NPL Report CISE 1/95

October 1995

Experiences in the use of formal methods in the standardisation of a complex OSI protocol

R M Barker and F A Brady
Cnetre for Information Systems Engineering
National Physical Laboratory
Teddington
Middlesex
United Kingdom
TW11 OLW

Abstract

This report describes the development of a formal description (in LOTOS) of a complex OSI protocol (OSI Distributed Transaction Processing). The work was part of the development of the protocol standard and the formal description was an annex to the final published version of the standard. The work highlights many of the problems of describing a real-world protocol. These were particularly acute because the standard and the formal description were evolving together. The report also identifies some problems with LOTOS and discusses possible enhancements. The report describes how the formal description is continuing to be used and discusses what lessons can be learnt from the work.

NPL Report CISE 1/95

© Crown Copyright 1995.
Reproduced by permission of the Controller of HMSO

ISSN 1361–407X

National Physical Laboratory
Teddington, Middlesex TW11 0LW, United Kingdom

Extracts from this report may be reproduced
provided that the source is acknowledged

Approved on behalf of the Managing Director of NPL by
Mr A J Marks, Director, Centre for Information Systems Engineering

Contents

1 Introduction

This report describes the work done at NPL on the formal description (in LOTOS) of the OSI Distributed Transaction Processing protocol (OSI TP). The following section describes the history of the work from its start in 1990 to its completion in 1992 with the publication of the standard for OSI TP. The report goes on to introduce OSI TP and the particular problems it presents for formal description techniques; and then gives an overview of the LOTOS formal description technique and the problems found with LOTOS during the work. The next section details the additional problems of providing a formal description of a standard while the standard itself is being progressed. Finally, there is a section on the possible uses of the LOTOS description and the conclusions.

2 Background

A team of three people worked at NPL for three years on providing the LOTOS formal description to be included in the standard for the OSI Distributed Transaction Processing protocol. The worked was delegated to NPL by the ISO committee responsible for the standard after some initial work has been done on the LOTOS description elsewhere. JTC1 (the ISO/IEC Joint Technical Committee) had decided that formal descriptions should accompany protocol standards (where possible) and so the standard for OSI TP (ISO/IEC 10026-3) would include formal descriptions of the protocol in both LOTOS and Estelle (the two FDTs standardised by ISO), as annexes. In common with previous OSI standards, there was also an annex describing the operation of the protocol in terms of a state tables.

NPL took over the work at the beginning of 1990 and OSI TP progressed to International Standard at the end of 1992 and NPL was involved right up to the end. Obviously there was a need for liaison between the team at NPL and the ISO committee which was writing the base standard English text, originally attendance was at meetings at national level (BSI) and comments were submitted on the English text through BSI. As the work progressed, a large number of comments were generated as increasing numbers of problems were found with the English text, and NPL started to attend the international (ISO) committee meetings. The international work on the standard became so intense that discussion continued between editing meeting, on a network of participants communicating (principally) by electronic mail, and by fax. NPL was a of the major contributors to the fax network, working to resolve problems both in the text and the LOTOS description.

The resulting LOTOS description is 180 pages long, the Estelle description is about the same length and the state tables description is 110 pages long, with nearly 80 pages of tables. The English text is a hundred pages long, but some of the operation of the protocol has to be deduced from the service description, which is given in a different numbered part (ISO/IEC 10026-2) of the standard.

3 OSI Distributed Transaction Processing

The OSI TP standard defines mechanisms which allow several distributed systems to be part of the same transaction, with the guarantee that resources (normally database entries) will only be changed as a result of the transaction if all the various systems agree. In the terminology of concurrent processing, OSI

TP is a "two-phase commitment, presume rollback" protocol. The standard defines an application layer protocol which uses the lower layer of the OSI stack (e.g. Presentation and Session), and other standardised elements of the application layer: ACSE (for association establishment) and CCR (for commitment and recovery).

OSI TP is an attempt to standardise the functionality provided by a number of existing commercial products and so has great relevance to industry. OSI TP products will cover a wide variety of applications, including financial applications. The standard for OSI TP was of real commercial importance, and it was essential that it was a high quality piece of work, which was nonetheless produced on time. The work to produce a formal description of OSI TP was tackling a real-world problem, and had (according to other experts working on standardisation) a significant impact on the quality of the standard.

There are several aspects of OSI TP which make it difficult to specify formally. Firstly there is the sheer complexity of the protocol machine which is described as number of inter-communicating extended finite state machines, as illustrated below. Secondly, the number of connections to an OSI TP protocol machine can change during a transaction, so connections must be modelled in a way which can change dynamically and can not be modelled as a fixed part of the architecture of the formal description.

4 LOTOS

4.1 Introduction

LOTOS is a formal description technique which has been standardised by ISO. It is based on the Calculus of Communicating Systems (CCS) developed by Robin Milner at the University of Edinburgh. The intended meaning (formal semantics) of LOTOS is given a rigorous mathematical definition in the standard, and was the first international standard to use formal mathematics.

The main constituents of a LOTOS specification are processes, gates, and data. Processes constrain the way that actions in the specification can happen; they can be combined using various constructs so that the possible actions are interleaved, synchronised, or occur sequentially. Gates are the mechanism for simultaneous processes to inter-communicate, an action in the specification consists of data appearing at a gate. The data are elements of abstract datatypes, these datatypes are defined using an abstract datatyping language ACT ONE, which forms part of LOTOS.

When writing the LOTOS description of OSI TP, it proved very difficult to express various aspects of the description in a way which was at all natural. These problems are outlined below, together with suggestions as to how LOTOS could be enhanced to avoid the problems. A summary of these problems and enhancements was submitted to the ISO committee which is looking at possible extensions to LOTOS.

4.2 Datatyping

By far the most all pervading problem with LOTOS was the awkwardness of the datatyping language. To start with, every datatype has to be defined from scratch, there are no short cuts (e.g. macros). This would have been much easier if there was some convenient notation for constructing new datatypes (e.g. list) from old (e.g. element of the list). It would be better still, if the construction carried over to the new datatype definitions of basic operations,

such as equality (so that equality would be defined on list, if it were defined on the element of the list).

Equally frustrating is the cumbersome notation for element of datatypes. Because there are no special ways of constructing datatypes, there is no terse syntax/notation for the data. Values representing lists, records, and even integers have to be written out in terms of the definitions of the corresponding datatype. For example, all that is defined for the natural numbers are the number '0' and the operation 'succ', i.e. the successor operation "one more than". The number three is written 'succ(succ(succ(0)))', i.e. "one more than one more than one more than zero". In the OSI TP specification, names of service primitives number into the hundreds and so this notation is impractical. Instead one has to define all the constants that are needed, before hand, but this still takes a lot of space:

```
1 = succ(0);
2 = succ(1);
3 = succ(2);
```

etc.

4.3 Priority of choices

In many places in the protocol description, an error is detected when none of the prescribed actions is possible. We may have the a situation where

> if event event1 occurs then do process P1, else if event2 occurs do P2, else if event3 occurs do P3, otherwise call the process error to handle the error.

In the LOTOS it would be good to trap these errors by some (deterministic) choice construct such as:

```
event1; P1
[]
event2; P2
[]
event3; P3
[]
error
```

However this construct does not work, as the semantics of LOTOS allow this construct to handle the error even if 'event1' can happen, a non-deterministic choice is made between allowing 'event1' to happen and handling the 'error'. What is needed is some way of assigning priorities to choices, so that one branch would not happen if one of the other branches were possible. Similarly, it would be convenient to have a construct similar to the disables operator whose semantics were "if the process has deadlocked, i.e. no events are possible, then activate the following process".

4.4 Inflexible gates

In OSI TP, there are coordination rules which allow one finite state machine to lock up the other machines. This is to prevent a machine from having to deal with two inputs at once. This can be expressed in LOTOS, although it is rather complicated. Problems occur because the set of machines affected by such

locking up changes dynamically as the transactions progresses. This requires that the processes describing the individual machines must be able to switch on and off under the influence of the process which controls the coordination. Solving this problem requires much more complicated mechanisms and makes all the LOTOS code which describes the coordination rules completely unreadable. One way in which the solution to this problem could be made neater is for LOTOS to allow a process to dynamically change the synchronisation of a gate used by the process.

The verbosity of the description would have been reduced if gates were more flexible in other ways. If the synchronisation of the gates could alter with the type of data which appeared at the gate, this would reduce the number of gates needed. If some "wild card" notation was available to stand for any type of data at the gate, then that would simplify processes which just allowed events to occur without being in the data which appeared. Currently the signature of the data must match the signature expected by the event.

4.5 Uncontrolled iteration

The OSI TP protocol requires that the state machines called SAOs are created (and destroyed) by other state machines, this corresponds to OSI TP dynamically setting up new connections during a transaction. To model this, the following process could be used:

```
process create_SAOs :=
  new_SAO ||| create_SAOs
endproc
```

"let a new SAO operate in parallel with the process that creates new SAOs".

However this causes problems because an infinite number of processes are available to operate at the same time: this is a little awkward for the formal semantics, and impossible for any LOTOS tools to cope with. If the first event of the process to be created is known, i.e.

```
process new_SAO
  start; SAO
endproc
```

then the process which creates SAOs can be written as

```
process create_SAOs
  start;
  ( SAO ||| create_SAOs
endproc
```

and the problem is solved because the further iterations of the `create_SAOs` process will not be available until the event start has happened.

What is needed is a more general solution to this problem: a construct that iterates a given process, but only allows further instantiation of the process to be available once the first instantiation has been activated.

4.6 State vectors

The LOTOS description is highly object-oriented rather than constraint-oriented. A constraint-oriented approach was tried first, but it proved to hard to view the specification as being "constraint-driven". Inevitably, the development of

the English text specification of the standard had been unstructured and any attempt to retrospectively impose a modular constraint-oriented structure was impossible. Also, a constraint oriented LOTOS description would not have been maintainable against an evolving specification.

The object-oriented approach meant that much use was made in the LOTOS description of state-vectors to describe the state of the various machines. However the mechanisms for maintaining and accessing such state-vectors are awkward, especially when the information has to be shared between more than one machine. LOTOS is not designed to handle global state information, so it is difficult to see how this problem can be alleviated.

5 Describing an evolving standard

5.1 Overview

There were benefits from producing a formal description of a standard as it was being produced. The advantage for NPL was that there were experts on-hand who could explain how the protocol was meant to work. The advantage for the writers of the English text was that they could correct their specification when questions and comments from NPL revealed genuine problems.The disadvantages where that in many cases the various experts did not agree on the answers to the questions, and every time the English text changed (even to correct problem highlighted by NPL), many more changes would be necessary to the LOTOS description.

5.2 Problems of formally describing a changing specification

One of the major differences between the work on OSI TP and the normal application of formal description techniques is that the object being described was something of a "moving target". The English language text of the standard would change every six months, and as the final editing meeting approached (April 1992) these changes became more (not less) frequent, i.e. every two to three months. These changes were often very dramatic: the creation of a new state machine, the whole-scale move of functionality from one machine to another. This caused great problems for the development and alignment of the LOTOS description.

The main way that it was possible to keep abreast of the changes to the English language text was to have the LOTOS description follow very closely the English text. As a consequence, it was possible to go directly from a change to the English text to the corresponding place in the LOTOS description, and make the necessary changes there. This was another reason for the object-oriented approach taken in the LOTOS description. Even if a constraint-oriented structure had been successfully applied to one version of the standard, it is highly likely that when the standard changed the current structure would be useless and a new constraint-oriented structure would have to be fitted to the English text specification. Any given constraint-oriented description would not be flexible enough to meet changes in the standard.

The architecture of the LOTOS description mirrored the structure of the protocol machine in the English text, but many more processes were added to handle coordination of machines and data which was implicit in the English text. Whenever the structure of the protocol machine in the English text changed, so the LOTOS architecture had to change also; however apparently small changes

to the English text also resulted in (often quite major) changes to the LOTOS description. This would arise when one state machine had to pass information or data to another state machine, with which it had not previously communicated; this was perhaps an extra sentence in the English text, but in the LOTOS description it would mean the introduction of new gates and processes. Such a change might falsify assumptions which had been made earlier in the development so could have effects throughout the LOTOS architecture.

As the architecture changed, the state-vectors of the state machines would also have to be altered. Because of the inflexible nature of the datatyping language, carrying out the necessary changes to the datatyping could be a large amount of work, even for apparently inconsequential changes to the English text.

5.3 Problems of formally describing a specification designed by a committee

The members of ISO standards committees are not just technical experts, they are also representatives of national bodies, and of commercial companies; consequently they have to protect the positions of those national bodies and the commercial interests of their companies. Often the only way to resolve issues on which there are conflicting views is to compromise, rather than try and agree the correct technical solution. Because of the commercial interest in OSI TP and the complexity of the standard there have been many such compromises, with varying impacts.

There are subtle compromises where a form of words has been found which different experts understand in different ways. This gives rise to the position where different experts who have attended the same meeting can have different view on how the protocol is supposed to operate. Even more acute are the problems caused by deliberate ambiguities: a "diplomatic" form of words has been found (to resolve what is essentially a "political" stalemate), where there are two intended interpretations of the "diplomatic" form of words. In many cases these ambiguities can not be reflected in the formal description, this is, after all, one of the advantages of using formal methods. In practice the LOTOS description interprets such text in a way which could be made self consistent, and at times feedback to the ISO committee did cause some of these ambiguities to be resolved.

There were also some problems which were only apparent in the LOTOS; when reading the English text there did not appear to be a problem, and there were some areas (mainly concerned with the structure and coordination) which were not addressed in the state tables. In such cases the ISO committee were content to leave these problems unresolved, and the LOTOS description was left to find its own solution.

6 Uses of the LOTOS description

6.1 Initial experiences with tools

When completed, the LOTOS description was so large that it defeated all the LOTOS tools that it was given to. Most tools failed to read in the entire LOTOS description and none made any progress in trying to animate the description. The only verification possible of the LOTOS description in the standard was by manual comparison with the corresponding English text.

When attempts were made to animate some parts of the description, the user of the tool was offered a mass of internal actions, with no indication to their supposed interpretation. Some internal actions represent definite internal decisions in the protocol machine, corresponding to events outside the control of the real user of an OSI TP system; but most internal actions represent the data being passed between the different state machines within the protocol machine, and so these actions must occur for the machine to continue to function. In many tools all these internal actions are indistinguishable, and so it is very difficult to steer a path through the possible behaviour, without getting to a state where no useful events were possible.

6.2 Uses for conformance testing

NPL will use the LOTOS description for the development of conformance tests for OSI TP. NPL has already worked with the LOTOS description to produce test purposes for consideration by the ISO Upper Layer Conformance Testing (ULCT) group. Transformations were applied to the LOTOS description, to produce a form of the specification which looked liked test purposes written in LOTOS. The LOTOS test purposes were converted into English using some straightforward substitutions of English text for LOTOS identifiers and constructs. This was possible because of the correlation between the English text specification and the LOTOS description, this meant that the LOTOS was in a form in which useful transformations could be applied to it, and the translation from LOTOS back into English was possible. However the submission could not be accepted by ISO ULCT because they wanted to see cross references from test purposes to the state table and the English text does not provide the level of granularity of referencing required.

NPL is continuing to work on automatic test case generation from the LOTOS description. This involves mechanically searching the possible sequences of actions of the LOTOS description for behaviour which corresponds to that required by a particular test purpose. It is not likely that there is general solution to this problem, and most of the existing techniques are not applicable to "full" LOTOS, i.e. LOTOS which uses datatyping. The datatyping is essential to the LOTOS description of OSI TP: both to model structure of the service primitives and protocol data units, and to describe the state vector for the object-oriented approach. It is clear that new techniques, probably tailored to this particular specification, will be needed for the automatic test cases generation from the LOTOS description of OSI TP.

6.3 Recent experiences with tools

Since the standard has been published, we succeeded in using two tools to animate the LOTOS description.

LOTOSTOOL, HIPPO

The LOTOSTOOL would not read the description to start with; but when various table sizes in the source of the LOTOSTOOL were increased, and the tool set recompiled, it was possible to input the entire description. When we attempted to run the LOTOSTOOL animator (called "HIPPO") on the description, the menu of possible events that the user was presented with was very large and consisted mainly of events which could not occur. These events corresponded to attempts to synchronise actions of the shape:

```
g ? sp   ServicePrim   IsReq(sp) and .

g ? sp : ServicePrim [ IsInd(sp) and ..  ]
```

i.e. attempts to synchronise the receipt of the service primitive request by one process with the receipt of a service primitive indication by another process. Unfortunately HIPPO does not have the facility to discover that the the two guards were incompatible (because they can not be simultaneously satisfied) and so the event is impossible.

We got round this problem by labelling the internal gates with up/dn (for "down") to indicate whether this event corresponded to a received PDU appearing as an indication to the service user or a request from the service user to issue a PDU.

The actions above would become:

```
g   dn ? sp   ServicePrim [ IsReq(sp) and .   ];

g ! up ? sp : ServicePrim   IsInd(sp) and     ];
```

and HIPPO can tell that these events are incompatible, because it only has to compare the values dn/up.

This resulted in a much shorter menu being presented to the user and it was possible to drive the simulation. All internal events to push a service primitive from the service boundary to the presentation layer had to be sequentially triggered but it was normally obvious which event should be triggered at each stage to achieve the desired result. We succeeded in establishing associations and dialogues and in sending data, but then another obstacle was encountered.

HIPPO saves all states it has been in so it can step back though the states, enabling different paths to be explored. As each PDU required about ten events, and so ten new states, computer memory was quickly filled up just with storing all the previous states. In conjunction with the developers of the tool set, we tried to find a way to throw away old states but this proved impossible and we had to give up with HIPPO for animating the full description. Hippo has still proved useful in testing smaller parts of the description, and other LOTOS code we have written for the test suite validation tool.

TOPO

TOPO, from the University of Madrid, was more successful in animating the LOTOS description. It works by generating a C program from the description and compiling the program. It is possible to annotate the description so that particular bits of behaviour or datatyping can be treated specially. We worked closely with the University of Madrid when attempting to get TOPO to work on the OSI TP LOTOS description and we succeeded in producing an executable version of the protocol machine. We had to write our own code to produce menus to select events from the animated protocol machine and annotations in the LOTOS description allowed us to include in the menu only the events external to the protocol machine. All the events at internal gates which are enabled by preceding events flowed automatically without the intervention of the user. It was much less laborious to drive the TOPO animation than the HIPPO animation. TOPO does not save its previous states, which means it is not possible to back-track through the animation, but it does mean there was no problem with filling memory with saved states.

6.4 Validation of OSI TP Abstract Test Suites

NPL has produced a tool to validate abstract test suites against the LOTOS description. We simulate a test campaign using the TOPO animation of the protocol machine as the IUT. We have written our own Lower Tester which reads in the abstract test suite (in TTCN.MP) and plays out the test cases by communicating with the TOPO animation. This will reveal errors in the test suite when it is impossible to process the TTCN.MP or when the test case does not result in the verdict "PASS".

There is currently only one TP test suite (developed by INTAP) and it uses the coordinated test method. So, as well as a Lower Tester, we also needed to produce an Upper Tester (which plays the role of the service user). The Upper Tester behaviour (in the INTAP test suite) is described in a language called "UDL" (Upper Description Language) and the test suite contains a piece of UDL for each test case. Our Upper Tester was written in LOTOS and again animated by TOPO. A system (written in prolog) was produced which converted the UDL into a LOTOS data item, and the Upper Tester LOTOS description operates as an interpreter of the UDL code (as represented by the LOTOS data item).

The original validation tool handles version 1 TTCN (ISO/IEC 9646–3: 1992) and has been extended to handle version 2 TTCN (ISO/IEC 9646–3: 1994) which includes concurrency. Concurrency in TTCN permits the description of multiparty testing, which is required to test distributed transaction processing.

7 Conclusions

The main lesson from these experiences is that use of formal methods in the standardisation of a complex OSI protocol involves a large amount of effort on the part of those providing the formal specification. This effort would be much reduced if a structured approach was taken to the design of the protocol which allowed a (constraint-oriented) formal description to be used from the beginning.

The work was very successful from the perspective of OSI, the feedback from the team at NPL found a large number of problems with the English language specification, which lead to a remarkably high grade standard, considering the complexity of the protocol. This benefit to the final standard would have obtained from any work which attempted to provide a machine processable interpretation of the English text; but it is doubtful that such work would ever be done in parallel with the development of the standard unless the result were to appear in the standard. For our work on conformance testing, the LOTOS description is being used for: automatic production of test purposes, test suite validation, and automatic generation of test suites.

There are several possible enhancement to LOTOS which would improve the conciseness and readability of a LOTOS specification. There is a need for a more user-friendly datatyping language, more flexibility in the synchronisation of gates and the expression of actions, and a means to express the precedence between various (non-deterministic) choices.

A Further reading

1. ISO/IEC 10026-1:1992 Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 1: OSI TP Model.
2. ISO/IEC 10026-2:1992 Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 2: OSI TP Service.
3. ISO/IEC 10026-3:1992 Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 3: Protocol Specification. Including Annex H: LOTOS description of the TP protocol
4. EWOS/ETG 18 Part 1: OSI TP tutorial — General Overview. September 1992
5. EWOS/ETG 18 Part 2: OSI TP tutorial — Services. May 1993
6. ISO 8807: 1989 Information processing systems — Open Systems Interconnection — LOTOS — A formal description technique for the temporal ordering of observable behaviour.
7. ISO/IEC JTC1/SC 21/WG 1/N 1157 Contribution on LOTOS enhancements.
8. Ken Turner. A LOTOS-Based Development Strategy. In Forte '89, Vancouver December 5th — 8th, 1989.