1

Open Issues in Conformance Test Specification

Bernd Baumgarten
GMD, Rheinstr. 75, 64295 Darmstadt, Germany

ABSTRACT

In this paper, we deal with semantic problems of test specification in the OSI Conformance Testing Methodology and Framework, especially in the test notation TTCN. They concern mainly the PCO model and the test purpose/ test verdict complex. In either case, we identify several open questions. We show up merits and disadvantages of various ways to fill the remaining gaps in standardization, and recommend some specific solutions.

1. INTRODUCTION

As any regulation written in natural language and by human beings, protocol and protocol test standards are prone to gaps, ambiguities and contradictions. Such defects are often felt more painfully in testing than in the implementation of protocols. The reason is that protocol implementors will often produce "satisfactory" implementations by creating their own interpretation the standard, complementing it or deviating from it in reasonable ways whenever necessary. If different interpretations lead to interoperability problems, these may be ironed out in implementor groups harmonizing the implementations. On the other hand, clients of a test laboratory, trying to obtain a favourable test result and later a certificate, but confronted with negative test results caused by defects in test standards, may become quite annoyed. Testing and certification procedures do not leave much room for compromise. Therefore, test standards require particular care and scrutiny.

In this paper, we deal with some test specification problems encountered in the OSI Conformance Testing Methodology and Framework (CTMF) [1], especially in the test notation TTCN as published in [2]. They concern mainly

- the incomplete specification of points of control and observation (PCOs) and undesired effects of frequently found interpretations of the PCO model, as well as
- unclear effects of non-determinism in protocol specifications on the choice of test purposes, and, closely related to that, weaknesses of verdict assignment rules.

Some of these problems arise not purely within TTCN, but rather from the interplay of CTMF and other OSI standards. Some were already addressed in the standardization work, but could not be solved due to reasons of schedule and compromise. As far as we know, the progression to the next version of TTCN, due in 1995, which is to include two amendments, will not affect these problems in any way.

For the sake of readers acquainted with CTMF, let us remark here that, in order to provide a clear and simple picture of these problems, we stick to a widespread illustrative paradigm, used in many discussions of conformance testing: We confine ourselves to protocols and do not talk about profiles or information objects etc.; we confine ourselves to single-party testing, in particular to the Distributed test method, to layers without substructure, to single-layer and single-protocol IUTs, and to ASPs (and not PDUs) as the primary kind of events

T. Mizuno et al. (eds.), *Protocol Test Systems*

dealt with at PCOs. The terms which are essential to understanding our arguments are explained in the paper.

In order to give an unbiased view, we do not speak in terms of specific formal description techniques, such that no one will be tempted into dismissing the problems as merely description language specific issues.

In Section 2, we summarize some traits of the OSI BRM and service conventions. In Section 3, we will deal with the PCO model in TTCN, the motivation behind, and the problems arising from it. We only explain the barest basics of TTCN. Readers desiring to gain more insight may consult [2, 3, 4]. Section 4 will be devoted to problems centered around protocol non-determinism, test purposes and verdicts. Throughout the paper, statements derived from standards (BRMx, QUx, DISC) or implementation needs (IMPLx) are analysed and contrasted with one another, leading to problems and questions (PRx).

2. A SHORT RÉSUMÉ OF THE OSI BRM AND THE ROLE OF SERVICES

In this section, we summarize the OSI modelling concepts from [5] and [6]. Later, we will relate them to CTMF notions.

2.1. OSI Basic Reference Model

The OSI *Basic Reference Model* (BRM) provides a framework in which the OSI protocol specifications can be expressed. In its simplest form, as shown in Figure 1, i.e. disregarding routing and relays, it models two communicating *open systems*, A and B, as two stacks of corresponding *protocol entities*, A_1-A_7 and B_1-B_7, residing on top of a global physical medium. The *(N)-service* is provided by the interplay of the entities A_N, A_{N-1}, ... , A_1, of the physical medium, and of the entities B_1, ..., B_N; in other words, the (N)-service provider consists of A_N, B_N, and the (N-1)-service provider.

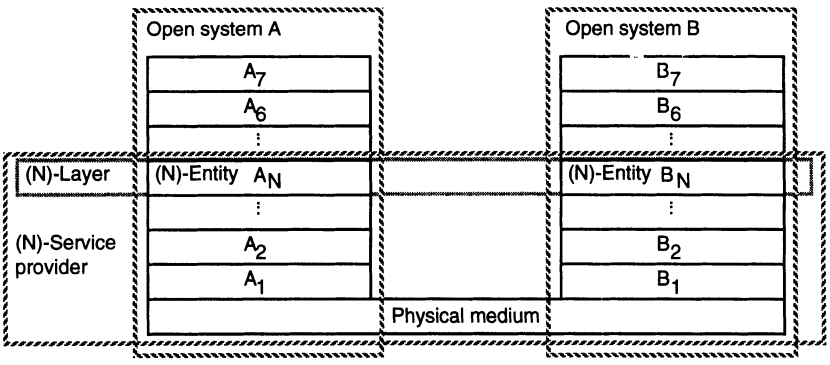

Figure 1. The OSI BRM global structure

2.2. ASPs and PDUs

Protocol entities are communicating locally at *service boundaries*, to be more precise, at *service access points* (SAPs), by means of *service primitives*. In Figure 1, this local communica-

tion is going on between vertically neighbouring protocol entities. We will abbreviate "service primitives" by "ASPs" (abstract service primitives), as in CTMF terminology.
Protocol entities are also communicating, via lower layer entities and the physical medium, with peer protocol entities, by means of *protocol data units* (PDUs). In Figure 1, this communication can be interpreted to be going on horizontally, e.g. between A_N and B_N. In the operational view of the BRM, the (N)-PDUs are wrapped up in (N-1)-ASPs, i.e. they are communicated vertically as data components (called *parameters*) of ASPs, see Figure 2. Some clarifying views on ASPs were developed in [7].

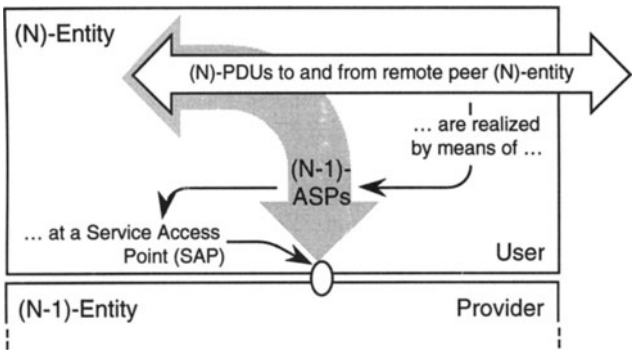

Figure 2. ASPs and PDUs

An ASP is an abstract, atomic, implementation-independent representation of an interaction between an OSI-service user and its OSI-service provider. The interaction shall be regarded as taking place as an instantaneous event, which cannot be interrupted by another interaction. ASPs do not necessarily imply implementation-specific mechanisms or direct relationships to protocol elements. ASPs are invoked by either user or provider. They may involve bidirectional information exchange, but they have a direction indicating the main information flow. It seems that they are invoked by the producer of the main information.

The OSI-local view and an OSI-service definition are described in terms of the set of ASPs which the OSI-service-user and the OSI-service-provider are allowed to exchange, together with the sequencing rules which apply to these exchanges, thus defining an "ASP language." Existing OSI-service definition standards often do not (fully) describe the global effects of the service as observed by two remote service access points, although [6] requires that "the correlation among OSI-service primitives for this set of OSI-local views" be defined.

[6] allows for strictly local interactions between user and provider in implementations that do not count as service primitives and are not considered in the service definitons. These may, in particular, involve declarations of willingness to issue or of readiness to accept an ASP and similar flow control informations.

2.3. Operational model and reality

Note that the BRM, together with the relevant protocol specifications and service definitions, provides only an operational specification for correct behaviour of local OSI-open systems. The valid behaviour of an OSI-open system is, in the last analysis, only specified with respect to the exchange of *bit streams* via the physical medium and not with respect to ASPs at the upper boundary.

Operational specifications and operational semantics are widely accepted as a means of defining behaviour. A well-known example is the definition of computable string functions in terms of a Turing machine model working with mechanically oriented notions. In a very similar manner, the OSI concepts of vertically neighbouring protocol entities, exchanging service primitives at service boundaries, are merely fictitious constructs, made up to convey in an operational manner the definition of correct behaviour of a stack of protocol entities. An implemented open system need not execute any activities that could be identified as a counterpart for any of the ASPs involved in its specification. We will see that in testing the situation is somewhat different.

OSI explicitly does not prescribe actual implementation structures, cf. e.g. "Only the external behaviour of open systems is retained as the standard of behaviour of real open systems" in [5].

Commercially available OSI implementations are often only *partial* open systems, which do not span all 7 layers. The realization of the uppermost service boundary of an implementation as an actual interface is not regulated by OSI standards. It is however a widely accepted scheme that OSI implementations provide an actual "upper" interface representing their upper service boundary, at which the occurrence of ASPs is an observable reality, even though it is completely left to the implementors how the upper ASPs are realized, e.g. by means of asynchronous mechanisms. The lower interfaces of lower layer entities are quite well described, down to physical characteristics of plugs.

As the real activities in implementations constituting an ASP necessarily consume time, the question arises what is actually retained in reality from the atomicity of an ASP and its common performance by user and provider.

2.4. Collected theses

BRM1. OSI does not prescribe actual implementation structures, neither of the interior, nor of the interfaces.

BRM2. In protocol specifications, ASPs are common atomic actions of a user and a provider at a service boundary, occurring in observable orders. Incomplete ASP attempts are not considered. ASPs are abstract constructs, introduced to define valid open system behaviour operationally.

BRM3. An OSI service definition constrains the behaviour of the OSI-service user by permitting only selected sequences of ASPs.

IMPL1. Many OSI implementations provide an upper interface with real counterparts for upper ASPs. User and provider at the SAP always observe the same order.

3. ASYNCHRONY AND PCOS

Many authors found that the structure of the points of interaction between the tester above and the tested implementation or the underlying provider below have a marked influence on the formulation of the test cases and on the verdicts assigned to event sequences in a test, e.g. [8,9,10,11,12]. One structure for these points is described in CTMF, in particular in [2]. This description is not unproblematic, as we will discuss in this section.

3.1. Conformance testing and ASPs

In this subsection, we give a short summary of the operational semantics of TTCN on a level that does not force us to describe too many details of TTCN syntax.

CTMF AND TTCN BASICS

Under the simplifying assumptions made in the introduction, in protocol conformance testing we are faced with the following problem: The behaviour of an implementation of an

(N)-layer protocol entity, called *implementation under test* (IUT), is to be analyzed by means of a *tester* sitting on top of the IUT and on top of the common *underlying service provider* (USP) realizing an (N-1)-service used by both the IUT and the Tester. Their interaction points, called *points of control and observation* (PCO), are called UT and LT in Figure 3, reminding of "upper and lower tester."

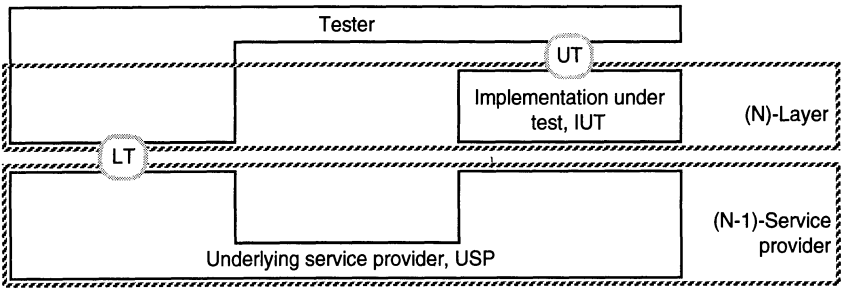

Figure 3: A simple test configuration

We note the following requirement in testing:

IMPL2. Service boundaries have to be realized as actual interfaces, if testers are to be granted access to them.

The tester executes test cases written in TTCN, describing possible sequences of events. In the sequel we will often have to speak of "the IUT and/or the underlying service provider USP." We will abbreviate this by "the IUT/USP." CTMF represents the tester by two entities, called abstract testing functions, a lower and an upper tester, and considers different ways how these two can be coordinated. This substructure does not matter in the following discussions and was therefore omitted from the picture.

An event is the reception or emission of an ASP (a Receive, or a Send) at a PCO or the reception of a notification from an expired timer that was started earlier by the tester (a Timeout). In contrast with [6], TTCN assumes ASPs to be unidirectional and of two general kinds: Send ASPs are initiated by the tester and transfer information exclusively from the tester to the IUT/USP. Receive ASPs are initiated by the IUT/USP and transfer information exclusively from the IUT/USP to the tester. The rare OSI ASPs that transfer information both ways have to be substituted by short sequences of Send and Receive ASPs.

TTCN test cases are expected to be written in a way that the tester behaviour is deterministic, such that the next ASP it sends or the verdict it reaches is always uniquely determined by the previous history of events. The aims behind this determinism requirement, namely the repeatability of test cases with the same results, motivates several more rules of TTCN. Note that repeatability is related with impartiality towards test clients, because it precludes that the execution of a test case takes arbitrary turns and goes to different depths with different clients.

A test case consists (in a simplified picture) of a behaviour tree, describing possible sequences of events or event patterns. It is written by means of indentation, where alternatives are left-aligned at the same level of indentation and the sequentially next event is indented below its predecessor. Figure 4 demonstrates the principle, omitting a few details required in full TTCN. The maximal event sequences described are (UT!A, UT?B), (UT!A, LT?C, LT!D, UT?E), and (UT!A, LT?C, LT!D, UT?F).

```
UT!A
   UT?B
   LT?C
      LT!D
         UT?E
         UT?F
```

Figure 4: A (simplified) TTCN behaviour tree

During the execution of the behaviour tree, a path through it is chosen, depending on the events actually happening. The next event to be executed is always determined from a list of next alternatives. Such a list is either a "Receive list", i.e. a collection of Receive, Otherwise (a sort of indiscriminate Receive) or Timeout alternatives, or a single Send alternative. Note that this rule for alternatives does not appear explicitly CTMF, but it can be inferred from the standard and the observation that alternatives that are obviously never reachable can be dropped [8].

PCOs are closely related with SAPs; we will shortly come back to this point. Let us observe here that the SAPs at UT and LT will usually be realized in systems remote from one another, i.e. the tester is a distributed system. In non-concurrent TTCN, the behaviour tree describes global orders of events at LT and UT.

PCOs AND SERVICE BOUNDARIES

Note that a real tester, or at least a component of it, is interacting with the upper service boundary of the IUT. In the OSI BRM, this boundary was only a conceptual abstraction; in conformance testing it has to be actually realized as a useable interface. This creates a conflict of aims:

- Outside of CTMF, OSI standards explicitly renounce prescribing details of the realization of service interfaces, just as they renounce prescribing details of protocol entity realization.
  Customers acquiring an OSI implementation will ordinarily accomodate any user programs they want to run on top of it to the available interface; they can be seen as "partners" of the implementor.
- But if this boundary has to be accessed by the tester, and the tester's behaviour towards the IUT/USP and its possible interactions with them have to be specified precisely, then this indifference is not acceptable: Imagine, to use an analogy, a class of students required to write an exam but being permitted to decide arbitrarily in which language, on which medium and at which speed they will do so. In CTMF, a tester, as an examining instance, would therefore be served best by a complete specification of the upper IUT/USP interfaces both at UT and at LT.
  Test laboratories accessing the IUT's upper interface are rather to be seen as "opponents" of the implementor, because it is their task to spot errors. If they cannot get along well with the upper interface, they might consider some of the less successful interactions as IUT errors. The LT interface is less of a problem, because in the tester's local system we have again the "partner relation."

One aspect slightly defuses the "opponent relation" at UT: Implementors of tester software also want to sell their products and will try hard to enable them to cooperate smoothly at least with the more common types of IUT upper interfaces. But neither will this do justice to a good but unconventional IUT, nor does the appeal to common sense and good will solve all interoperability problems – why else would we need standards and testing, in the first place?

In the absence of standardized interface definitions, TTCN and CTMF came up with a minimal solution to this dilemma, by fixing a (somewhat sketchy) structure of PCOs, giving them slightly more structure than that provided by the atomic interactions:

QU1. The PCO model is based on two FIFO queues: one *output queue* for the tester sending ASPs and one *input queue* for the tester receiving ASPs (under our simplifying assumptions). These queues are free of loss and reordering. No capacity bound is mentioned in CTMF. These queues are explicitly meant abstractly, not as implementation requirements.

QU2. The output queue is assumed to be located within the IUT (at UT) and within the underlying provider (at LT), respectively.

With these provisions, TTCN feels safe to postulate:

QU3. ASPs, at least valid ones (cf. BRM3), can always be sent by the tester and are never lost.

3.2. Resulting problems

THE INPUT AND OUTPUT QUEUES

By QU2, TTCN prescribes a queue at the upper boundary of the implementation for ASPs sent from above. This may be seen as contradicting BRM1 and QU1. Hence, TTCN users are faced with the problem:

PR1. Which of these two positions, BRM1 and QU1, has precedence? Can one actually ask for a tester output queue in the IUT/USP? And if not: should it be dropped or placed somewhere else?

The second problem naturally arising from the PCO queues is:

PR2. If the PCO queues are retained, what can be assumed about their capacities?

An analysis of the effects of the tester output queue on the observable behaviour of the IUT/USP depends partly on the capacity of the queue. If it is unbounded, then the only observable effect is that send events from above always succeed "immediately."

Now, unfortunately,

IMPL3. Unbounded queues are usually very hard to implement.

If the unbounded PCO output queue (or, at least, its effect QU3) were dropped, we would face a flow control problem. A guarantee for the regular success of tester send events would require particular provisions like the following:

(a) The IUT/USP processes ASPs issued by the tester at any speed and in any number – a not too realistic demand.

(b) The test suite writer uses prudence for limiting, and timers for delaying, its Send events to a degree that the IUT/USP is confronted with tester-initated ASPs only at a rate that it is explicitly required to cope with in the relevant specifications. Presently, however, such rates are usually not mentioned in standards. Nor is there any mention of how long a user may have to wait for an ASP to succeed.

(c) Another way out of the dilemma would be to permit loss of ASPs at PCOs before they are received; but this would entail an excessive loss of control by the tester and even more reachability analysis to be performed by the test suite writer.

(d) Analysis of OSI protocols may reveal that validly interacting instances fill the queues only up to a calculable fixed upper bound. It could be required that the tester, even in displaying invalid behaviour, should not send more ASPs in a row than the queues are

required to hold under valid circumstances. However, TTCN could then only be used for a small set of protocols, which happen to fulfill some strong requirements.

THE DISCONNECT PROBLEM

The *Disconnect problem* turns up if the input queue is at the tester side of a PCO and the output queue is inside the provider. In many OSI protocols, cf. e.g. [13], in the user-provider dialogue, as soon as a connection (or association etc.) is set up, the provider may issue a Disconnect (or similarly named) primitive any time it desires. After that, the connection is discontinued, and the user cannot perform any more service primitives with the provider, except those required for establishing a new connection. If the user is the tester, this means that whenever it tries to send in a test case, the provider may meanwhile have issued a Disconnect that is now sitting in the input queue. The Send cannot possibly succeed, contrary to requirement QU3.

DISC. If a provider has issued a Disconnect, ASPs related to the disconnected dialogue and attempted by the user will no longer succeed. If the user is unaware of the Disconnect, it cannot possibly know which ASPs will be successful next.

The Disconnect problem shows that Sends may not always be successful. Even if the tester has made sure there is no Disconnect in the input queue and then turns to the output queue to "send" an ASP, the Disconnect may have occurred while the tester "was turning," because Disconnects are usually unconfirmed and considered to be possible at any time.

PR3. DISC conflicts with (QU1-3).

Note that present TTCN prevents the tester from checking for the absence of Disconnects before every Send, anyway: looking can only happen in a Receive line, and Receive lines must not be followed by Send lines in a list of alternatives.

PCO QUEUES VERSUS SERVICE BOUNDARY

If we retain the queues, unbounded or not, we are faced with the question:

PR4. When do the ASPs actually "happen" at the service boundary:
- on being put at the back of the input or output queue or
- on being taken out at its front?

In other words: Where exactly is the service boundary located at a PCO that is associated with the two queues?

If we consider unbounded input and output queues, and if we put aside exotic variants (like having the service boundary cut across these queues or permitting unconventional operations on the queues), then the four possibilities shown in Figure 5 remain. The "Tester core" in the illustration represents the fictitious entity looking at the queues and executing the behaviour tree; the tester comprises everything outside the IUT/USP. In (A), e.g., the input queue belongs to the tester, and the output queue belongs to the IUT/USP. In fact, having the input queue inside the IUT/USP (as in B and C) seems a bit questionable: what should be the meaning of a "parking" status of an ASP not having occurred yet?

If we want to decide which of these PCO structures fits best the various requirements posed by various standards and the needs of real testing, then we have to check for each one, whether it places the output queue into the IUT/USP (QU2), whether it refrains from restricting the implementation beyond the OSI behaviour specifications (BRM1), and whether the send event is always successful (QU3). Finally it should avoid serious practical problems with the real testing of real protocols (IMPL3, PR3). Table 1 shows the assessment of A-D.

If we simply count the arguments introduced in favour and to the disadvantage of the four PCO structures, then D gains a narrow victory, with B and C as runners-up. None of the four candidates meets more than three out of the five demands. More than four cannot be expected, anyway, because they are in conflict, as shown before. We desisted from deriving a

linear ranking between the four models, e.g. by assigning different weights to the various demands.

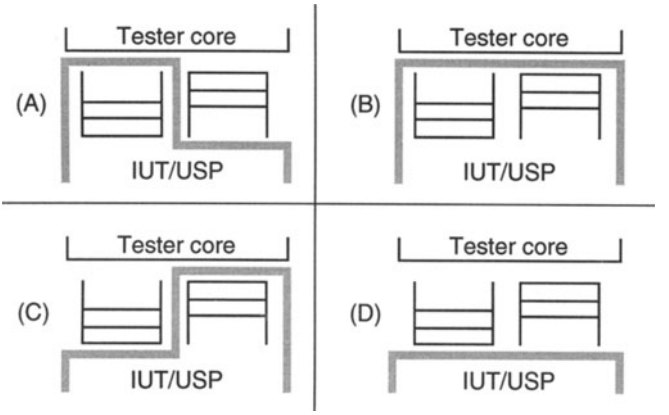

Figure 5. PCO structures relating two unbounded queues to the service boundary

Table 1.
Some pros and cons of possible PCO structures

Demands to be met \ PCO structure	A	B	C	D
Output queue in the IUT/USP	+	+	–	–
No implementation details of IUT/USP	–	–	–	+
Sends always successful	–	–	+	+
Unconditionally implementable	–	–	–	–
No "IUT/USP Disconnect" problem	–	+	+	+

For lack of space we leave it to the reader to investigate the suitability of various other possible PCO structures, like combinations using bounded queues or synchronous interactions etc. We tried quite a few of them, and they always revealed similar disadvantages, besides being even further away from the BRM and the (admittedly not very specific) TTCN model.

3.3. Alternative PCO structures

ATOMIC ASPs

Let us assume for the moment that, contrary to CTMF, we apply the abstract service boundary model to TTCN test specifications, equating PCOs with SAPs and ASPs with atomic interactions between the tester and the IUT/USP. Even such a very abstract view leaves room for various conflicting interpretations. As opposed to BRM2, a real tester can observe more than only which ASPs have completed in which order, i.e. more than the ASP words happening at the service boundary in the BRM. For example, if the IUT is due to issue

an ASP at UT by a certain time, but erroneously does not, then the tester can use a clock to find out and call the test case failed. But just how much more can it observe? In particular, with respect to Send events, the following questions arise:

PR5 What happens if the user of the IUT, in our configuration the tester, issues a valid ASP at UT, and the IUT (which does not necessarily implement its specification), will not accept it? Can the two deadlock within this ASP? Can the user recognize that the provider declines and call this a failed test case? These questions affect the usability of test cases as well as verdict assignment in real testing and therefore cannot be put aside as mere implementation details.

PR6 What if the IUT, e.g. in some kind of backpressure situation, cannot accept the ASP right now, but would in principle accept it later, and therefore defers acceptance without downright declining it? Will the user recognize the delay? If so, will it wait arbitrarily long for the ASP to complete or stop the attempt? These questions are of similar rank as those under PR5

Questions of this kind are not unknown in theory: Process algebra deals, in various ways and usually without mentioning time explicitly, with at least some of these choices.

Assume test case execution is simply delayed in a Send event until the IUT/USP is able and willing to accept it. At least, this might be the view of TTCN operational semantics; in reality, this delay could be filled with local interactions. Of course, if a Send can delay indefinitely, it can also delay for unreasonably long times. Testers, however, always have only a limited amount of time available. Hence, delaying Send events calls for a facility that is presently not provided in TTCN: something like a "send-timeout" or an "elapsed-patience-default," which is attempted as an alternative to Sends that remain unsuccessful within reasonable time, in order to prevent excessive waiting for a Send to succeed. What reasonable time is, could be agreed between the test client and test laboratory (e.g. in the PIXIT, cf. 4.1). Note that with such facilities the snapshot does not determine any longer the outcome of an attempt of the current list of alternatives – no wonder, considering the Disconnect problem. At any rate, in Table 1, such an approach would earn a "+" on each applicable criterion except the guaranteed success of Sends.

SPECIFIC INTERFACE DEFINITIONS

Theoretically, one could implement any number of different types of upper interfaces for OSI implementations, making it hard for real testers both to connect to the SAPs at all, and, sometimes, even to find out whether they have just found an error or only misunderstood the interface. Prior interface testing, not standardized in any way, offers only some ad hoc relief. Standardized interfaces, no matter whether uniform throughout OSI or protocol-specific, would definitely help. So would, alternatively, a standardized *interface definition language*, permitting IUTs to be accompanied by a description of the realization of the upper service boundary (e.g. in the PICS, cf. section 4). The realization problem of ASPs was already addressed in [14].

4. NON-DETERMINISM, TEST PURPOSES AND VERDICTS

4.1. Non-determinism in protocol specifications

OSI specifications leave many choices for implementations, which we will now attempt to classify under practical aspects.

Static choices are choices in the implementation production process. Some functions of protocol entities are not mandatory. They can either be freely implemented or not (i.e. purely optional functions) or they can only be realized in particular combinations (choice or conditional functions). Furthermore, protocol entities may have parameters influencing the behaviour, that can also be chosen either freely or only in certain combinations. As the reali-

zation of non-mandatory functions can be considered as a value-assignment to, e.g. Boolean, parameters, value parameterization represents a general mechanism for static choices. These parameters are usually also parameters of the corresponding test suites. By corresponding value assignments in the test suite, the test cases are dedicated to the particular realization of the specification. The test client completes a form by filling in these parameter values. The completed form is called the PICS (protocol implementation conformance statement).

Other static parameters are rather parameters of a conformance test than of the implementation. System addresses and test-specific timers, e.g. to limit the duration of the test case, are typical examples. The test laboratory and the test client agree on these parameters in the PIXIT document (protocol extra information for testing).

Dynamic choices concern the remaining non-determinism of the IUT implemented with the parameter values written down in its PICS and PIXIT, i.e. after the static choices have been eliminated. Test cases have to provide the proper branches to cope with any valid choice. For the sake of simplicity we limit our analysis to binary choices. From a practical point of view, there are several flavours of choices, distinguished by factors such as preferability and controllability:

- choices between alternatives of "equal value," i.e. for none of which any preference is discernible, as opposed to
- choices between a "preferred, regular" behaviour and an "undesired, exceptional" behaviour, the latter only to be used in emergency situations that are expected to arise only seldom, furthermore
- choices that are expected to be controllable by the IUT and its local system, but for whose decision mechanism the specification just does not care, as opposed to
- uncontrollable choices, e.g. if a request is usually granted but may be turned down due to a temporary lack of resources.

It seems that in practice equal value choices often coincide with controllable choices, and regular/exceptional choices coincide with uncontrollable choices.

We will not consider the kind of choice created by gaps in the specification, which may be interpreted as permitting arbitrary IUT behaviour. Such situations should lead to defect reports concerning the protocol specification. Unfortunately, in practice, this does not always happen.

4.2. Test verdicts

CTMF defines the possible test verdicts as follows:

"*pass (verdict)*: A test verdict given when the observed test outcome gives evidence of conformance to the conformance requirement(s) on which the test purpose(s) of the test case is (are) focused, and when all test events are valid with respect to the relevant specification(s).

fail (verdict): A test verdict given when the observed test outcome either demonstrates nonconformance with respect to (at least one of) the conformance requirement(s) on which the test purpose(s) of the test case is (are) focused, or contains at least one invalid test event, with respect to the relevant specifications.

inconclusive (verdict): A test verdict given when the observed test outcome is such that neither a pass nor a fail verdict can be given."

Let us ask about the interpretation of some central terms and phrases in these definitions:

PR7 What is a valid test event ?

PR8 What is a conformance requirement?

PR9 What is a test purpose? What is a test purpose of a test case?

PR10 Which are the conformance requirements a test purpose focuses on?

PR11 When is evidence of conformance to a conformance requirement given?

PR12 When is nonconformance to a conformance requirement demonstrated?

In the following subsections, we will point out possible answers to these questions.

4.3. Validity and conformance requirements

In the configuration of Figure 3, the IUT/USP is observed interacting with the test case executed by the tester. An *IUT/USP observation* obtained this way is a sequence of observation events that correspond to the successful lines of the test case. Consider each observation to include its relevant data, such as PDU types and values, variable values or timer readings. We may assume that verdicts are assigned exactly in the leaves of the behaviour tree. Other possibilities can be reduced to this case.

Each IUT determines a set of possible *IUT behaviour sequences*, by which we denote sequences of interactions (including their timing) between the IUT and any arbitrary environment obeying the upper and lower service descriptions. In Figure 3, the tester cannot record directly IUT behaviour sequences, but only the IUT/USP observations. The possible IUT/USP observations are determined by the possible IUT behaviour sequences, the underlying service, and the test case.

The protocol specification determines the set of *valid IUT behaviour sequences*. The *valid (IUT/USP) observations* are determined by the valid IUT behaviour sequences, the underlying service, and the test case. Each valid observation consists entirely of *valid test events*. Certainly, the first event in an observation which shows that this observation is not valid, should be called an *invalid test event*. It is unnecessary to assign a validity status to subsequent events in the invalid observation. The IUT conforms to its specification if each possible IUT behaviour sequence is a valid one.

As conformance requirements (CRs) are not very clearly defined in CTMF, several views have been adopted; CRs have been identified with properties of protocol specifications, with clauses of protocol specifications, with properties of behaviour sequences, etc. Usually, a CR, regardless of how it is formulated, determines a set BS(CR) of *CR-conforming IUT behaviour sequences*. Usually the set of all CRs is equivalent to the protocol specification, in the sense that the set of all valid IUT behaviour sequences coincides with the intersection of all BS(CR), taken over all CRs. The *CR-conforming observations* are determined by BS(CR), the underlying service, and the test case.

4.4. Test purposes

According to CTMF, a test purpose is a prose description of a narrowly defined objective of testing, focussing on a single or some closely related CRs. For the sake of simplicity we assume that a test purpose TP, similarly to a CR, determines a set BS(TP) of behaviour sequences. It also calls for particular tester behaviour, TB(TP). This corresponds to the empirical fact that test purposes require special behaviour from the tester, or even from the IUT – we will dwell further on this point below. TP is focusing on CR if BS(TP) is a subset of BS(CR). TP is a test purpose of a test case if the tester behaviour defined by the test case stays within the limits of TB(TP).

Of course, as test verdict assignment involves the notion of test purposes, it is important to define unambiguously what a valid test purpose is. At present, it is very much an open question what a test purpose may require and how it is stated.

Some of the questions still remaining to be answered in this context are the following:

- Protocol specifications often use an extended finite state machine driven by ASPs and the PDU contained therein. Should test purposes require driving the "protocol machine" into particular states, or through particular state transitions?

- Should test purposes require performance of certain global event sequences? Should they prescribe the tester's Send actions in this sequence, or the complete sequence of tester Sends and Receives, or rather the Sends and Receives at the IUT's upper and lower boundaries?

At the present stage of standardization, these questions may be answered arbitrarily by the test purpose writers, depending on personal taste.

4.5. Evidence of conformance and nonconformance

There are two different intuitively reasonable views of when evidence of conformance to a CR is given:

(a) The recorded IUT/USP observation is CR-conforming. Informally, the IUT "may have behaved in conformance with CR."

(b) The recorded IUT/USP observation proves that the IUT behaviour sequence was CR-conforming. Informally, the IUT "behaved in conformance with CR."

The difference between these two views lies in the faithfulness of the underlying service. If, for example, the USP may have lost output from the IUT during the test case performed, then at most (a) may be inferred, but not (b), because the lost output may have been invalid.

Analogously, two alternative views of when a test outcome demonstrates nonconformance with respect to the a conformance requirement are:

(i) The recorded IUT/USP observation does not rule out that the IUT behaviour sequence was not CR-conforming. The IUT "may not have behaved in conformance with CR."

(ii) The recorded IUT/USP observation is not CR-conforming. The IUT "did not behave in conformance with CR."

These possibilities leave us with four possible (partial) interpretations of the verdict definition. Obviously, it should be clarified in the standard which of them is really meant, but let us try here to assess their relative merits. We will be able to identify four potential disadvantages of the various interpretations:

(1) The first condition of PASS, i.e. evidence of conformance, is redundant, because it is implied by the second, validity of events. This would cast some doubt on the interpretation considered, because the standard makers probably intended something else.

(2) The first condition of FAIL is implied by the second, cf. (1).

(3) In some cases, both PASS and FAIL apply, contrary to their characterization as mutually excluding alternatives. This would be highly counterintuitive.

(4) Inconclusive never applies, because either PASS or FAIL applies, or both do. This, too, seems not to be intended by the standard, which deals in several places with the verdict INCONCLUSIVE.

The last two seem to be of greater importance than the first two. A short analysis yields the following score for the four interpretations:

Interpretation:	(a) + (i)	(a) + (ii)	(b) + (i)	(b) + (ii)
Disadvantages:	(1), (3), (4)	(1), (2), (4)	(4)	(2)

Under the last interpretation, (b) + (ii), none of the more serious disadvantages (3) and (4) applies. It is therefore the only interpretation to be recommended here, pending further clarification in the standard.

Many experts therefore either agree silently on slightly other defininitions, e.g. replacing (a) by "the test purpose is achieved" – which delegates the meaning of verdicts to a clarifi-

cation of test purposes and their achievement – or call explicitly for a change of the standard, such as in [15]. Of course, even a widely accepted deviation from the standard is not a solution; rather, the standard has to be made acceptable. It seems hard to guess what the final definiton of verdicts will be, therefore we will leave this question open and only discuss some aspects of it. In 4.7 we will venture some recommendations on how to fix the verdict problem.

4.6. Test purposes and verdicts

Some experts identify test purposes (roughly) with global event sequences, or pieces thereof, some among them permit event sequences with invalid IUT behaviour as possible test purposes. What should the verdict be, if the IUT correctly declines to display the particular, or in fact any, invalid behaviour? The test purpose has not been obeyed, but the conformance requirement in the protocol has, such that the standardized criteria for PASS seem to be fulfilled.

PR13 Can a test purpose call for invalid IUT behaviour?

We think that an "objective, focussing on a conformance requirement," cannot be represented by an event sequence with invalid IUT behaviour, and that a previous analysis should be applied to exclude demands of invalid IUT behaviour from the test purposes. But this may just as well be another example of what kind of ambiguities 'natural' language can lead to ...

The most controversial point in the field of test purposes and verdicts is, in our opinion, whether test purposes may, or even should, require the IUT to pursue specific alternatives at points of dynamic choice in non-deterministic protocol specifications.

PR14 Can the IUT be obliged by the test purpose to choose one of several valid alternatives?

Different answers can obviously lead to different test purposes and, possibly, test verdicts: If the specification permits to choose between A and B, and if PR14 is answered with "yes" and the test purpose says "show A" and the IUT chooses B, then (assuming otherwise only valid and desired behaviour) many people feel that the verdict INCONC should result. At any rate, it seems not be advisable to demand the selection of uncontrollable exceptional alternatives. If PR14 is answered with "no" then the corresponding test purpose should rather say "show A or B", and the IUT choosing B should probably get a PASS. Test purpose and test suite writers do not answer this question uniformly, as pointed out in [15]. In fact, the authors cite cases where, in terms of our example, the IUT performing B would even be given a FAIL.

If PR14 is answered positively, then a FAIL does differentiate the reason:

PR15 If "PR14=YES" and the verdict is FAIL, did the IUT "refuse the test purpose", show invalid behaviour, or both?

A relatively independent problem with an impact on test purposes and verdicts arises from the notion of invalid PDU. Test suites dedicate a considerable portion of test cases to finding out whether the IUT displays the right kind of responses to invalid PDUs sent by the tester. PDUs of a type that should at the moment not be sent are commonly considered as invalid. But opinions differ about PDUs of the expected type, but containing a field value that a protocol-conforming tester would not produce at that moment.

PR16 Is a PDU of a valid type but with some invalid field value(s) invalid?

The test suite structure clause in part 2 of [1] seems to affirm the question, but some experts tend to negate this. Some protocols specify explicitly what to consider as invalid, and how to treat different kinds of invalid PDUs. A strict and clear, but not universally accepted, definition is to consider any PDU as invalid which could not have been produced (with exactly the observed field values) in the given situation by an entity conforming to its specification.

4.7. Possible improvements

The questions raised in this section have to be answered in CTMF standards, the sooner the better. PR9 would merit a CTMF part of its own. As to PR13, we are strongly inclined to answer "no." Regarding PR14, there seems to be room for a solution other than a mere yes or no: If we assume that dynamic choices in the protocol specification are either controllable and of equal value or uncontrollable and of the regular/exception type, then PR14 could be affirmed for controllable choices and negated for uncontrollable choices. The distinction between the types of choice need of course be made throughout the protocol standards. PR16 should either be answered individually in protocol specifications or collectively in CTMF.

PR11, PR12, the potential flaws (1-4) deriving from their answers, and PR15 can be resolved by always providing two test verdicts that separately answer the following questions:

1. Did the IUT exhibit only a valid behaviour sequence?
   (verdict suggestion: "VALIDITY: PASS/FAIL/INCONC")
2. Did the IUT fulfill the given test purpose?
   (verdict suggestion: "PURPOSE: PASS/ FAIL/ INCONC")

Furthermore, in case of INCONC it would be interesting to know whether the inability to decide was due to exceptional USP behaviour, or whether it is principally impossible to decide (e.g. if the medium is relatively "opaque", say, if we apply TTCN outside of OSI to some didactical example, or if the test purpose was ill-chosen).

Note the possible difference to present verdicts: If the test purpose is fulfilled and the behaviour "looks alright", the present verdict would be PASS. But the new validity verdict might be PASS or INCONC, depending on whether the same observations could have happened with an IUT committing errors. Of course, this proposal puts more burden on the test suite writers.

CONCLUSION

In this paper, we dealt with two important fields of CTMF which pose problems, partly by themselves, partly if held against general OSI principles or practical necessities: the PCO model and the test purpose/ test verdict complex. In either case, we identified and analysed several open questions and problems, and made specific suggestions to change existing standards.

Note that one could argue that the problems described in section 3 only provide a case against the Distributed method and the use of ASPs as test events. On the other hand, reverting e.g to the Remote method and PDUs as test events requires an equally thorough analysis as the one presented in this paper, probably with emergence of a similar number of problems. Moreover, it does not seem proper to give up useful test configurations only to avoid the clarification of some technical points.

Space in one paper is too limited to discuss all remaining problems in CTMF and TTCN. Some other areas in which we discern a demand for additional standardization or the need to improve existing standards, are outlined in [3].

Another crucial point that is, in our opinion, not yet satisfactorily treated is the missing formal definition of a correct test case. We hope that attempts to apply scores of different semantics, which might even multiply the need for different test suites, will be abandoned in favour of a single approach that realizes in theory that what is needed in practice. In particular, correct test cases must end after finitely many events and in finite time. Differences in infinity do not matter so much, whether they involve fairness considerations on infinite event sequences or notions of non-acceptance that can only be decided after infinite waiting.

It cannot be expected that all experts will agree with the views, arguments, and suggestions put forward in this analysis of CTMF and TTCN. Some readers may miss arguments

and criticisms they deem more important than those presented. In any case, we will be content if this paper can rekindle the discussion on TTCN, in order to clarify the meaning of test cases. In the end, test purpose and test suite writers, test laboratories, and test clients should profit from the results.

ACKNOWLEDGEMENTS

The author would like to express his gratitude to Heinz-Jürgen Burkhardt, Alfred Giessler, Christa Paule, Eckart Raubold, and Helmut Wiland, who helped in many discussions to work out the specific reasons for his initially more intuitive discontent with parts of CTMF.

REFERENCES

1. ISO/IEC IS 9646: Information Technology – Open Systems Interconnection – Conformance Testing Methodology and Framework, 5 parts, 1991/2
2. ISO/IEC IS 9646-3: Information Technology – Open Systems Interconnection – Conformance Testing Methodology and Framework, Part 3: The Tree and Tabular Combined Notation, 1992
3. B. Baumgarten, A. Giessler: OSI Conformance Testing Methodology and TTCN, to be published, Elsevier, 1994
4. A. Wiles: The Tree and Tabular Combined Notation – A Tutorial, Telia research, Uppsala, 1992
5. ISO/IEC DIS 7498-1: Information Technology – Open Systems Interconnection – Reference Model, Part 1: Basic Reference Model, 1992 (Revision of 1984's IS 7498)
6. ISO/IEC JTC 1, IS 10731: Information Technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services , 1994
7. C.A. Vissers, L. Logrippo: The Importance of the Service Concept in the Design of Data Communication Protocols, Protocol Specification, Testing, and Verification,V, North-Holland, 1986, 3-17
8. U. Bär: OSI-Konformitätstests: Validierung u. qualitative Bewertung, VDI-Verlag, 1994
9. B. Baumgarten: Structural Prerequisites for Unambiguous Conformance Testing, 2nd ISIIS, INTAP,Tokyo, 1988, pp 283-290
10. R. Gotzhein: On Conformance in the Context of Open Systems, 12th ICDCS, Yokohama, 1992, 236-243
11. M. Phalippou: The Limited Power of Testing, Protocol Test Systems, V , North-Holland, 1993, 43-54
12. J.G. Tretmans: A Formal Approach to Conformance Testing, CIP - Gegevens Koninklijke Bibliotheek, Den Haag, 1992
13. ISO/IEC IS 8072: Information Technology – Open Systems Interconnection – Transport Service Definition, 1986
14. G. v. Bochmann, C.S. He: Ferry approaches to protocol testing and service interfaces, 2nd ISIIS, INTAP,Tokyo, 1988, pp 283-290
15. S. T. Chanson, Qin Li: On Inconclusive Verdict in Conformance Testing, Protocol Test Systems, V , North-Holland, 1993, 81-92