23

Interoperability testing in an Implementation of ISO/OSI Protocols

P. Castori, P. Pleinevaux, K. Vijayananda, F. Vamparys
Swiss Federal Institute of Technology, Lausanne
Department of Computer Engineering
EPFL-DI-LIT, CH-1015 Lausanne, Switzerland
Email:{castori, ppvx, vijay, vamparys}@di.epfl.ch

Abstract

Practice has largely shown that interoperability problems often occur between implementations of different vendors. This paper presents the results of an experiment made at EPFL-LIT with an implementation of the CNMA profile that has been tested for interoperability with a number of commercial implementations. We present the results of the interoperability tests, propose a taxonomy of errors, compare with other interoperability tests and then discuss the lessons we have learned from this interoperability test campaign.

Keywords

Interoperability testing, CNMA, OSI

1 INTRODUCTION

Integration of enterprise applications requires the provision of a communication infrastructure that will allow applications running on different machines to exchange data. Essentially two approaches can be followed: adopt proprietary networks that must then be interconnected by expensive gateways or adopt a vendor independent communication architecture like CNMA (Communications Network for Manufacturing Applications) [7], MAP (Manufacturing Automation Protocol) [9] or TOP (Technical Office Protocol) [23]. The replacement of proprietary networks by standard networks does not come for free at the beginning. As practice has largely shown, interoperability problems appear between implementations of different vendors. These problems manifest themselves as obstacles to establish communication between applications. Very often, the complexity of the ISO/OSI protocols adopted in MAP, TOP and CNMA are mentioned as the reason why interoperability problems show up.

This paper presents the results of an experiment made at the Swiss Federal Institute of Technology in Lausanne with an implementation of the CNMA profile. This profile based on ISO/OSI protocols is tailored to industrial applications and used in industrial factories at Mercedes-Benz, Aerospatiale, etc. This experiment consists in testing this new implementation for interoperability with commercial implementations of the CNMA and MAP profiles recording the problems that appeared during the tests.

L. Camarinha-Matos et al. (eds.), *Balanced Automation Systems*

The MAP and CNMA profiles are compatible in the sense that implementations of one profile are able to communicate with implementations of the other profile. An Ada implementation of the CNMA profile made at the Swiss Federal Institute of Technology, consists of approximately 150 000 lines of Ada code, this figure giving an idea of the complexity of the software tested in this experiment.

The topic of testing for communication network is covered by a number of publications concerning conformance tests [15, 20, 21, 17, 16]. These articles concentrate on the methodology that ISO has been standardizing and on tools [17] for conformance testing. Interoperability tests have been discussed for the lower layers [13], for X.400 [14], for the HP MAP 3.0 implementation [18].

The paper is organized as follows. The implementations under test are described in section 2. Section 3 describes the methodology for the interoperability testing. Section 4 presents a taxonomy of errors. In section 5 we discuss the results using the above taxonomy and present the lessons learnt from this test campaign. In section 6 we compare our results with other interoperability tests and in particular with the results of the interoperability testing for HP MAP 3.0. The conclusion is then presented.

2 CONFORMANCE AND INTEROPERABILITY TESTING

2.1 Conformance testing

The objective of conformance testing is to establish whether the implementation being tested conforms to the specification of a given protocol. Conformance tests involve two implementations: a reference implementation which is assumed to perfectly reflect the standard being tested and the Implementation Under Test (IUT). The reference implementation interacts with the IUT according to well defined scenarios and the responses or behavior of the IUT are observed and recorded. For each interaction started by the reference implementation, a reaction is expected from the IUT.

In the ESPRIT CNMA project for example, conformance tests have been performed between 1988 and 1992 with tools developed first in the project then by ESPRIT TT-CNMA. Conformance tests have been very useful with the first versions of the CNMA implementations but as these matured the need for conformance testing has strongly decreased. Some of the drawbacks of conformance testing are their cost, incomplete coverage and they do not guarantee interoperability.

2.2 Interoperability testing

Several classifications of interoperability testing have been defined in the literature [8, 2]. These classifications are based on different levels of interoperability and criteria for success/failure of the interoperability testing. We now present an informal definition of an interoperability test. This definition will be based on function (Typically in ISO terminology, function can be considered as a service) provided by any layer of the stack. We try to keep our definition as general as possible and try to cover all aspects of the functions offered by all layers.

Let us first define a *function* offered by any layer N. This definition will form the basis for determining the success and failure of an interoperability test. Suppose layer N provides a_N number of functions. A $function_i^N$ ($i \in 1..a_N$) provided by layer N consists

of a *request PDU* (Req_i^N) and *response PDU* ($Resp_i^N$). $function_i^N$ is provided by the *responder* and the *initiator* requests for the function. An initiator accesses $function_i^N$ by sending (Req_i^N) to the responder. The initiator receives a $Resp_i^N$ in response to Req_i^N. The $Resp_i^N$ may be remote (sent by the responder) or local (generated by the local layer N-1).

Sometimes, $Resp_i^N$ has temporal requirements. This means that the initiator expects $Resp_i^N$ within a certain time T_i^N after it has sent the Req_i^N. This requirement is not a strict requirement for all $function_i^N$ ($i = 1..a_N$), Req_i^N and $Resp_i^N$ may be composed of multiple PDUs. For example, in the `data transfer` service offered by TP4, the request PDU may contain multiple DT PDUs (data transfer PDU). The `CMIP M-GET` function [12] response PDU also contains more than one PDU.

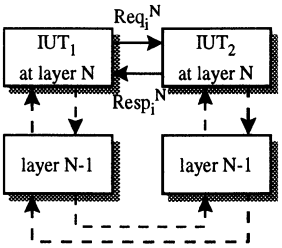

Figure 1 Interoperability test at a Layer N

The response PDU $Resp_i^N$ can be interpreted in many ways. Based on the correctness of response PDU it can be classified into the following categories.

Correct response: This includes all the responses that conform to the function definition. In some layers, the absence of a function response is also considered as a correct response.

Delayed Response: When $Resp_i^N$ includes multiple PDUs, some of the PDUs may be delayed. In other cases, the $Resp_i^N$ may not be available immediately. In this case, $Resp_i^N$ is sent by the *responder* when it is ready and available. The transport layer (TP4) retransmission function falls under this category.

No Response: When the initiator does not receive $Resp_i^N$ within a certain time T_i^N it is considered as no response. In this case, the absence of $Resp_i^N$ is a *failure* of the interoperability test. The absence of response can be due to non-availability of the responder or failure of the lower layer services.

Partial Response: This includes the cases where a $Resp_i^N$ includes multiple messages and only a few of them have been received. This type of response is considered as incorrect and results in *failure* of the interoperability test.

Incorrect Response: When a $Resp_i^N$ does not conform to its definition, it belongs to this category. The non-conformity can refer to the scope of the response (semantics) or its format (syntax).

Error Response: This type of response indicates that the responder cannot provide the requested function because an error or problem has been found in Req_i^N . The response indicates the reason for rejection or inability to provide the requested function. It results in the *failure* of the interoperability test.

The above criteria for classifying $Resp_i^N$ cover the syntax and semantics of the $Resp_i^N$ and also its temporal requirements. They also form a sound basis for determining the

correctness of an interoperability test. Later on, this is also used for classifying the errors encountered during the interoperability testing.

Now we are ready to define an interoperability test for a $function_i^N$ provided by layer N. Informally an interoperability test between two implementations IUT_1 and IUT_2 for a $function_i^N$ provided by layer N can be defined as follows:

Initiator IUT_1 *sends a* Req_i^N *to Responder* IUT_2.

If IUT_2 *responds with a correct and timely* $Resp_i^N$ *and it is received by* IUT_1 *then* IUT_1 *and* IUT_2 *are said to interoperate with respect to* $function_i^N$

This can be represented as

$$IUT_1 \overset{function_i^N}{\longrightarrow} IUT_2$$

Figure 1 shows the interoperability testing of a $function_i^N$ provided by layer N.

This informal definition of an interoperability test can be applied at any layer. It is applicable to both confirmed and unconfirmed services offered by all layers.Based on this definition we now define *layer interoperability.*

If $\overset{function_i^N}{\longrightarrow}$ *holds good for* $i = 1..a_N$, *then* IUT_1 *is said to be layer interoperable with* IUT_2 *with respect to layer N. This can be represented as* $IUT_1 \overset{layer^N}{\longrightarrow} IUT_2$

This definition of an interoperability test determines the success/failure of a test based on the response PDU $Resp_i^N$. In [5], Castro has presented the success/failure of an interoperability test based on the results of conformance testing. A predictive metric based on the results of conformance tests is used to determine the success/failure of a test. The drawbacks of this method is the inability to determine the success of those test cases that do not occur during a conformance test which is not the case with our definition.

An interesting point to be noted is that $\overset{function_i^N}{\longrightarrow}$ and $\overset{layer^N}{\longrightarrow}$ are not symmetric. That is

$$(IUT_1 \overset{function_i^N}{\longrightarrow} IUT_2) \not\Rightarrow (IUT_2 \overset{function_i^N}{\longrightarrow} IUT_1)$$
$$(IUT_1 \overset{layer^N}{\longrightarrow} IUT_2) \not\Rightarrow (IUT_2 \overset{layer^N}{\longrightarrow} IUT_1)$$

It is essential to distinguish between initiator and responder associated with a $function_i^N$. These two are well separated at the user application level and are provided as separate implementation. At the user application level, the initiator is also referred to as *client* and the responder is referred to as *server.* This is not the case with layer N. Normally an implementation at layer N contains both the initiator and responder. A successful interoperability test $IUT_1 \overset{function_i^N}{\longrightarrow} IUT_2$ tests only the initiator of IUT_1 and the responder of IUT_2. This explains why $\overset{function_i^N}{\longrightarrow}$ may not be symmetric.

Most of the interoperability tests are performed at the interworking level (between two user applications). At the layer level, the tests are conducted in an indirect manner. The user application layer make use of the lower layer services. As a consequence, interoperability testing of the user application services results in the testing of those lower

services that are utilized by the user application. It is interesting to note that all the lower layer services may not be tested using this strategy. It is necessary to design test strategies for interoperability testing at each layer. This will increase the confidence of the interoperability tests performed at the interworking level [14].

Interoperability tests are not perfect either, compared to conformance tests they do not cover as wide a spectrum of behaviors as conformance tests [18]. Very often interoperability tests involve two implementations from two different vendors using scenarios that are either identical or very close to those encountered in user applications. Performing a single battery of interoperability tests between any two applications is a relatively simple and straightforward process, but when many products are involved, the level of difficulty soars. For example, to test the interoperability of 12 different products, no fewer than 132 series of tests are needed ($n \times (n-1)$ because tests are not symmetric).

Interoperability tests are informal in the sense that no certificate will be issued by an accredited organization at the end of the tests. The Interoperability testing is not standardized and is still under study. Nevertheless for end users, interoperability tests are the most important tests that must be passed.

3 IMPLEMENTATIONS UNDER TEST

LITMAP is an implementation of the MAP/CNMA stack at our laboratory LIT (Laboratoire d'Informatique Technique) [22, 24]. It had been implemented on Sun Sparc workstations. Ada is the implementation language and has been chosen for its run time support and cross development facilities. LITMAP runs on embedded systems (Force Sys 68040 and MVME372A: MAP Interface Module from Motorola Inc.) [19] connected by token bus or Ethernet and on Sunsparc workstations connected by Ethernet.

Figure 2 shows the profile of LITMAP. The MAC layer supports IEEE 802.3 (Ethernet) and IEEE 802.4 (Token Bus) protocols. The network layer supports connectionless network layer protocol (CLNP). Transport Class 4 (TP4) is supported at the transport layer. The session and presentation layers support kernel services. The application layer services include ACSE and MMS (Manufacturing Message Specification) [11].

Interoperability testing was conducted between LITMAP and some commercial implementations. The following is a brief description of each implementation and its profile.

1. a Concord/SISCO implementation on PC, under DOS, attached to an 802.4 token passing bus on carrierband,

2. a Siemens S5 115 Programmable Logic Controller, attached to an 802.3 segment,

3. an SNI UNIX PC, attached to an 802.3 segment,

4. a Siemens PC, under Windows, attached to an 802.3 segment, and

5. a DEC ULTRIX DECStation, attached to an 802.3 segment. A router developed at EPFL-LIT was used to connect the 802.3 Ethernet network with the 802.4 carrierband network.

4 METHODOLOGY

In this section we present the testing methodology. The interoperability testing is done at the interworking level. The tests used for the interoperability testing are described briefly. The tools and utilities that helped during these tests are also discussed.

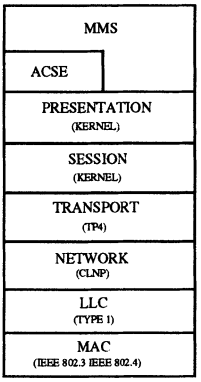

Figure 2 Profile of LITMAP

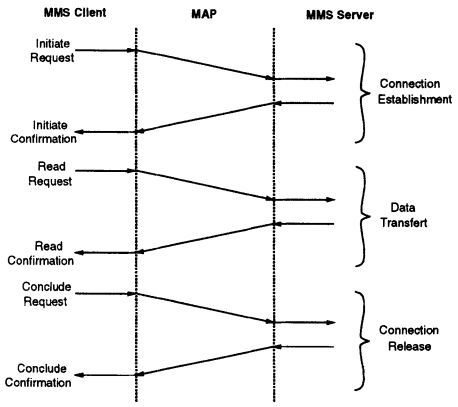

Figure 3 Testing methodology

4.1 Testing methodology

For all our interoperability tests we adopted the same methodology. It consisted of:

1. Opening an association between the MMS client and MMS server,
2. Transferring data on the opened association, and
3. Closing the association.

These various steps are illustrated on figure 3. Association establishment is the hardest part of interoperability tests. This is where we encountered the majority of problems. As a matter of fact, opening a connection means setting up all the information necessary for achieving reliable communication between two different entities. Communicating entities exchange information regarding their capabilities and requirements. A connection is then used as a safeway for data transfer. This phase is long and complex with respect to data transfer. It involves operations such as memory reservation, parameter negotiation, etc. At the user level, opening a association is done with the MMS `Initiate` service.

Data transfer is the second step of interoperability tests. Most known implementations support only a small subset of services. Only this subset could be used to perform tests at this stage. We systematically tested the well-known and classical services provided by all vendor implementations: `Identify`, `Status`, `GetNameList`, `GetCapabilityList`, `Read`, `Write` and `GetVariableAccessAttributes`. Read and and write services were restricted to simple variables. With one implementation we were also able to test MMS event management services such as `DefineEventCondition`, `DefineEventEnrollment` and `TriggerEvent`.

Finally, closing the association involves invoking the MMS `Conclude` service. This performs a gentle release of the connection environment and used resources are made available again for successive connections.

Not all vendor implementations supported both situations: acting as a server and acting as a client. But when it was possible, our implementation was tested both as client and as server. This is important because successful tests as a client (server) does not guarantee that tests will be successful as the server (client) as well.

Problems class	*Error*	*Implementation Problem*	*Interpretation Problem*	*Total*
Encoding / Decoding	1	0	0	1
Optional parameters	4	0	0	4
Incorrect Behavior	7	0	2	9
Addressing	1	3	0	4
Consistency checks	0	1	0	1
Error Code	1	0	3	4
Total	14	4	5	23

Table 1 Number of problems uncovered in interoperability tests

4.2 Limitations of our interoperability tests

The interoperability tests discussed in this paper were performed for the entire communication stack including layers 1 to 7 and even MMS servers. Only correct sequences of services and correct parameters were provided at the user interface.

Segmentation/Reassembly at the transport level was not tested because the size of the MMS PDUs sent was always smaller than the maximum PDU size negotiated at the transport level.

Error detection and recovery at the transport level was not exercised because tests were made in conditions that did not produce PDU losses such as electromagnetic noise or router overloads.

The above list shows that conformance tests are necessary and complementary to interoperability tests because they can test mechanisms that are not stressed in interoperability tests and rarely used in practice.

4.3 Utilities

During our interoperability tests, many utilities proved to be useful if not indispensable. First, we used a protocol analyzer from Siemens (K1102 for PCs). It includes decoders for the OSI protocols and MMS. A protocol analyzer is a fundamental tool for interoperability tests. It helps to discover and diagnose errors and is independent of the tested implementations. A trace facility was also used. This gives us the possibility to trace the PDUs between the different layers in LITMAP. A memory dump facility helped us to see on-line how memory was used in our stack. The dump facility allowed us to see whether memory was correctly allocated (or released) when the connection was established (or terminated).

5 ERROR CLASSIFICATION

During the interoperability tests, we encountered both problems and errors with the various implementations. We call *error* an incorrect implementation of a mechanism or parameter that is well defined in the standard or profile. We call *implementation problem* a problem related to implementation choices made by the vendor which are not incompatible with the standard. Different choices made by two different vendors may lead to impossibility to interoperate. Finally we call *interpretation problem* a problem that appears when the standard is not clear on the presence or details of parameters or mechanisms.

This first classification of problems and errors is orthogonal to a second classification we introduce and use here which makes a distinction among problems or errors according to the nature (coding, addressing, behavior, etc...) of these. In the following subsections we define these types of problems/errors.

This classification helps to identify the areas where most of the errors are found. It acts as a guideline for future tests and also helps to minimize the time taken to isolate and rectify the errors encountered during interoperability testing.

Encoding/Decoding errors: This type of error is encountered when data is encoded using different formats. When there are many choices for the encoding mechanisms, and a particular choice is not supported by an implementation, it results in a decoding error. This error is mostly encountered in the presentation layer which uses ASN.1 BER for encoding data. Another potential source of this class of error is encountered when decoding the headers of PDUs in the lower layers (session, transport and network). When certain features like *concatenation of PDUs* is not supported, it can result in a decoding error (when the decoder does not recognize the rest of the concatenated PDU). Another source for decoding errors is incorrect composition of PDUs. When octets are out of place in a PDU it can result in a decoding error.

Behavior problems: This type of error is encountered when a certain functional aspect of the protocol does not exhibit the expected behaviour. It is very difficult to isolate and diagnose this type of error. A behavior error occurs when the vendor does not implement a mechanism correctly. A behavior implementation problem occurs when a vendor imposes restrictions on a mechanism, for example forbids to have a transport credit larger than one. Finally a behavior interpretation problem occurs when the standard does not clearly define the operation of a protocol mechanism.

Addressing problems: Addressing errors occur due to mismatch in addresses (SAPs), limitation of the configuration of addresses (NSAP) or incorrect address formats (number of bytes used for SAPs). Since address configuration is an implementation detail, it can result in problems that are specific to particular implementations.

Consistency Check: When parameters are negotiated, the value of the negotiated parameters must be checked by the initiator of the negotiation. When an implementation does not perform this consistency check, it may lead to other kinds of errors (ex. cascade effect). For example, if the value of the *presentation context identifier* is not checked by the initiator of an association, it can result in the presentation data PDU being rejected by the responder.

Optional fields: Optional fields are another source of errors during interoperability testing. Most of the errors are the result of an implementation requiring a field that is optional or not handling a field that could optionally be present in the PDU. When optional fields are present in a PDU, the implementation can ignore it, but cannot reject a PDU due to its presence.

6 RESULTS AND OBSERVATIONS

6.1 Results

The results of the interoperability testing with different vendors are discussed in this section. Table 1 gives the results of the test according to the classification discussed in section 5. Table 3 presents the results for each layer. This tables gives an idea about the

Response	Layer	Application
No	1	6
Partial	0	0
Incorrect	12	5
Error	10	12

Table 2 Classification of errors based on the type of response received

number of errors detected in each layer and it also reflects on the complexity of the layer (refer to 6.2 for more details). Table 2 classifies errors based on the response received by the initiator. This classification is based on discussion in 2.2.

6.2 Observations

After this interoperability test experiment, we can make the following observations:

1. It is interesting to note that ratio of number of errors encountered to the number of implementations with which tests have been performed is very low. The total number of errors detected is also very low for the complexity of the protocols involved in the interoperability test. It is striking to see that the number of problems that have been found is as low as 23 in total and less that 4 by product tested in average.

2. Each time a new implementation is used, additional errors or problems are found. This fact reflects the large number of possible implementations of an OSI stack and the large number of ways in which these can be configured.

3. Testing as a client and as a server are two different situations. Having succeeded in client (or server) tests does not imply that tests in the reverse direction will succeed. This fact is taken into account by our definition of interoperability in 2.2.

4. Comparison of tests performed with two different implementations may point to unseen problems by one of these implementations. For example, an attempt to establish an association with one commercial implementation failed while it succeeded with another. Careful examination showed that the latter did not check a parameter, while the former did.

5. About 50% of the errors fall under the *Error* category. In particular, we can say these errors are a result of incorrect implementation and testing strategies.

6. Now that ASN.1 compilers are quite stable and ASN.1 is well understood by people, we observe a new distribution of errors different from that referenced in [5, 18] and which corresponds to the complexity of layers.

7. Addressing problems could be significantly reduced if the different vendors adopted a standard way of defining the addresses of the communicating entities. Having to deal with as many addressing definition notations as there are commercial implementations was a serious problem that delayed the interoperability testing with these implementations.

8. There is a lot of improvement to be done in standards bodies to define meaningful error codes and to precisely indicate in which circumstances these error codes must be used (Table 1). Simply defining a `Protocol Error` code is not enough and makes diagnosis of errors more difficult. The same remark is valid for MMS where the `Other` error is used by vendors in many contexts. Note that for application services like MMS, this means that an application cannot be 100% generic since it depends on the error codes returned by different vendors.

9. A considerable number of problems is due to differences of interpretation because of imprecisions, misleading definitions and the big number of options in standards and profiles. For example, when a connection is established in MMS, supported services and capabilities (S&C) are negotiated. It is not clear whether the returned S&C in the `Initiate` response are the intersection of supported S&C of both end-users or if it is the common S&C subset of the server application and the MMS-provider(The part of the application that conceptually provides the MMS service through the exchange of MMS PDUs). One of the tested implementation simply chose the former for supported services and the latter for supported capabilities!

7 COMPARISON WITH OTHER TESTS

In this section, we compare our results with other interoperability tests and in particular with the results of the interoperability testing for HP MAP 3.0[18]. These series of tests were conducted in 1990 with 8 different products from companies such as Allen-Bradley, Computrol, Motorola, etc. These tests were performed for FTAM and MMS. We will only compare the results of the MMS tests in this article.

In table 3, when we look at the number of problems by layer, we can see that the distribution of errors is very different between the two series of tests. Let's try to explain this. First, we can see that defect rates from MAC layer and session layer are low in the two columns.We can clarify this by the fact that the various implementations of these layers are quite stable and these layers are less complex compared to the others. Three of four defects found in the network layer are addressing limitations. The Swiss academic network SWITCH utilizes the *39 format* for NSAPs. The non-use of this address format accounts for the errors detected at the network layer. This explains the differing resilts observed for the network layer. In the transport layer, the larger number of errors comes from the fact that the transport class 4 protocol is complex and that for two of the implementations, it was their first interoperability test. In the case of the three upper layers, the comparison is altered by the large number of encoding errors in the HP tests. This can be explained by the multiple ways of encoding PDUs with the Abstract Syntax Notation One (ASN.1)[1]. In particular, ASN.1 permits the length of PDUs to be encoded in two different ways and several vendors had some errors in decoding both the encoding methods. At this point, it is worth mentioning that before 1990, complete and good ASN.1 compilers were scarce.

When we look at the HP tests, we see that the lower layers (up to session) are quite stable and the upper layers have several problems with ASN.1. Errors in our tests are more homogeneous in the sense that we can notice a relation between the number of errors and the complexity of the layer. The reason is certainly that some products we have tested are less mature and have been submitted to no or very few interoperability tests.

When we compare the results of the two tests, we note several identical lessons. First, interoperability testing is necessary because defects were uncovered in the tests with all vendors. Second, defect rates for a given implementation normally decrease steadily as interoperability testing is done with more vendors. We confirm that a considerable number of problems come from optional fields in network, transport and MMS layers and from negotiation of contexts in application and presentation layer. In our results, we also have several problems with the format of NSAP addresses and with the protocol error codes in network, transport, presentation and MMS layer. A small number of errors at the application level is the result of the availability of stable ASN.1 compilers[3].

Layer	*HP tests*	*LIT tests*
MMS	14	7
ACSE	14	1
Presentation	14	4
Session	2	1
Transport	1	6
Network	0	4
Data link	0	0
Total	45	23
Total/Vendor	5.6	3.8

Table 3 Number of problems uncovered in interoperability tests

8 CONCLUSION

Integration of enterprise applications increasingly relies on the use of standard communication networks. Practice has largely shown that interoperability problems occur between implementations of different vendors. In this paper we have presented the results of an experiment made at EPFL-LIT with an implementation of an OSI profile that has been tested for interoperability with a number of commercial implementations. The problems that were found during these tests have been classified and compared with other tests. Finally, we have drawn lessons on the way to proceed for interoperability testing.

Improvements can be made by both protocol specifiers and vendors. In particular, a significant effort should be made for the definition of precise error codes for all protocols. Profiles that in the context of OSI define subsets of services, parameters and mechanisms are not precise, complete and restrictive enough to ensure proper implementation by vendors. A clear documentation of a vendor implementation restrictions or particularities would certainly make the job of testers easier. Indeed, implementations may be 100 % correct and yet not be able to interoperate because of incompatible choices or implementation restrictions.

The distribution of problems and errors found in this study is different from that presented by other researchers. This difference is explained by the difference in maturity of the tested implementations and by the fact that ASN.1 compilers for example now produce error-free code for presentation and application protocols.

Work on interoperability tests will be continued in two directions: additional tests with other implementations such as those from DEC, HP and Bull; systematic exploration of interoperability problems at server level, above the application layer.

9 ACKNOWLEDGMENTS

We would like to thank our laboratory's colleagues of the CNMA group: F. Restrepo , G. Berthet and our colleagues of the ESPRIT CCE-CNMA for providing us with their implementations and their help in understanding some of the uncovered problems: A. Lederhofer and W. Blumenstock (Siemens), A. Horstmann, P. Wimmer (SNI), C. Bezencon (DEC).

REFERENCES

[1] "ISO/IEC 8824, CCITT X.208, Specification of Abstract Syntax Notation One (ASN.1", 1988.

[2] Y. Benkhellat, M. Siebert, J-P Thomesse, "Interoperability of sensors and distributed systems", *Sensors and Actuators*, Vol. 37, 1993, pp. 247–254.

[3] G. Berthet, "ASNADAC User Manual", EPFL-LIT Internal Report, 1993.

[4] G. Bonnes, "Verification d'inter-operabilite OSI, X.400 et LU6.2", *De Nouvelles Architectures pour les Communications Eyrolles*, 1988, pp. 133–140.

[5] S. Castro, "The relationship between conformance testing of an interoperability between OSI systems", *Computer Standard Interfaces*, Num. 12, 1991, pp. 3–11.

[6] "ESPRIT Project 7096 CNMA Implementation Guide Revision 6.0", November 1993.

[7] "ESPRIT CCE-CNMA, CCE: An Integration Platform for Distributed Manufacturing Application", Springer Verlag, 1995.

[8] J. Garde, C. Rohner, C. Summers and S. Symington, "A COS study of OSI interoperability", *Computer Standard Interfaces*, Num. 9, 1990, pp. 217–237.

[9] "General Motors, Manufacturing Automation Protocol, Version 3.0", August 1988.

[10] ISO 7498-1, "Information Processing Systems – OSI – Basic Reference Model".

[11] IS0/IEC 9506-1, "Manufacturing Message Specification: Service Definition", 1990.

[12] "ISO/IEC 9595-1, CCITT X.710 Common Management Information Service Definition", 1991

[13] L. Lenzini, "The OSIRIDE-Intertest Initiative: Status and Trends", *Computer Networks and ISDN Systems*, Vol. 16, 1989, pp. 243–255.

[14] L. Lenzini, F. Zoccolini, "Interoperability tests on OSI Products in the framework of the OSIRIDE-Intertest initiative", *Comp. Net. and ISDN Systems*, Vol. 24, 1992, pp. 65–79.

[15] R.J. Linn Jr, "Testing to Assure Interworking of Implementations of ISO/OSI protocols", *Computer Networks and ISDN Systems*, Vol 11, 1986, p. 277-286.

[16] R.J. Linn, "Conformance testing for OSI protocols", *Computer Networks and ISDN Systems*, Vol. 18, No. 3, pp. 203–220.

[17] R.S. Matthews, K.H. Muralidhar, S. Sparks, "MAP 2.1 Conformance Testing Tools", *IEEE Trans. on Software Engineering*, Vol. 14, No. 3, March 1988, pp. 363–374.

[18] J.D.Meyer, "Interoperability testing for HP MAP 3.0", *Hewlett-Packard Journal*, August 1990, pp. 50–53.

[19] "MVME372: Map Interface Module User's Manual", Motorola Inc., 1989.

[20] D. Rayner, "Progress on Standardizing OSI Conformance Testing", *Computer Standards and Interfaces*, Vol. 5, 1986, pp. 317–334.

[21] D. Rayner, "OSI Conformance Testing", *Computer Networks and ISDN Systems*, Vol. 14, 1987, pp. 79-98.

[22] D. Sidou, K. Vijayananda, G. Berthet, "An Architecture for the Implementation of OSI protocols: Support Packages, Tools and Performance Issue", *Proc. of SICON/ICIE'93*, Singapore, Sept. 93.

[23] "Boeing, Technical Office Protocol, Version 3.0", August 1988.

[24] F. Vamparys, P. Pleinevaux, "Architecture of the LITCommEngine", *EPFL-LIT Internal Report*, Feb. 1994.