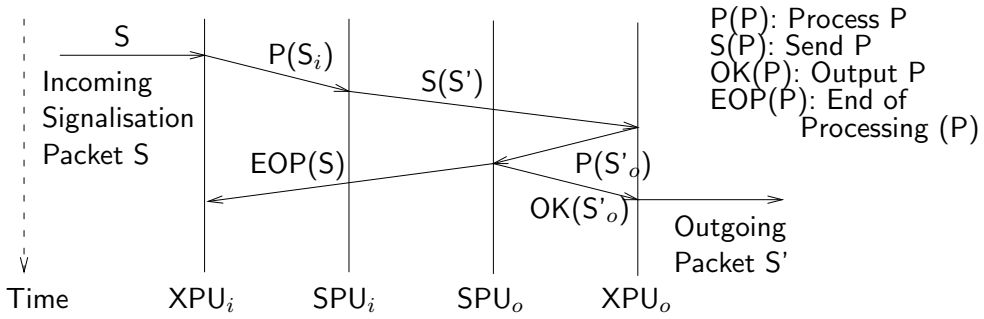


Figure 6: Typical signalling packet processing

abstraction.

This implementation fulfilled all its tests and was to be run on the actual TRANSPAC network, so we were quite confident on its correctness. We wanted to use it as benchmark to test validation methods. We develop a three stages validation procedure, using *ECHIDNA* because our Estelle description was too much big for our VEDA commercial version (1200 lines —100 transitions— for a single module).

1. interactive simulation, with a full view on internal protocol states. This was helpful to get rid of trivial errors due to transcriptions during the modelisation phase, but this kind of *white box* simulation (where one can see every interaction of the system) is rather slow.

2. *black box* interactive simulation, which looks like classical testing procedures. The T-PSS is a black box, and we can only check its external reaction when we feed it with test suites. Much faster than white box simulation, this method gave us a good confidence in our Estelle description as an accurate model of the real thing.

3. intensive automatic simulation: our T-PSS Estelle model is connected to two XCM automata (automatically derived from the abstract T-PSS Estelle specification), being able to randomly send both XCM correct and incorrect packets to the T-PSS and check the conformance of the T-PSS reaction to XCM.

   Upon error detection, the seed of the random number generator is recorded, and the simulation is executed again with that seed and an option to trace each internal interaction. This can be performed automatically, saving in a file only the 100 last lines of the trace for each error condition detected. If more information is needed to understand what happened, we can come back to step 1 with that precise scenario.

Using this approach, we detected some ($\approx$ 10 truly different) error conditions (in facts actual bugs) which could arise on the TRANSPAC network with very small probability, for the necessity of precise timings upon collisions of signalling packets in the T-PSS.

This study helped to explain and patch some real network problems —until then unexplained— and to improve validation test suites.

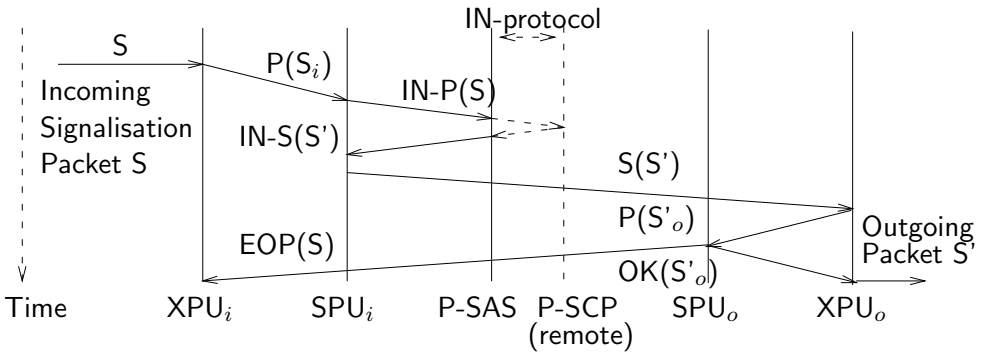


Figure 7: New signalling packet processing

2.3 Integration of the IN protocol in the T-PSS

The T-PSS being itself a parallel machine with its own internal protocols, the integration of the new IN protocol raises several problems, as potential signalling packet collisions and errors must be dealt properly.

The principles remain simple—the incoming-side SPU sends revelant signalling packets to the remote P-SCP, and waits for a proper request to be performed according to the IN protocol, see figure 7 for an example— but the informal specifications are rather tricky and show that the number of internal interactions within the T-PSS are twice the initial number.

The Estelle formal description of these new specifications is still in progress, but given the complexity of this protocol we expect interesting results from its validation. The interesting point here is that we can easily —thanks to Estelle modularity— simulate the behaviour of this new implementation in an environment obtained melting both environments developed during previous rounds: XCM and P-SCP entities, from whom we can use directly error detection features.

3 Conclusion on this experiment

3.1 About Estelle

What is said here is not a synthesis on strength and weakness of the Estelle language as itself, but much more an overall feeling about its usage in the above defined context.

Protocol people were already thinking in terms of communicating finite state automata, extended with some private variables and delay clauses, so no one found Estelle hard to understand: as we only use the usually called *static subset* of Estelle, there were few new concepts. Estelle was easy to be explained and taught, which is very important in an industrial context where people have limited time to learn new concepts.

In the IN specification project, it was quite easy to make Estelle description from both unformal specifications and pieces of automata —in facts easier than to remove ambiguities in the initial description.

On the other hand, it was very difficult to model real time systems communicating with both shared memory and message exchanges using Estelle. Indeed atomicity in this kind of system is very small (bounded by two consecutive accesses to the shared memory). As a fair shared memory construct does not really exist in Estelle, either one has to model every possible interleaving —which gives a lot of very small, unnatural atomic transitions— or to abstract from reality. After various tries with the first solution —every time more complicated— we finally choose to model the system at an higher level, and got interesting results. As far as we know, there is no suitable tool to deal with this kind of system, therefore we think that Estelle was still useful to do this job.

For instance, the kind of modularity offered by the Estelle *module* related constructs was appreciated to connect easily various simulated interface modules to the very same Estelle description of the T-PSS, or to its specification. Furthermore, this Estelle description is very readable —and brings a much better understanding of the protocol— and can be used as an implementation specification of the actual protocol, thus allowing us to perform some kind of validation before any new modification of this implementation.

3.2 The tools used

Using Verilog VEDA 1_00, we found *observers* a very interesting feature to observe global properties of a system. However, as we used the very first commercial —rather slow and closed— version of VEDA, we encountered various problems and limitations which are expected to be solved in future releases.

ECHIDNA features a much faster compiler (100 l/s) generating *C* code —thus interfacing with *C*-based software is easy—, but although it has very good interactive simulation capabilities (useful for fast first step debugging), its main interest is for distributed code generation to meet experimentation requirements on real systems. *ECHIDNA* has its own Estelle restrictions, and there is nothing in it equivalent to observers.

The main problem was the lack of a tool being able to make graph coverage analysis from an Estelle specification (with IS 9074 syntax) using unbounded FIFO queues. This is a serious obstacle against multi-side validation of a protocol, where one could mix partial graph analysis and interactive or automated simulation. We hope that new tools will soon fill this gap between exhaustive validation and simulation.

Finally, we saw in this experiment the interest of *open tools*, i.e. which can be easily interconnected with user's favourite environment. Here, it allowed us to build protocol demonstration models with user's friendly —mouse driven— interface to let protocol designer "play" with their creation in order to see if the formal specification fitted their ideas; and also to connect prototypes with already existing environments (X25 package for instance).

3.3 Economical balance

We have made a little study to value the cost of real network problems caused by T-PSS software anomalies versus early problem detection using specification and implementation modeling with Estelle. As this is a complex problem to address, numeric results should be interpreted with great care, above all before any generalization.

In facts, the only data we can collect is about *problem correction costs* and not really about

problem costs, which could be rather high (see for instance the billions of dollars lost by ATT with its famous bug in USA this year).

We collected three years long existing statistics on total maintenance costs for T-PSS at TRANSPAC, and tried to see how there were related to network problem identifications and corrections meanwhile.

To identify and fix network problems, we got an average used manpower of 0.4 month.man per network problem. At TRANSPAC, every hardware or software modification must fulfill a validation step (test suites checking intensively the modification, and no regression, interoperability and endurance testing of the full thing) before being installed on a pilot site on the real network. The extension to the full network occurs only several weeks later if no new problem is detected. This leads to additional costs estimated to 0.3 month.man per network problem fixed, so the average total cost of a network problem would be around $0.7 m.m$.

These data are valid for a cruising situation, where people know both products and maintenance procedures. In order to be fair, we have to assume that formal validation is performed by someone already knowing Estelle (or another FDT) and the TRANSPAC context.

Our experiment on the T-PSS internal protocol shows that a two months formal validation course could find out an high proportion of problems linked to signalling processing —usually detected during implementation, partial or full testing or even only on the actual network— and discovered around ten new bugs. However, it is difficult to draw any conclusion about exact savings, because we cannot know the proportion of signalling related patches vs. total maintenance costs.

Acknowledgments

I would like to thank both TRANSPAC people (and especially J. Méar, R. Rigault and P. Rocher) and IRISA people (C. Jard) who made it possible this fruitful collaboration between an industrial company and a research laboratory by their everyday help.

References

[1] R. Groz. Unrestricted verification of protocol properties on a simulation: an observer approach. In 6^{th} *IFIP International Workshop on Protocol Specification, Testing, and Verification, Montréal, Gray rock*, North Holland, June 1986.

[2] ISO 9074. *Estelle: a Formal Description Technique based on an Extented State Transition Model*. ISO TC97/SC21/WG6.1, 1989.

[3] C. Jard, R. Groz, and J.F. Monin. Development of VEDA: a prototyping tool for distributed algorithms. In *IEEE Trans. on Software Engin.*, March 1988.

[4] C. Jard and J.-M. Jézéquel. A multi-processor Estelle to *C* compiler to experiment distributed algorithms on parallel machines. In *Proc. of the* 9^{th} *IFIP International Workshop on Protocol Specification, Testing, and Verification, University of Twente, The Netherlands*, North Holland, 1989.

[5] TRANSPAC. *Spécifications Techniques d'Utilisation du Réseau*. Direction Commerciale, Tour Montparnasse, 33 av. du Maine 75755 Paris, 1989.