RELATING INTEROPERABILITY TESTING WITH CONFORMANCE TESTING

Sungwon Kang
Korea Telecom Research & Development Group
Sochogu Umyundong 17, Seoul 137-792, Korea

Abstract

Conventionally the term *conformance testing* has been used to indiscriminately denote different types of conformance testing in a broad sense and the term *interoperability testing* has been used without regard to possibly different interoperability test architectures. This study classifies conformance testing into interface conformance testing and entity conformance testing and investigates how they can be related to interoperability testing as interoperability test architecture varies. It also discusses how the result of the investigation can be put into practical use for development of test suites and real test systems.

1. Introduction

It is universally agreed in the field of communication protocol testing that in addition to conformance testing, interoperability testing is essential to ensure functional correctness of communication systems. However, research on theory and practice of interoperability testing has begun only very recently and is scarce. As early work on interoperability testing, there are [3][4][9]. In [8], the approach was refined and generalized to handle non-determinism more efficiently. [7] develops efficient interoperability test derivation method for symmetric communication protocols. As theoretical work, [6] study the notion of interoperability and its relation to conformance.

However, there is no work studying various test architectures for conformance and interoperability testing and relation between conformance and interoperability testing based on the test architectures. In particular as the result of this situation, often in practice of protocol testing people had difficulty in matching the intended kind of testing (i.e. conformance testing or interoperability testing), the intended test coverage (such as complete testing or partial testing) and the test architecture to be applied. Also often decisions on the choice of test architecture and test coverage were not adequately made depending on availability of other test suites.

In this paper, we classify conformance testing into interface conformance testing and entity conformance testing and studies how they can be related to interoperability testing as the testing capability of interoperability test architecture varies. We also discuss ways of applying the result of the investigation for developing test systems and making decisions on appropriate test architectures and test coverage.

The remainder of the paper is organized as follows: In Section 2, the notions of interface and entity are defined and compared. In Sections 3 and 4, concepts and test architectures of conformance testing and interoperability testing, respectively, are explained. Conformance testing and interoperability testing are related in Section 5 and the result is summarized in Section 6. Finally, Section 7 discusses its application.

2. Interface vs. Entity

An *entity* is a unit that acts in itself or by receiving stimuli from environment and has one or more (communication) interfaces. An *interface* is a window through which entities interact with each other.[1] Figure 1 depicts a situation where entities are interconnected with each other.[2]

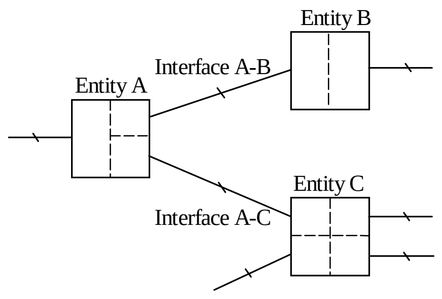


Figure 1. Interface vs. Entity

Whereas current protocol standards tend to define only interfaces, actual implementations exist as entities. In order to know a complete behavior of an entity, one has to know behavior at all the relevant interfaces of it together with interrelation between the behavior at the interfaces.[3]

In this paper, we consider only entities with two interfaces.

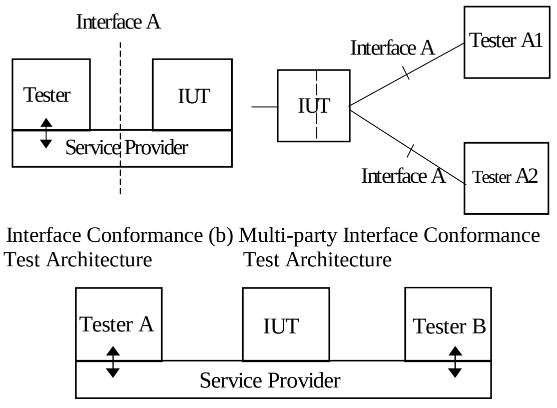
[1] An entity without interfaces exists in isolation from the rest of the world and hence is not interesting to us.
[2] Figure 1 can also be considered a view that abstracts from underlying layers.
[3] This is usually missing in current standardized protocol specifications. However, so far no report has been made that a serious problem has been caused by that.

Table 1. Comparison of Interface and Entity Conformance Testing

	Interface Conformance Testing	Entity Conformance Testing
Characteristics	- Interface description explicit in standardized specification (usually as a pair of sides or directions) - Interface tested direction by direction	- Entity description implicit in standardized specification - Entity tested at once through all interfaces
Advantages	- Easier to develop standardized test suites (small number of interfaces in contrast with potentially many different kinds of entities)	- Higher test coverage (certain necessary stimulus may not be available when interfaces are tested separately)

3. Conformance Testing

In this section, we consider the problem of verifying correctness of interface and entity by testing.[4] Conventionally, the term *conformance testing* has been used vaguely to denote *testing for a single implementation*. Both testing of an interface and testing of an entity of a single implementation were referred to as conformance testing. In this paper, we call them more precisely *interface conformance testing* and *entity conformance testing*, respectively.



(a) Interface Conformance Test Architecture (b) Multi-party Interface Conformance Test Architecture

(b) Entity Conformance Test Architecture

Figure 2. Conformance Test Architectures

Figure 2 shows test architectures for interface and entity conformance testing. The arrow as in Figure 2(a) indicates functionality of testers. So a double-headed arrow represents a *Point of Control and Observation* (PCO) and indicates that the tester has both observability and controllability. A single-headed arrow as in Figure 3(b) represents a *Point of Observation* (PO) and indicates that the tester has only observability. Such a tester is commonly called *monitor*.[5]

The architecture of Figure 2(b) depicts *multi-party interface conformance testing*. It is important to note that it is the number of different kinds of interfaces a system of testers is dealing with rather than just the number of testers that differentiates entity conformance testing and interface conformance testing. Entity conformance testing can likewise be *multi-party entity conformance testing* by having more than one testers on one interface or on both interfaces. Test architecture determines the nature of the derived test suites and hence selecting test architecture should be done very carefully.

Conformance testing checks for each transition of a specification Finite State Machine (FSM) operation errors, i.e. whether it is correct with respect to the input/output behavior, and transfer errors, i.e. whether it is in the expected state after the transition, in an Implementation Under Test (IUT) [5]. For this purpose, a *conformance test case* consists of preamble, test body and postamble. *Test body* is the part of a conformance test case that checks operation errors and transfer errors. Test body ends by reaching a stable state. *Preamble* is the part that takes M to the stable state at which test body can begin. *Postamble* is the part that takes M to a stable state and is often empty if IUT would be already in a stable state after executing test body. A set of test cases is called a *test suite*. A conformance test case is obtained from a sequence of actions of an entity specified in its specification that begins at a stable state and ends at a stable state.[6] The expected sequence of actions that constitutes the testing target of a test body is called a *test focus*. As the length of test focus gets shorter, the analytic power of the test case becomes greater. A finest test focus is obtained by choosing test body such that it begins with a stable state and ends with the immediately following stable state.[7]

The fact that implementations exist as entities makes it difficult to test one interface while ignoring the other interfaces. By considering not only the interface on focus but also the other interfaces as well, often we can perform more complete testing. On the other hand, it is costly to deal with the interfaces that are not on focus.

[4] Although specification commonly exists as an interface specification, implementation exists as an entity. Therefore, an interface as a target of testing is an interface as part of entity implementation.

[5] A tester with controllability only could be represented with a single-headed arrow pointing in the opposite direction from that of monitor.

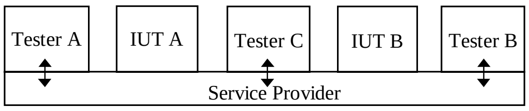
[6] If only interface specifications exist, stable states of an entity must be inferred.

[7] This is essentially 'Single Stimulus Principle' used in [3].

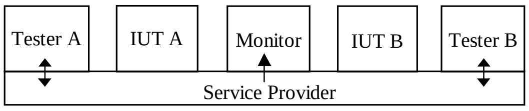
Also one interface may combine with various interfaces to make entities with different functionalities. And it would be hard to get standardized test suites for entities since there would be many combinations of interfaces. In relation to conformance testing, these observations can be summarized as in Table 1.

4. Interoperability Testing

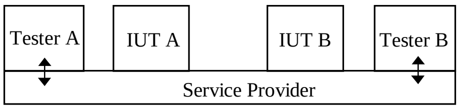
The essential differences of *interoperability testing* from conformance testing are (1) that the target of testing is two or more entities as a whole or as a system rather than a single entity and (2) that the behavior to be expected for testing should be inferred from the respective specifications of entities. In this section, we show interoperability test architectures for the system consisting of two entities and the structures of interoperability test cases under the test architectures.[8] Depending on how completely we want to observe the behavior of a system, various *interoperability test architectures*[9] are possible as shown in Figure 3.



(a) Interoperability Test Architecture I



(b) Interoperability Test Architecture II



(c) Interoperability Test Architecture III

Figure 3. Various Interoperability Test Architectures

To examine the consequences of test architectures on interoperability testing and on the resulting interoperability test cases structures, we use the typical call establishment procedure of signaling protocol in Figure 4 as in [1][2]. The interface between *terminal equipment* (TE) and IUT is called *User-Network Interface* (UNI) and the interface between IUT's is called *Network-Network Interface* (NNI). In Figure 4, messages prefixed with '*i*_' and with no such prefix are two different kinds of messages, the latter belonging to UNI and the former belonging to NNI.

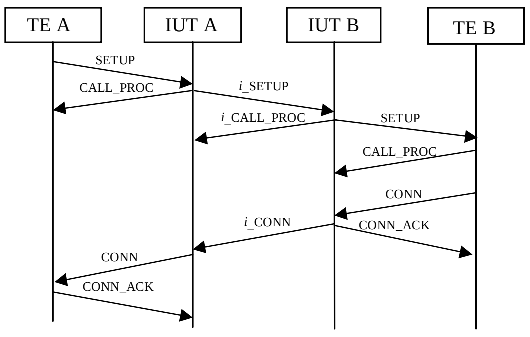
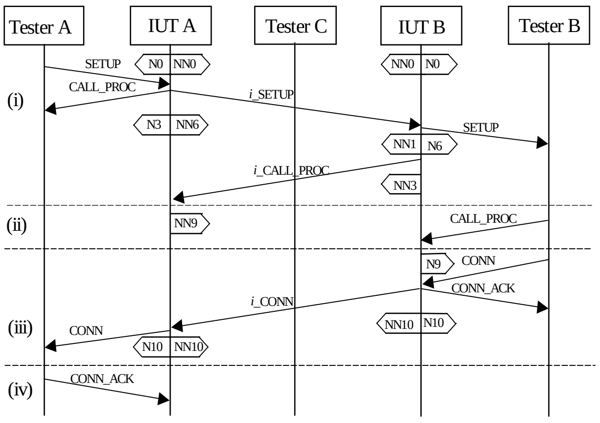
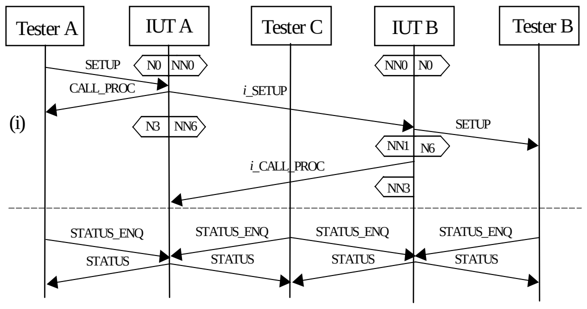


Figure 4. Call Establishment Interaction

With this example we show how the same interaction sequence would look differently as the interoperability test architecture varies from I through II to III.



(a) The View



(b) IOP-TC-I: An Interoperability Test Case for (i)

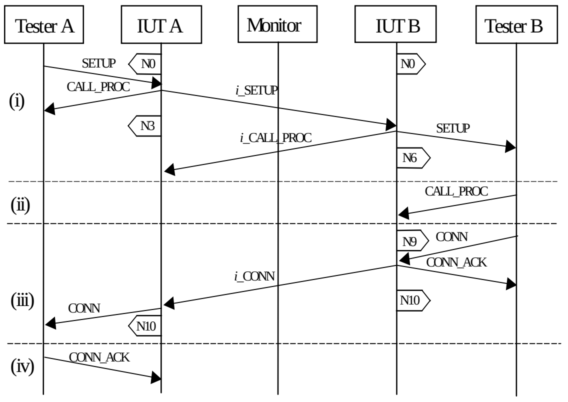
Figure 5. View under Test Architecture I

Figure 5(a) shows how the interaction sequence in Figure 4 would look under Interoperability Test Architecture I. For the interaction sequence, four test focuses (i.e. (i)-(iv)) are obtained. A stable global state and its immediately following stable state separates out each of the test cases. Two of them ((i) and (iii)) are interoperability test cases since two IUT's are involved in the interaction and two of them ((ii) and (iv)) are (indeed, entity) conformance test cases since only one IUT is involved in the interaction. Actual test cases

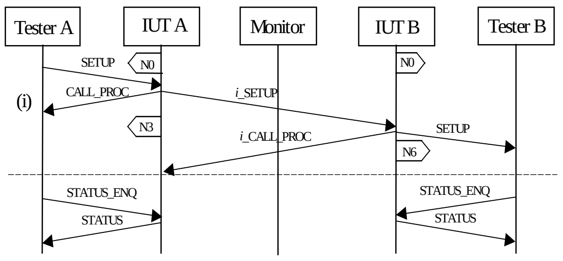
[8] Even if one may have the view that interoperability is a matter of behavior at the interface between two implementations under test, other interfaces (at the far sides) cannot be completely ignored for interoperability testing. Therefore complete interoperability testing can be achieved only by considering whole entities.

[9] The reason we used the term 'interoperability test architecture' here instead of the more precise 'system test architecture' is that *the* prominent aspect of system test architecture is interoperability testing. For conformance testing, it would be more cost-effective to use simpler test architectures shown in Figure 2.

would differ from the normal interaction sequence in that they contain state verification steps. An example is shown in Figure 5(b) as IOP-TC-I. Given the test architecture, not only the states at the UNI side but also the states at the NNI side should be examined to check correctness thoroughly.

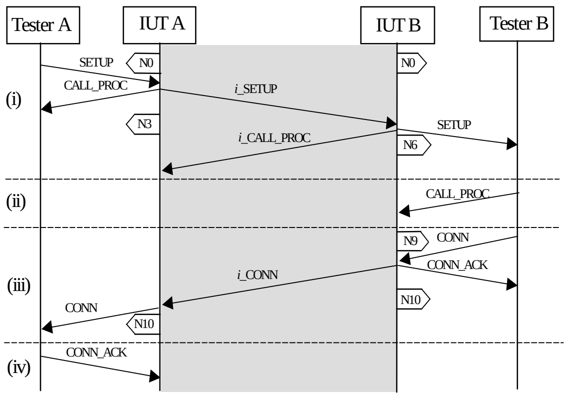


(a) The View

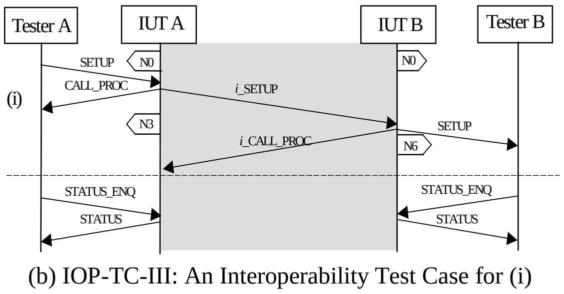


(b) IOP-TC-II: An Interoperability Test Case for (i)

Figure 6. View under Test Architecture II



(a) The View



(b) IOP-TC-III: An Interoperability Test Case for (i)

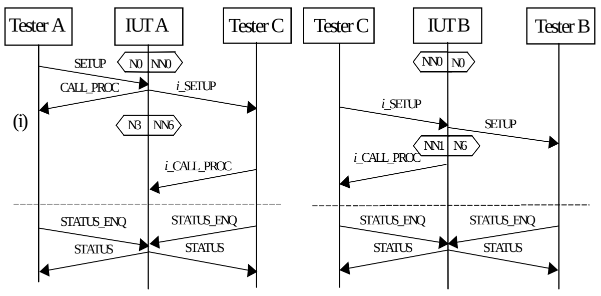
Figure 7. View under Test Architecture III

When Interoperability Test Architecture II is chosen as in Figure 6, messages cannot be artificially inserted or observed at NNI. Therefore, status inquiry cannot be performed at NNI and only comparatively little information about NNI can be obtained.

Interoperability Test Architecture III provides the most abstract perspective of interoperability testing among the three test architectures. Now the messages back and forth at NNI cannot be observed at all as implied by the shaded area in Figure 7. Of course, no state verification can be performed at NNI, either. When setting up a tester between two IUT's is infeasible, this is the only choice for interoperability test architecture.

5. Relating Conformance Testing and Interoperability Testing

In the previous sections, we examined various conformance and interoperability test architectures and the consequences and implications of test architectures on the structures of test cases. We now investigate how various conformance testing types and interoperability testing types can be related with each other.



(a) EC-TC-I-A (b) EC-TC-I-B

Figure 8. Dissection of IOP-TC-I into Conformance Test Cases

Figure 8 shows that when IOP-TC-I is applied the effect is in essence as if both EC-TC-I-A an EC-TC-I-B were applied. Conversely, one can say that the effect of applying two conformance test cases EC-TC-I-A and EC-TC-I-B can be achieved with IOP-TC-I under Test Architecture I.[10] [11]

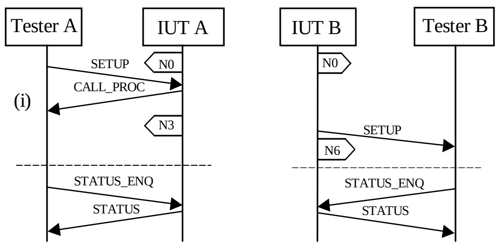
In contrast, there is no tester between IUT A and IUT B under Test Architecture III. However, as shown in Figure 9, IOP-TC-III can be divided into two conformance test cases.[12] But now these test cases are

[10] Such dissection of an interoperability into conformance test cases is possible because Tester C not only can observe messages but also can send messages, thereby enabling examination of the NNI side of IUT A.

[11] Statements in the previous paragraph hold because we only consider messages types ignoring the internal contents of messages. For example, Tester C cannot in general anticipate the *i*_SETUP message that IUT A sends since different implementations would send different *i*_SETUP messages with different contents.

[12] In contrast with Test Architecture I, in this case we cannot say that conjunction of IC-TC-III-A and IC-TC-III-B

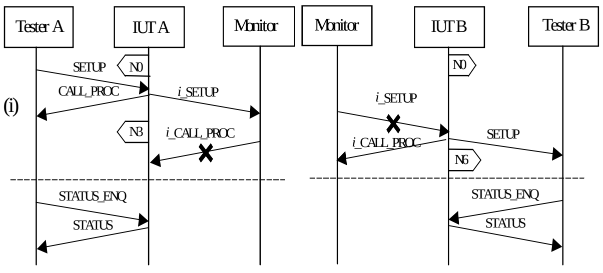
not entity test cases but interface test cases. In the case of IC-TC-A, it is straight to see that it is a conformance test case. In the case of IC-TC-B, IUT B may or may not be able to send SETUP. If NNI message were to be used, then since NNI messages cannot be sent under Test Architecture III, we must rely on a test architecture like Test Architecture I.[13] However, in order to have IUT B send SETUP, non-NNI messages such as UNI message may be employed by having an additional tester at the UNI side.



(a) IC-TC-III-A (b) IC-TC-III-B

Figure 9. Dissection of IOP-TC-III into Conformance Test Cases

In this way, IOP-TC-I and IOP-TC-III can be divided into two pairs of different kinds of conformance test cases. That is, a single interoperability test case has the effect of two conformance test cases. In the case of IOP-TC-II, however, the interoperability interaction cannot be cleanly divided into two conformance test cases.[14] As shown in Figure 10, C-TC-II-B is neither an entity conformance test case nor an interface conformance test case. The arrow with cross on it indicates the message is unrealizable with Test Architecture II. Under this architecture, we cannot have IUT B send SETUP using *i*_SETUP. For using *i*_SETUP implies using NNI and for this we need entity conformance test architecture, which can be obtained from Test Architecture I only.



(a) C-TC-A (b) C-TC-B

Figure 10. Dissection of IOP-TC-II

6. Comparing Interoperability Testing

Let us denote application of a test suite to a system under test as:

$$exec\text{: set of test suites} \times \text{set of systems under test} \rightarrow \{\text{PASS, FAIL}\}$$

If '$exec(TS_2,SUT)$ = FAIL' implies '$exec(TS_1,SUT)$ = FAIL' for consistent test suites[15] TS_1 and TS_2 and an arbitrary system under test SUT, let us denote it as $TS_2 \leq_{TC} TS_1$ read *the test coverage of* TS_2 *is less than or equal to the test coverage of* TS_1. Also if $TS_1 \leq_{TC} TS_2$ and $TS_2 \leq_{TC} TS_1$, then let us write it as $TS_1 \approx_{TC} TS_2$ read TS_1 *and* TS_2 *have the same test coverage*.

If interface conformance testing is denoted as $\mathcal{CT}_{\mathcal{I}}$, entity conformance testing as $\mathcal{CT}_{\mathcal{E}}$ and all the interfaces of S are $S_{\mathcal{I},1}$ and $S_{\mathcal{I},2}$, then $\mathcal{CT}_{\mathcal{I}}(S_{\mathcal{I},1}) + \mathcal{CT}_{\mathcal{I}}(S_{\mathcal{I},2}) \leq_{TC} \mathcal{CT}_{\mathcal{E}}(S)$. The reason is, as explained in Section 2, entity testing views behavior of interfaces as a whole whereas interface testing does not. Now if the difference between $\mathcal{CT}_{\mathcal{I}}(S_{\mathcal{I},1}) + \mathcal{CT}_{\mathcal{I}}(S_{\mathcal{I},2})$ and $\mathcal{CT}_{\mathcal{E}}(S)$ is denoted as ε_S, then

$$\mathcal{CT}_{\mathcal{E}}(S) \approx_{TC} \mathcal{CT}_{\mathcal{I}}(S_{\mathcal{I},1}) + \mathcal{CT}_{\mathcal{I}}(S_{\mathcal{I},2}) + \varepsilon_S \qquad (*)^{16}$$

In Section 5, we investigated how interoperability testing can be related to conformance testing. The result can be summarized with the help of these notations as in Table 2. In Table 2, α denotes the set of test cases that involve only one IUT that are obtained from naturally arising interaction of two IUT's and thus are excluded from the set of (pure) interoperability test cases. Then for $\mathcal{IOPT}_i(A,B)$ with i = I, II, III, $\mathcal{IOPT}_i(A,B) + \alpha$ can be regarded as a *system test suite* where the system under test consists of IUT's A and B. β denotes the set of test cases to be initiated by Tester C. Such test cases cannot be obtained from natural interaction sequences not involving Tester C and thus cannot belong to $\mathcal{IOPT}_i(A,B)$. Then the entry in the first column of the first row of Table 2 reads "Given Interoperability Test Architecture I, the system test suite consists of interoperability test cases for IUT A and IUT B together with conformance test cases for IUT A and IUT B, respectively, which are equivalent to the set of entity conformance test cases for IUT A and IUT B together with the test cases of β."

The underlying assumption in stating with (in)equalities in the table is that the effect of testing a whole (on the left-hand side of (in)equalities) can be achieved with piece-by-piece testing (on the right-hand-side). It is true that in some cases local correctness does not imply global correctness. As is suggested with (*) above, we have reason to prefer one entity conformance testing to two interface conformance testings.[17] Similarly, there is more to interoperability testing than what conformance testing can offer.

constitutes IOP-TC-III. For if because of a fault in IUT a wrong message is sent at NNI then it would not be observed and hence would not be reflected in the test case verdict.

[13] One with experience in test suite development would have no difficulty in seeing that a new interface means new PCO(s) and hence major changes requiring different test architecture.

[14] The reason is that a monitor by definition cannot generate messages. The attempted dissection is depicted in Figure 10.

[15] That is, test suites that do not give FAIL verdict to systems with no fault.

[16] Equation clearly shows what difference exists between the thing on the left-hand side and the thing on the right-hand side.

[17] However, piece-by-piece testing has the advantage of localizing the cause of failure.

Table 2. Relation between Various Conformance Testing Types and Interoperability Testing Types

	Entity Conformance Testing $\mathcal{CT}_{\mathcal{E}}$	Interface Conformance Testing $\mathcal{CT}_{\mathcal{I}}$
Interoperability Testing I $\mathcal{IOPT}_{\mathrm{I}}$	$(\mathcal{IOPT}_{\mathrm{I}}(\mathrm{A,B}) + \alpha) + \beta \approx_{\mathrm{TC}}$ $\mathcal{CT}_{\mathcal{E}}(\mathrm{A}) + \mathcal{CT}_{\mathcal{E}}(\mathrm{B})$	$(\mathcal{IOPT}_{\mathrm{I}}(\mathrm{A,B}) + \alpha) + \beta \approx_{\mathrm{TC}}$ $(\mathcal{CT}_{\mathcal{I}}(\mathrm{A}_{\mathcal{X},1}) + \mathcal{CT}_{\mathcal{I}}(\mathrm{A}_{\mathcal{X},2}) + \varepsilon_{\mathrm{A}}) + (\mathcal{CT}_{\mathcal{I}}(\mathrm{B}_{\mathcal{X},1}) + \mathcal{CT}_{\mathcal{I}}(\mathrm{B}_{\mathcal{X},2}) + \varepsilon_{\mathrm{B}})$
Interoperability Testing *i* $\mathcal{IOPT}_{i}$ where i = II, III	$\mathcal{IOPT}_{i}\,(\mathrm{A,B}) + \alpha \leq_{\mathrm{TC}}$ $\mathcal{CT}_{\mathcal{E}}(\mathrm{A}) + \mathcal{CT}_{\mathcal{E}}(\mathrm{B})$	$\mathcal{CT}_{\mathcal{I}}(\mathrm{A}_{\mathcal{X},1}) + \mathcal{CT}_{\mathcal{I}}(\mathrm{B}_{\mathcal{X},2}) \leq_{\mathrm{TC}}$ $\mathcal{IOPT}_{i}(\mathrm{A,B}) + \alpha \leq_{\mathrm{TC}}$ $(\mathcal{CT}_{\mathcal{I}}(\mathrm{A}_{\mathcal{X},1}) + \mathcal{CT}_{\mathcal{I}}(\mathrm{A}_{\mathcal{X},2}) + \varepsilon_{\mathrm{A}}) + (\mathcal{CT}_{\mathcal{I}}(\mathrm{B}_{\mathcal{X},1}) + \mathcal{CT}_{\mathcal{I}}(\mathrm{B}_{\mathcal{X},2}) + \varepsilon_{\mathrm{B}})$

7. Conclusion

Relating interoperability testing with conformance testing as studied in this paper can guide us in making important decisions for test suite and test system development. Commonly, when choosing interoperability test architectures, we need to balance expected test coverage and availability of conformance test suites and available number of PCO's. From the result of this paper, it is clear that availability of conformance test suites must be considered in choosing interoperability test architectures. So, for example, if both UNI and NNI conformance test suites are available, then it is wise to choose $\mathcal{IOPT}_{\mathrm{III}}$(A,B) instead of the much heavier $\mathcal{IOPT}_{\mathrm{I}}$(A,B). Under Interoperability Test Architecture III, UNI conformance testing, NNI conformance testing and interoperability testing can all be accommodated in a test system with two PCO's as far as a single-party testing is concerned. If none of UNI and NNI conformance test suites are available, then it is preferable to have $\mathcal{IOPT}_{\mathrm{I}}$(A,B), which would allow us to examine both UNI and NNI to some extent. If UNI conformance test suite exists but NNI conformance test suite does not, then either $\mathcal{IOPT}_{\mathrm{I}}$(A,B) or $\mathcal{IOPT}_{\mathrm{II}}$(A,B) would be a good choice. For certainly we would not want to completely by-pass looking into NNI for checking system functionality. Interoperability Test Architecture III has the advantage that when NNI specification changes, the interoperability test suite is applicable with minimal change. In more detail, only the part related to NNI need to be added and/or changed.

For practice of testing, we also need to consider that PCO is an expensive component of a test system. When the number of available PCO's is limited, we may be compelled to choose less powerful interoperability test architectures. For example, putting a monitor between two IUT's means having two additional PCO's (one for each direction of message flow) in the test system. It also helps to remember that the more PCO's the test architecture requires, the more difficult it becomes to develop a test suite. This follows from that when messages movement is observed, we cannot avoid basing our judgement of correctness on the observed behavior[18] and thus monitors should be able to expect correct messages and incorrect messages (even in the case of automatic testing).

Acknowledgment The author would like to thank his colleagues at Protocol Testing Division for having discussions with him and making helpful suggestions on the subject of this paper.

8. References

[1] ATM Forum, *ATM User-Network Interface Specification, Version 3.1*, 1994.
[2] ATM Forum, *ATM Forum PNNI Draft Specification*, 1996.
[3] Arakawa, N., and Soneoka, T., "A Test Case Generation Method for Concurrent Programs", *Protocol Test Systems, IV*, J. Kroon, R.J. Heijink and E. Brinksma (Eds.), Elsevier Science Publishers B. V., 1992.
[4] Arakawa, N., Phalippou, M., Risser, N., and Soneoka, T., "Combination of conformance and interoperability testing", *Formal Description Techniques, V (C-10)*, M. Diaz and R. Groz (Eds.), Elsevier Science Publishers B. V., 1993.
[5] Chow, T. S., "Testing Software Design Modeled by Finite-State Machines", *IEEE Trans. on SE, Vol. SE-4, No. 3*, May 1978.
[6] Kang, S., "Interoperability and Conformance Relations of Communicating Systems", *Journal of the Korean Information Science Society* (A): *Computer Systems and Theory*, Vol. 24, No. 12, Dec. 1997.
[7] Kang, S., and Kim, M., "Interoperability Test Suite Derivation for Symmetric Communication Protocols", *Formal Description Techniques and Protocol Specification Testing and Verification Forte X / PSTV XVII '97*, Osaka, Japan, Nov. 1997.
[8] Kang, S., and Kim, M., "Test Sequence Generation for Adaptive Interoperability Testing", *Protocol Test Systems VIII*, Chapman & Hall, 1996.
[9] Luo, G., Bochmann, G., and Petrenko, A., "Test Selection Based on Communicating Nondeterministic Finite-State Machines Using a Generalized Wp-Method", *IEEE Trans. on S.E., Vol. SE-20, No. 2*, Feb. 1994.

[18] Verdict is a mechanism that warns us of presence of a problem. When there is no warning, the test operator would not bother to analyze the test result with the consequence that the problem remains undetected.