An Iterative Procedure for Efficient Testing of B2B: A Case in Messaging Service Tests

Jaewook Kim and Serm Kulvatunyou
Manufacturing Systems Integration Division, National Institute of Standards & Technology, Gaithersburg, MD 20899-8263, USA
jaewook@nist.gov and serm@nist.gov

Keywords: conformance and interoperability testing, B2B messaging service, interoperability of E-Business solutions, tools for interoperability

Abstract

Testing is a necessary step in systems integration. Testing in the context of inter-enterprise, business-to-business (B2B) integration is more difficult and expensive than intra-enterprise integration. Traditionally, the difficulty is alleviated by conducting the testing in two stages: conformance testing and then interoperability testing. In conformance testing, systems are tested independently against a referenced system. In interoperability testing, they are tested simultaneously against one another. In the traditional approach for testing, these two stages are performed sequentially with little feedback between them. In addition, test results and test traces are left only to human analysis or even discarded if the solution passes the test. This paper proposes an approach where test results and traces from both the conformance and interoperability tests are analyzed for potential interoperability issues; conformance test cases are then derived from the analysis. The result is that more interoperability issues can be resolved in the lower-cost conformance testing mode; consequently, time and cost required for achieving interoperable solutions are reduced.

1 Introduction

Although common B2B standards have been developed, interoperability cannot be achieved without testing. This is because (1) the software systems involved in the

inter-enterprise integration vary greatly and are implemented with various user-specific assumptions, and (2) business terms used in the standards do not always have precise meanings, which are left open for different interpretations by systems implementers.

Given these difficulties, integration testing often requires collaborations among experts from different locations and time zones. It is a difficult, costly, and time-consuming activity. These experts have adopted a two-stage, sequential approach: Conformance then Interoperability Tests (CIT). First, they perform the conformance tests independently against a reference implementation or a test data suite. After conformance is verified, they then perform pair wise testing of the two solutions to resolve interoperability problems. The expectation is that most interoperability issues are resolved in the conformance testing, which is less expensive and easier to perform [1].

The sequential CIT has two challenges. First, conformance and interoperability tests require different testing tools, suites, and methodologies. Second, a significant amount of time is still required to achieve interoperability because both the conformance and interoperability test suites are static and cannot account for all interoperability issues. There are existing research works that deal with the first challenge including [2], [3], and [4], but little has been done to address the latter challenge.

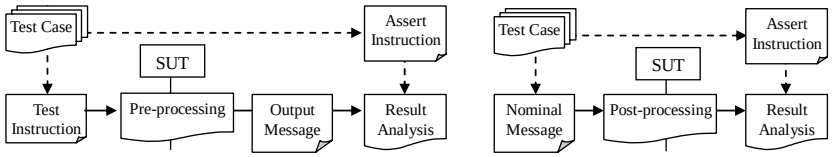
In this paper, we discuss a procedure to increase the capabilities of conformance testing to reduce the time and cost of interoperability testing. In the traditional two-stage testing there is little or no feedback between the conformance and interoperability tests. That means the interoperability issues presented in the interoperability testing mode are resolved only in that mode. Resolving issues in the interoperability mode is difficult and costly. The reasons are two-fold. The interoperability testing first requires engineers to be physically present at both ends of the test, and second, one system cannot step through (or control) the other system (e.g., to pause, restart, or make changes). In addition, test results (for both conformance and interoperability tests) and traces are left only to human analysis or even ignored when the test evaluation is non-negative.

2 Background

In this research, we characterize interoperability issues into `anticipated', `potential', and `unanticipated' issues. In general, conformance test suites are written mainly for anticipated issues - standard specification experts write test cases for those issues they can anticipate. The subsequent interoperability tests are performed typically on those same issues, but the issues lead to potential issues which are not exposed. This is frequently not sufficient to guarantee interoperability, because there are always unanticipated issues. Consequently, we characterize the interoperability testing as the process to find and resolve issues not covered by the conformance test suite.

To analyze the traditional two-stage testing approach, we follow the approach in [7], where the authors classify the type of conformance test cases according to a conformance test harness, which can be derived from the test framework.

Generally, the functionality of the Systems Under Test (SUT) can be viewed as either a pre-processing or post-processing [2]. In pre-processing mode, the system generates a message conforming to a standard specification; in the post-processing mode the system interprets a standard message. Typically, a pair of reciprocal, conformance test cases is written against the standard message structure to test both [5]. Figure 1 illustrates the conformance testing processes for the two functionalities. Every SUT should be verified by the conformance testing with these pairs of test cases before integrating with another partner SUT.



a. pre-processing conformance test b. post-processing conformance test

Fig. 1. Conformance testing processes for pre-processing and post-processing

An interoperability test is defined with respect to a pair of SUTs. The test typically requires one SUT to produce a standard message, which is used as input to another SUT. Figure 2 shows the interoperability testing process. From Figure 2, we conclude that an interoperability test case can be represented by a pair of pre- and post-processing conformance test cases. That is, we can decompose each interoperability test case into two independent conformance test cases [3]. Consequently, we believe that testing issues from the interoperability test cases can be resolved by simulating localized conformance test cases.

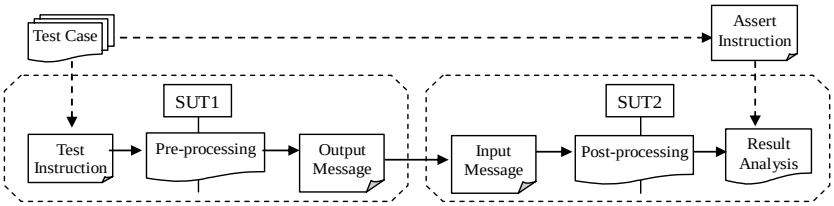


Fig. 2. Interoperability testing

The RFC 2119 [6] standard classifies conformance test cases on three levels: required, recommended, and optional. Every software solution must pass all required test cases, and may selectively pass the other two. Recommended test cases are called conditional-mandatory. This means that, in general, software solutions must pass those test cases, but developers may ignore them in particular and well-controlled circumstances. Optional test cases may be ignored [7]. These flexible requirements complicate interoperability testing and lead to the potential and unanticipated issues described above.

In this paper, we propose a test procedure, namely, the Iterative Test Procedure (ITP). The procedure employs two notions, (1) detecting as many as possible unanticipated issues before performing the interoperability test and resolving them

in the conformance testing mode; and (2) resolving remaining interoperability using a conformance testing approach. The ITP achieves these two objectives by analyzing test results and traces.

3 Related Works

The general approach to interoperability testing is to apply a conformance testing approach that individually checks software solutions against relevant standards. The approach for conformance testing has been standardized by the International Organization for Standardization (ISO) [8] and International Telecommunication Union Telecommunication Standardization Sector (ITU-T) [9]. This approach reduces testing costs by resolving interoperability issues during conformance testing, which is less expensive. Numerous researchers [7], [10], [11], and [12] have proposed architectures for implementing this approach. However, as noted in [13], this approach cannot resolve all interoperability issues. Thus the expensive interoperability testing is still needed.

To reduce the interoperability testing cost, a number of researchers developed efficient test-suite generation techniques. For example, [14], [15], and [16] provide methodologies or tools for deriving test suites automatically and [17] provides efficient algorithms to minimize the number of test cases. However, this research has only dealt with the static test suites generated from the original specification. None of these researchers has considered constructing test cases dynamically from the results of other tests.

In the architecture area, [3] proposed an approach to choose and apply an effective interoperability test harness based on both the type of conformance test suite and the relationship between the conformance and interoperability test architectures. In addition, [2] proposes an architecture that analyzes the interoperability problems by applying the conformance tool and test suite in the interoperability test harness. The former approach only helps to choose a proper interoperability harness. The latter approach practically integrates the conformance test assertions into the interoperability test harness. Neither of the approaches helps reduce the number of interoperability trials.

4 Proposed Iterative Test Procedure

The ITP intelligently anticipates interoperability issues and generates conformance test cases from conformance and interoperability test results and traces. The procedure is depicted in Figure 3. Two intelligent modules, a Conformance Test Results Analyzer (CTRA) and an Interoperability Test Results Analyzer (ITRA), are added to the traditional CIT procedure. We note that the procedure in Figure 3 does not require that SUT1 and SUT2 be concurrently present except in the interoperability testing.

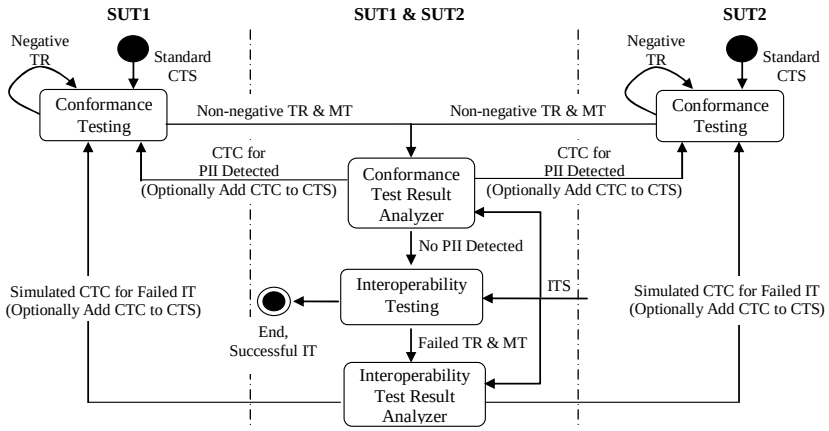


Fig. 3. Proposed iterative test procedure (CTS: conformance test suite, ITS: interoperability test suite, CTC: conformance test case, TR: test result, MT: message trace, IT: interoperability test, and PII: potential interoperability issue)

The purpose of the CTRA is to detect potential and unanticipated issues from two conformance test results. SUTs should conform to the conformance test suite before the CTRA analyzes their test results (as indicated by the retry loop associated with the negative test results in Figure 3). We note that conformance to a test suite does not necessarily mean that the SUT passes all test cases. In other words, the CTRA only analyzes non-negative conformance test results where results associated with all required test cases are `pass', while the recommended and optional test cases are either `undetermined' or `fail'. If any issues are detected, the CTRA recommends and generates corresponding conformance test cases. Each SUT is then checked against those test cases before proceeding to perform the interoperability test. The CTRA also recommends moving unanticipated issues into the status of anticipated issues by adding the corresponding test cases to the conformance test suite.

As described earlier in Section 2, the role of the interoperability test is to find unanticipated issues, which typically are induced by a specific interoperability context such as specific system configurations. The ITRA analyzes failed test results and traces from the interoperability test and simulates conformance test scenarios so that SUTs can independently find and resolve the problem. In Section 5, we describe the CTRA and ITRA in detail.

5 Test Results Analyzers

5.1 Conformance Test Results Analyzer

As we mentioned in Section 2, an interoperability test case may be a derivative of a pair of reciprocal pre- and post-processing test cases. For example, Figure 2 can be

viewed as a superimposition of Figure 1a and Figure 1b. The CTRA pays attention to this type of conformance test cases. Ideally speaking, if SUT1 and SUT2 pass the pre-processing and the post-processing test cases in the conformance testing mode, they will pass the respective interoperability test case. However, the following scenarios can happen.

First, the pre- and post-processing test cases may be optional. If an SUT skips the tests or ignore the test results, it will most likely fail the corresponding interoperability test cases. Second, SUT1 may produce a peculiar message (such as the output message in Figure 2), which, though still valid, is different from the nominal message (the nominal message in Figure 1b) parametrically defined by the conformance test suite. The SUT2 may not be able to process the peculiar message from the SUT1. Third, it is not possible that the conformance test suite will cover all cases. The SUT may make some assumptions that are not accounted for in the conformance test suite and/or the conformance test case verification.

In the first scenario, the CTRA identifies interoperability issues using conformance test results from both SUTs and a target interoperability test suite. If the interoperability test suite contains any test case that corresponds to a pair of optional pre- or post-processing test cases that either or both SUTs have not passed, the CTRA recommends that the SUTs perform these conformance tests first.

For the second and third scenarios, the CTRA identifies interoperability issues from the trace of output messages including the nominal message in the pre-processing test case and the actual message output from the SUTs. Even if the actual message conforms to the standard specification, if there is any discrepancy between the nominal message and the actual message, the CTRA constructs new, conformance test cases. If the SUT1 produces the same message as the nominal message of Figure 1b or the SUT2 is tested by a post-processing test case including the output message of the SUT1 in Figure 1a before this interoperability test scenario, they will more likely pass the derived interoperability test-case scenario. It is important to note that when the CTRA compares messages, it intelligently disregards the areas of the message where the contents are parameterized.

Thus, we summarize a set of rules the CTRA uses to recommend conformance test cases.

Rule 1 CTRA recommends pre- and/or post-processing test cases where the result is `undetermined' or `fail' if an interoperability test case derived from the corresponding optional pre- and post-processing test cases is included in the interoperability test suite.

Rule 2 For each pair of `pass' pre- and post-processing tests, from which the interoperability test case in the interoperability test suite is derived, compare the pre-processing's actual output message with the nominal message in the post-processing test case. If they are not identical,

Rule 2.1 Generate a new post-processing test case using the pre-processing's actual output message and recommend the post-processing system to pass it, or

Rule 2.2 Recommend that the pre-processing system produce the same output message as the nominal message.

Rule 3 CTRA recommends the expert of the target specification to analyze the conformance test cases generated from RULE 2 and updates the conformance test suite according to the generality and consistency of those test cases. This can help increase future interoperability efficiency by extending the coverage of the conformance test suite.

5.2 Interoperability Test Results Analyzer

The ITRA provides an efficient mechanism to detect and correct issues in the failed interoperability tests. The correlation between SUTs' interactions typically makes it difficult to analyze test results and resolve the failure in the interoperability testing mode. To reduce these correlations, the ITRA constructs a pair of conformance test cases that reflect the interoperability tests and recommends each SUT to investigate the issue independent of each other.

Figure 4 illustrates an execution scenario of an interoperability test case to verify whether a pair of SUTs can perform a basic message exchange (Message A and B). The ITRA requires that the interoperability test harness includes a monitor test component that transparently records messages exchanged between SUTs. The test procedure is as follows. First, the SUT1 receives a test instruction from an interoperability test case. It then creates and sends Message A to the SUT2. Message A is generated according to the test instruction, presumably, in accordance with the standard specification. Second, the SUT2 receives and processes the message and then creates a response Message B. The response is generated, presumably, according to the standard specification. This response message will be sent to the SUT1. Finally, the test executor analyzes and reports a test result by comparing the response message with an assert instruction in the test case. It is noted that the monitor records all messages exchanged.

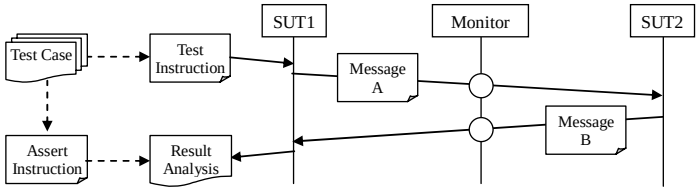
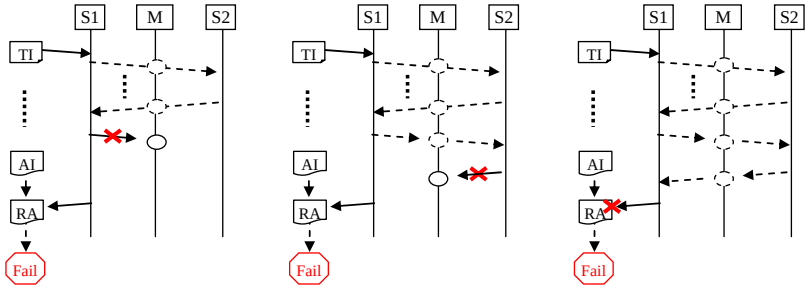


Fig. 4. Interoperability test case example

In this interoperability test, the ITRA defines three types of scenarios to detect interoperability problems as shown in Figure 5. In the Figure 5a and 5b, the monitor detects that each SUT did not send any message. In this case, the ITRA generates and recommends a new conformance test case for either the SUT1 or SUT2. In the Figure 5c, the monitor receives all messages but the test result is `fail' because SUT1 cannot process the response message from the SUT2 (the message has unexpected content). In this case, the ITRA generates and recommends a pair

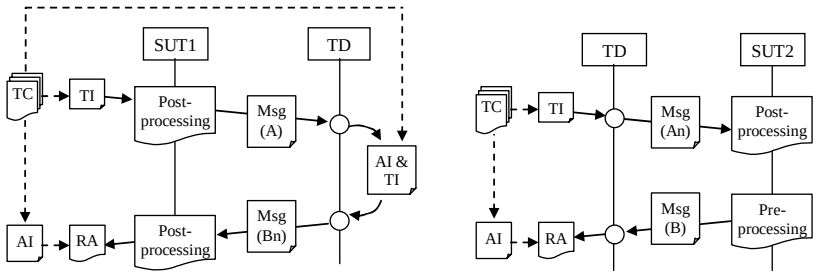
of conformance test cases for both SUTs. These test cases simulate the same interoperability test, but they allow SUTs to figure out and resolve the problem independent of each other instead of repeatedly retrying the more convoluted interoperability tests.



a. scenario 1 b. scenario 2 c. scenario 3

Fig. 5. Failed interoperability scenarios (TI: test instruction, AI: assert instruction, and RA: result analysis, S1: SUT1, S2: SUT2, and M: monitor)

Figure 6 illustrates a pair of conformance test cases generated from the interoperability test case in Figure 4. ITRA generates these test cases by using test traces recorded by the monitor. Note that the test driver component in Figure 6 simulates the partner SUT, and the nominal Message A and the nominal Message B are parameterized based on actual message traces; e.g., the message date and time has to be current.



a. conformance test case for SUT1 b. conformance test case for SUT2

Fig. 6. Generated conformance test case for SUT2 (TC: test case, TD: test driver, Msg: message, Msg An: nominal message based on message A, and Msg Bn: nominal message based on message B)

We summarize below a set of rules that the ITRA will use to recommend conformance test cases.

Rule 1 When the monitor does not receive a message from a SUT, the ITRA generates a conformance test case and recommends the SUT to pass the test via an independent conformance testing.

Rule 2 If the monitor receives all messages but the test result is `undetermined' or `fail', the ITRA generates a pair of conformance test cases for SUT1 and SUT2 and recommends each SUT to pass these test cases independently.

Rule 3 ITRA recommends that the expert of the target specification analyze the conformance test cases generated from RULE 1 and 2 and update the conformance test suite in the same way as Rule 3 for CTRA in Section 4

An engineer should consider how to eliminate the unanticipated interoperability issues. It may be that the standard specification is ambiguous or the conformance test suite is missing some important test cases and/or verification conditions.

6 Case Study

For comparative study, we illustrate an experimental test scenario to apply the ITP to the traditional CIT in the ebXML IIC test framework [7]. This experimental test scenario is inferred from experiences in the testing of the ebXML Messaging Service (ebMS) specification [18] within the Korea B2B Interoperability Testbed (KorBIT) [19]. Two examples of fatal interoperability problems are illustrated below:

- XML Prolog mismatch: SUT1 supports and generates the SOAP message with an XML declaration in XML Prolog but SUT2 does not. The declaration is an optional functionality of the ebMS specification. The SUT2 cannot parse any message from the SUT1 because it disregards this test case.

- Timestamp expression mismatch: SUT2 supports post-processing of various timestamp expressions but in a certain configuration setting it generates a timestamp using the expression containing a decimal fraction as default such as “20050630T103420.60”. The SUT2 only supports timestamp expressions such as “20050630T103420”, “2005-06-30T10:34:20” or “050630T103420”. When the SUT B sends a message including the decimal timestamp to the SUT1, the SUT1 cannot parse the message because it cannot process the decimal timestamp. The standard ebMS conformance test suite [20] does not account for timestamp representation and processing.

To resolve these interoperability problems, experts can use either traditional CIT or proposed ITP. For a comparison between the two approaches, Table 1 shows how each approach resolves the interoperability problems. In addition, Table 2 shows

how the ITP requires less time and expense than that of the CIT for resolving the interoperability problems.

Table 1. Procedure for CIT and ITP to resolve interoperability problems

Interoperability Problems	Procedure for CIT	Procedure for ITP
XML Prolog mismatch	Conformance testing may not detect this interoperability problem because the conformance test case regarding the XML Prolog is optional, but interoperability testing will fail due to the difference in message packaging. The CIT recommends that both SUTs repeat the same interoperability test to debug the problem without any suggestion.	Using Rule 1 in Section 5, the CTRA can detect this interoperability problem before the interoperability testing. It recommends that the SUT2 passes a post-processing test case corresponding to this problem in the conformance testing or the SUT1 packages messages without an XML declaration in the XML Prolog. In this case, both SUTs know in advance about the problem.
Timestamp expression mismatch	Conformance testing cannot detect this interoperability problem, but interoperability testing fails due to the difference in timestamp formats. The CIT recommends that both SUTs repeat the same interoperability test to debug the problem.	The SUT2 configuration does not produce the decimal timestamp during the conformance test. Hence, the CTRA cannot detect this problem. The SUTs subsequently fail the interoperability test. ITRA uses Rule 1 in Section 6. It generates and recommends a new conformance test case for SUT1, which simulates the same interoperability test. The SUT1 can perform the test and analyze the cause of the problem independently without consuming the time of SUT2 engineers.

Table 2. Solution comparison of CIT and ITP

Interoperability Problems	Test Cost for CIT	Test Cost for ITP
XML Prolog mismatch	In the KorBIT interoperability trials, an ebMS vendor had taken several weeks to detect and resolve these problems because the vendor could not expect the problems in the interoperability tests. The vendor had to analyze his system in detail to find the cause and consider correlations with his integrating vendor.	Because the ebMS vendors could find this problem using CTRA, it took them only a few days to perform the recommended conformance tests before the interoperability testing.
Timestamp expression mismatch		Because the SUT1 could simulate the interoperability test without the SUT2 by using the conformance test case generated from ITRA, it

		had taken the ebMS vendor for SUT1 a few days to perform and analyze the generated conformance tests. After detecting the failure cause , it had takenboth ebMS vendors a few more days to resolve it.

In this case study, ITP could efficiently resolve the interoperability issues with two additional conformance test cases. Furthermore the expert can update new test cases in the conformance test suite, such as a conformance test case to check if the SUT can process a timestamp expression containing a decimal fraction.

7 Conclusion and Future Work

This paper characterizes the interoperability issues into the anticipated, potential, and unanticipated. The conformance test cases are written for the anticipated issues while potential and unanticipated issues are discovered in the interoperability test. Software solutions may not pass all conformance test cases, resulting potential interoperability issues that can occur in the interoperability test. By the virtue that the conformance testing is less expensive and less complicated than the interoperability testing, we proposed an Iterative Test Procedure (ITP), which recognizes potential and unanticipated interoperability issues and generates/recommends test cases to resolve them in the conformance testing mode. The ITP includes rules for recognizing these issues implemented in the Conformance Test Result Analyzer (CTRA) and the Interoperability Test Result Analyzer (ITRA). These rules are not standard specific. CTRA and ITRA modules analyze test results and message traces, intelligently compose parameterized test messages, and subsequently form conformance test cases.

The ITP has been applied to the testing of the messaging standard specification. We hypothesize that the same approach can benefit the conformance and interoperability testing of other e-business layers such as the business process and business document specifications. Our future research will follow this hypothesis.

Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply that these products are necessarily the best available for the purpose.

8 References

[1] Scott Moseley, Steve Randall, Anthony Wiles, (2004) In Pursuit of Interoperability, International Journal of IT Standards and Standardization Research, vol. 2 no. 2, pp. 34-48

[2] James D. Kindrick, John A. Sauter, Robert S. Matthews, (1996) Improving Conformance and Interoperability Testing, StandardView, vol 4, issue 1, pp. 61-68

[3] Sungwon Kang, Relating interoperability testing with conformance testing, (1998) Global Telecommunications Conference, vol 6, pp. 3768-3773

[4] Durand, J., Kass, M., Wenzel, P. (2003) The ebXML Test Framework and the Challenges of B2B Testing, ebXML Conference 2003

[5] Boonserm Kulvatunyou, Nenad Ivezic, Albert T. Jones, (2005) Content-Level Conformance Testing: An Information Mapping Case Study, International Conference on Testing of Communicating Systems: 17th IFIP TC6/WG 6.1 International Conference, TestCom 2005, Montreal, Canada, pp.349-364

[6] S. Bradner, (1997) RFC2119: Key words for use in RFCs to Indicate Requirement Levels

[7] ebXML Implementation, Interoperability and Conformance (IIC) Technical Committee, ebXML IIC Test Framework Version 1.0

[8] ISO/IEC 9646: 1994, OSI Conformance Testing Methodology and Framework Parts 1-7

[9] ITU-T X.290 Series, (1994) Conformance Testing Methodology and Framework

[10] ETSI TS 102 237-1, Interoperability test methods & approaches, Part 1: Generic approach to interoperability testing

[11] Stephen Castro, (1991) The relationship between conformance testing of and interoperability between OSI systems, Computer Standards and Interfaces 12, pp.3-11

[12] Arakawa, N., Phalippou, M., Risser, N., Soneoka, T., (1993) Combination of conformance and interoperability testing, in Formal Description Techniques, M. Diaz and R. Groz, Eds. New York: Elsevier, 1993, vol. C-10, pp. 397-412.

[13] O. Rafiq, R. Castanet, (1990) From conformance testing to interoperability testing, Proceedings of the 3rd International Workshop on Protocol Test System

[14] Sungwon Kang, Myungchul Kim, (1997) Interoperability Test Suite Derivation for Symmetric Communication Protocols, FORTE, pp. 57-72

[15] Soonuk Seol, Myungchul Kim, Sungwon Kang, Jiwon Ryu, (2003) Fully automated interoperability test suite derivation for communication protocols, Computer Networks 43(6), pp. 735-759

[16] Mazen, M., Dibuz, S., (1998) Pragmatic method for interoperability test suite derivation, In Proceedings of 24th Euromicro Conference, pp. 838-844, IEEE

[17] D. Richard Kuhn, Vadim Okun, (2006) Pseudo-Exhaustive Testing for Software, 30th NASA/IEEE Software Engineering Workshop, April 25-27

[18] OASIS ebXML Messaging Services Technical Committee, ebXML Message Service Specification Version 2.0

[19] Korea B2B/A2A Interoperability Testbed (KorBIT), http://www.korbit.org

[20] ebXML Implementation, Interoperability and Conformance (IIC) Technical Committee, ebXML Messaging (2.0) Conformance Test Suite Version 1.0