# CONFORMANCE TESTING METHODOLOGIES FOR OSI PROTOCOLS

**Matthew Bush, Kris Rasmussen, and Fai Wong**

As open systems interconnection (OSI) protocol standards mature and systems are developed to implement them, facilities to test protocol implementations for conformity to relevant international standards and recommendations are becoming the key to vendor interoperability. This paper examines the test methods applicable to OSI conformance testing as defined by the International Organization for Standardization (ISO) and the International Telegraph and Telephone Consultative Committee (CCITT). The advantages and disadvantages of each test method are discussed. Each test method is analyzed in terms of its technique in providing synchronization between the test system and the system under test (SUT), and its ability to supply a controlled environment for comprehensive protocol testing. We also discuss the applicability of each test method to different OSI protocols, and the generic aspects of test system architecture common to all test methods. Specific attention is paid to those test methods that will be used to test X.25 implementations and network management protocols.

***Matthew T. Bush*** *is a member of technical staff in the Network Management Systems Engineering Department of AT&T Bell Laboratories' Red Hill facility, Holmdel, New Jersey, where he is responsible for systems engineering of AT&T'S Accumaster® software, specifying conformance testing requirements, and translating OSI/network management forum protocol specifications into requirements. He has a B.S.E.E. from Worcester Polytechnic Institute, Worcester, Massachusetts; and is a candidate for an M.S.E.E. from Columbia University, New York. Mr. Bush joined AT&T in 1987.* ***Kris A. Rasmussen*** *is a protocol test group supervisor in the Software Technology Department at Red Hill, where he is responsible for providing cost-effective OSI interfaces to AT&T Network Systems*
(continued on page 100)

## Introduction

As protocol standards mature and systems that implement them are developed, facilities to test protocol implementations must be provided. In the realm of OSI, conformance testing has been introduced to ensure that protocol implementations satisfy the requirements of their standardized specifications. ISO and CCITT are jointly developing conformance testing standards that can be used to ensure that OSI products are consistently tested worldwide. These standards specify procedures for developing protocol test suites. They also define the test methods used to administer tests to OSI protocols.

This paper examines the ISO and CCITT methods defined for conformance testing. Each test method is characterized by the technique that provides synchronization between the conformance test system and the SUT. The tests are also discussed according to their abilities to provide a controlled environment for comprehensive protocol testing. In addition, the applicability of each method to testing different OSI protocols is presented. An example of an OSI conformance test system, developed and used by AT&T for X.25 conformance testing, is introduced as an illustration.

## OSI Conformance Testing

OSI conformance testing involves verifying that the external behavior of OSI implementations complies with the requirements contained in their relevant ISO and CCITT protocol specifications. The purpose of this kind of testing is to increase the probability of OSI interworking and vendor interoperability by determining whether different vendors' OSI implementations operate the same under the conditions simulated by a standardized suite of test cases.

During conformance testing, each protocol contained in the OSI stack is individually tested to ensure it meets the conformance requirements contained in the relevant ISO or CCITT protocol specification. Testing begins at the physical layer and proceeds upward through the OSI stack until each individual protocol has been tested. The tests are performed in an upward sequence because testing any OSI protocol requires reliable operation of all underlying OSI services (see the Basic Test Architecture section).

## Conformance Testing Methodologies

This section discusses the fundamental conformance test architecture and three of the test methods being defined by ISO:
- The coordinated single-layer (CS) test method
- The related distributed single-layer (DS) test method

**Panel 1. Terms and Acronyms in This Paper**

| | |
|---|---|
| AFNOR | Association Francaise de Normalisation |
| ANSI | American National Standards Institute |
| ASE | application service element |
| ASP | abstract service primitive |
| ASN.1 | Abstract Syntax Notation One |
| AXCTS | AT&T X.25 conformance test system |
| CCITT | International Telegraph and Telephone Consultative Committee |
| CMIP | common management information protocol |
| CMISE | common management information service element |
| COS | Corporation for Open Systems |
| CS | coordinated single-layer |
| DS | distributed single-layer |
| FTAM | file transfer, access, and management (application protocol) |
| ISO | International Organization for Standardization |
| IUT | implementation under test |
| MFOS | multi-function operations system |
| MHS | message handling service |
| NIST | National Institute for Standards and Technology |
| OSI | Open Systems Interconnection |
| OSI/NMF | Network Management Forum |
| PCO | point of control and observation |
| PDU | protocol data unit |
| PICS | protocol implementation conformance statement |
| PIXIT | protocol implementation extra information for testing |
| PTT | Postal, Telegraph & Telephone |
| ROSE | Remote Operations Service Element |
| RS | remote single-layer |
| SAP | service access point |
| SPAG | Standards Promotion And Applications Group |
| SUT | system under test |
| T1M1 | ANSI internetworking committee |
| TMP | test management protocol |
| TMPDU | test management protocol data unit |
| X.25 | standard international communications protocol used in packet switching networks |
| XPE | X.25 protocol emulator |

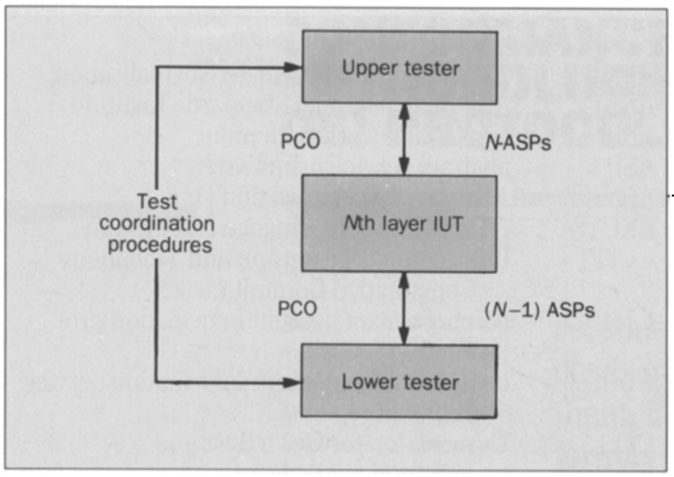


**Figure 1. Basic test architecture.**

- The remote single-layer (RS) method.

We also discuss both embedded and multi-layer versions of these single-layer tests.

**Basic Test Architecture.** The basic test architecture fundamental to each test method defined by ISO is shown in Figure 1. The figure depicts an *N*th-layer protocol entity being tested in isolation from all other OSI protocols. The locations where the testing device interfaces with the protocol implementation under test (IUT) are called points of control and observation (PCOs). In Figure 1, PCOs are present at the service interfaces above and below the IUT. The IUT is tested by exchanging abstract service primitives (ASPs) between the testing device and IUT at the two PCOs. (*ASP* has been defined for use in the evolving OSI conformance testing standards. There is no difference between an ASP and the service primitives defined in the OSI protocol service definitions.) The exact protocol exchanges that occur at the PCOs are specified in the *N*th-layer protocol test suite.

Each test method employs a different variant of this basic test architecture to execute protocol tests. Because the OSI testing device's operations are distributed over two PCOs (see Figure 1), each test method is defined in terms of three distinct testing functions: an Upper Tester, a Lower Tester, and test coordination procedures.

- An *Upper Tester* is a device for observing and controlling protocol events at the upper service boundary of the IUT.
- A *Lower Tester* is a device for observing and controlling protocol events at the lower service boundary of the IUT.
- *Test coordination procedures* specify a set of rules enabling the Lower Tester to control the actions of the Upper Tester. The procedures synchronize the activity of the testers and ensure that protocol events begun by these testing functions are transferred to the IUT in the proper sequence. The procedures needed to test a given OSI protocol are specified in that protocol's test suite.

Exact implementation of the basic test architecture shown in Figure 1 would require both the Upper and Lower Testers to reside in the same end system as the IUT. Therefore, they would not provide a practical means to test OSI implementations developed by different vendors. Each test method used by OSI conformance test systems allows the Lower Tester to interact with the IUT via some physical medium and the IUT's underlying OSI services (see Figure 2). [These OSI services—i.e., the ($N$−1) services in Figure 2—are provided by peer protocol entities operating in the end-

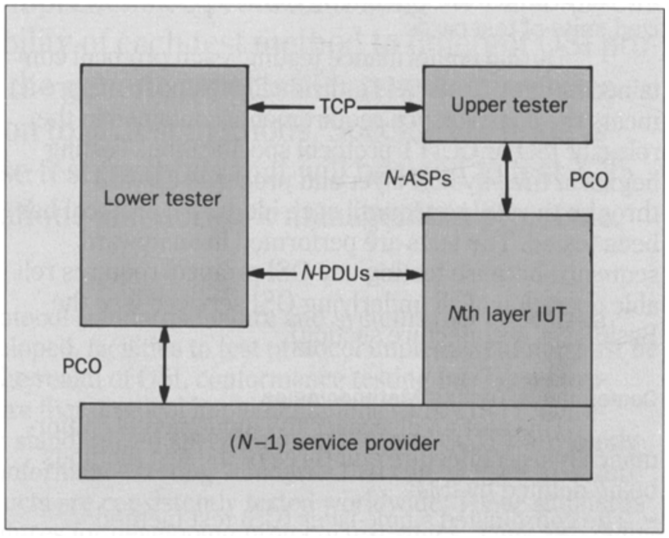


**Figure 2. External test method.**

system with the Lower Tester and in the end-system with the IUT.] In this configuration the Lower Tester acts as the peer entity of the protocol being tested, and resides in a different end-system than the IUT. The Upper Tester typically exists externally from the end-system with the Lower Tester, and functions as the user of the services provided by the IUT. When the Upper and Lower Testers reside in separate end-systems, the importance of the test coordination procedures and the synchronization provided by these procedures becomes obvious. These test coordination procedures provide the only means by which the Upper Tester can cooperate with the Lower Tester during the execution of a protocol test.

Test methods using a configuration like that in Figure 2 are called *external test methods*. This is because the Lower Tester exists and functions *externally* from the SUT. In this configuration the Upper and Lower Testers exhibit a master-slave relationship where the Lower Tester uses the Test Coordination Procedures to invoke the functions of the Upper Tester. When using external test methods, it is important to understand that the Lower Tester provides the principle means to coordinate and control the protocol test execution. All test suites used to test an *N*th-layer protocol implementation are stored and maintained in the Lower Tester. The Upper Tester typically does not have access to these test suites. Therefore, the Upper Tester cannot begin executing a protocol test until it receives instructions from the Lower Tester. The test coordination procedures define the instructions the Lower Tester gives to the Upper Tester. The Upper Tester is capable of interpreting the test coordination procedures and responding appropriately to those instructions.

Each OSI test method specified by ISO can be distinguished by how the Upper Tester and test coordination procedures may be implemented, and by the number and location of the PCOs it uses. The sections that follow describe the architecture of the external test methods being standardized by ISO in terms of these characteristics. Because the test methods are defined independent of the protocols (i.e., each test method can test any layer of the OSI reference model), the functions performed by each test method discussed in these sections are described in terms of the protocol actions associated with an *N*th-layer protocol implementation.

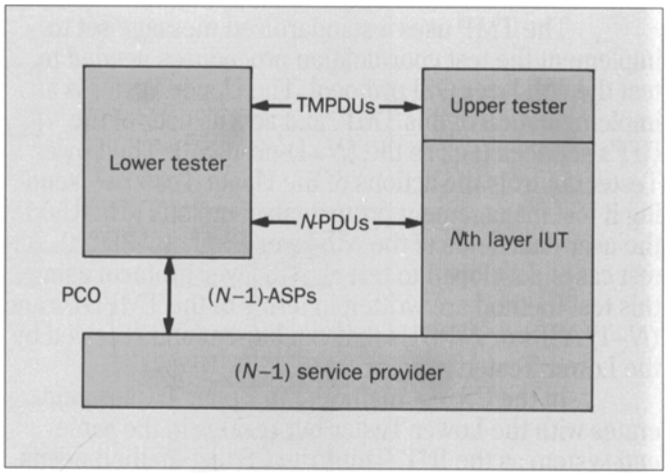


**Figure 3. Coordinated single-layer (CS) test method.**

**CS Test Method.** The CS test method shown in Figure 3 is the most robust test method ISO is standardizing. Though it uses both an Upper and Lower Tester, only one PCO is needed to implement its architecture. This PCO is specified to exist above the ($N$−1)-service provider at a service access point (SAP) remote from the IUT. Note that, despite the Upper Tester, there is no PCO at the service interface above the IUT.

The Upper Tester in the CS test method is used to execute test events (i.e., service requests and confirmations) at the service interface above the IUT. But it does not provide a means to *externally observe* test events at the interface in the SUT. Instead, it records information received from the IUT and transmits it to the Lower Tester for observation at the specified PCO. This transfer of information—and any other communication between the Upper and Lower Tester—is performed through a standardized test management protocol (TMP)[1,2].

The TMP uses a standardized message set to implement the test coordination procedures needed to test the *N*th-layer OSI protocol. The Upper Tester is an implementation of this TMP, and acts as user of the IUT's services (i.e., as the $(N+1)$-protocol). The Lower Tester controls the actions of the Upper Tester by sending it test management protocol data units (TMPDUs) in the user data fields of the *N*th-layer PDUs (*N*-PDUs). All test cases developed to test an *N*th-layer protocol using this test method are written in terms of the TMPDUs and $(N-1)$-ASPs or *N*-PDUs that can be sent and received by the Lower Tester.

In the CS test method, the Upper Tester cooperates with the Lower Tester but resides in the same end-system as the IUT. Thus, the CS test method needs the service interface of the IUT to be exposed so that the IUT can interface to the Upper Tester. Typically, the Upper Tester software is provided by the test laboratory for porting to the SUT. However, since the TMP is a standardized protocol, it may also be implemented by the SUT's owner.

The CS test method is well suited to test general purpose protocols—the transport layer and common management information protocol (CMIP)—that are likely to support a variety of users. This test method uses a single, standardized user (the Upper Tester), designed to exercise all services the IUT supports. Using an Upper Tester allows general-purpose protocols to be tested once rather than multiple times with different users.

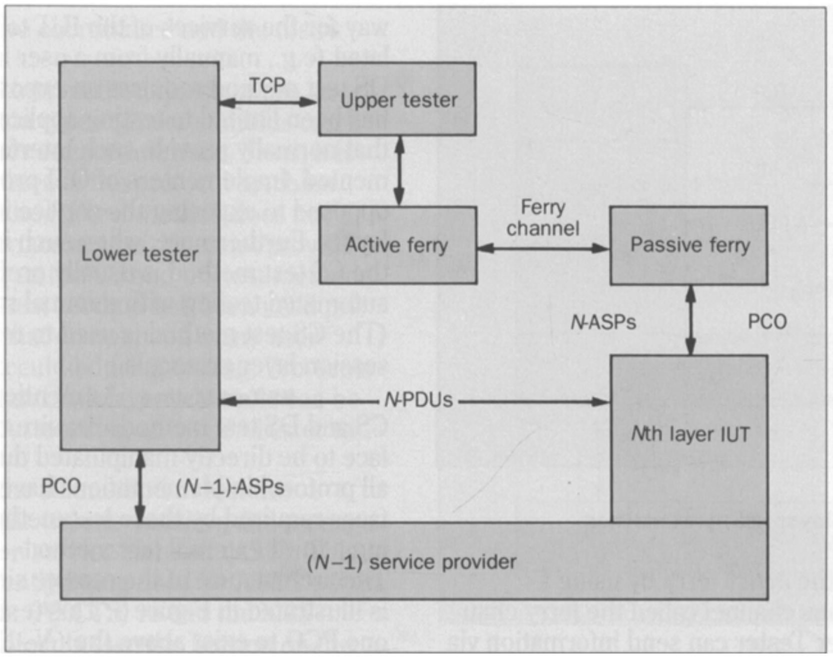
The Corporation for Open Systems (COS) uses the CS test method to test transport layer protocol implementations. It is among the test methods ISO selected for session layer testing. The test method also will be used to test the CMIP used by network management applications (see "Choosing a Test Method"). The major difficulty with the CS test method is synchronizing the activity of the Upper and Lower Testers[3]. As previously mentioned, all test cases are stored and maintained in the Lower Tester. Therefore the Upper Tester cannot execute actions specified in the tests unless the Lower Tester instructs it to do so. Some conditions defined in the protocol specification are difficult to test under these circumstances. For example, testing the IUT's reaction to a connection release "collision" requires both the Upper and the Lower Tester to simultaneously begin a connection release. (A connection release collision occurs when two end systems simultaneously attempt to tear down the connection between them.) However, since the Upper and Lower Testers reside on separate machines, it is almost impossible for these devices to simultaneously execute the required test events.

A second problem, unique to the CS test method, is that the IUT must be able to send *and* receive data if a TMP will be used to accomplish the test coordination procedures. Not all protocol implementations can do both. If the IUT cannot receive data, there will be no way for the Lower Tester to control the Upper Tester's activity by sending it TMPDUs. Similarly, if the IUT cannot send data, the Upper Tester will have no way to record test events at the IUT's service interface, and send this information across the $(N-1)$ service provider to the Lower Tester for logging.

Because of these problems, ISO has specified another test method that allows the same amount of control over the IUT as the CS test method, but does not use a TMP to implement the test coordination procedures. This test method, called the distributed single-layer (DS) test method, is described in the following section. Both the CS and DS test methods can test all states of the IUT because they allow the input/output sequences above and below the IUT (see Figure 1) to be controlled and observed.

**DS Test Method.** Figure 2 illustrates the architecture of the DS test method. The DS test method's testing level is comparable to the level of the CS test method; but it does not specify how the Upper Tester is to be implemented. The DS test method uses two PCOs, one at the service interface above the *N*th layer IUT, the other above the $(N-1)$ service provider at a SAP remote from the SUT. Both Upper and Lower Testers monitor

**Figure 4. The ferry approach.**



the protocol activity at the PCOs. Test cases executed with the DS test method are specified in terms of the *N*th-layer ASPs, and ($N$−1)-ASPs or *N*-PDUs that can be sent and received at the specified PCOs. The protocol test suite specifies functional ***requirements*** for the Upper Tester and for the test coordination procedures, but does not specify ***how*** the testing device will satisfy these requirements. (The CS test method specifies that the test coordination procedures and Upper Tester are to be implemented via a standardized test management protocol.) Consequently, the test coordination procedures and Upper Tester needed to execute the *N*th-layer protocol test suite may be implemented differently by different OSI test systems and test laboratories.

Typically, conformance test systems execute tests written for the DS test method in an interactive mode. The test coordination procedures and Upper Tester are implemented via verbal communication between the test system and SUT operators. Under these circumstances the operator of the test system—the Lower Tester—would verbally instruct the operator of the SUT to record and execute protocol events at the PCO above the IUT's service interface during test execution. In this situation the SUT operator acts as the Upper Tester, and test coordination procedure requirements are satisfied by verbal communication between the test system operator and SUT operator.

A second way to implement Upper Tester and test coordination procedures needed by the DS test method is by using a ***ferry protocol*** to transfer the events at the IUT's upper boundary to an Upper Tester on the same end-system as the Lower Tester[4]. Figure 4 shows the configuration used by the ferry approach. In this configuration a ferry device is situated at the service interface above the IUT. This device—the passive ferry—receives information from the IUT and relays it to the

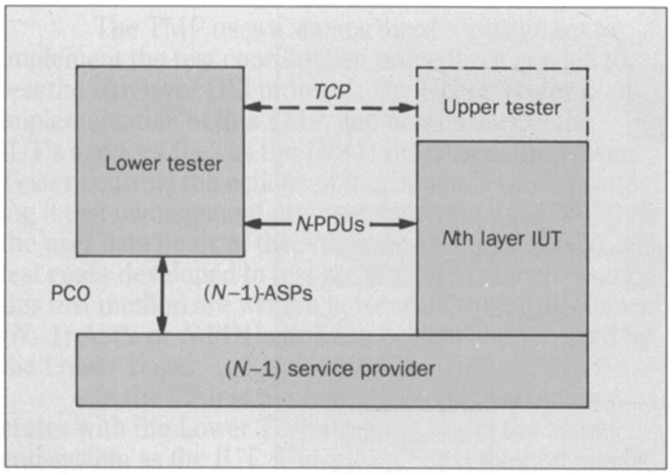


**Figure 5. Remote single-layer (RS) test method.**

Upper Tester through the *active* ferry by using a separate communications channel called the ferry channel. Similarly, the Upper Tester can send information via the active ferry and ferry channel to the passive ferry device, causing this device to execute test events at the IUT's service interface. The passive and active ferry devices shown in Figure 4 communicate using a ferry protocol[5,6] devised by the test laboratory.

The advantage of a ferry configuration is that the Upper and Lower Testers can reside on the same end-system and can synchronize using interprocess communication. This minimizes the timing problems discussed in the previous section. The disadvantage of this configuration is that it requires two separate communications channels between the Lower Tester and SUT. Since not all OSI implementations can support two such channels, the ferry configuration is not widely used, and has not become a distinct, standardized test methodology.

The DS method allows comprehensive testing of protocol implementations, but requires the IUT's service interface to be exposed to serve as a PCO. Exposing the IUT's service interface means the SUT must provide a way for the services of the IUT to be externally manipulated (e.g., manually from a user interface). Because the DS test method requires an exposed service interface, it has been limited to testing application layer protocols that normally provide such interfaces when implemented. Implementors of OSI protocols generally are opposed to exposing the service interfaces at other layers. Furthermore, when such interfaces are exposed, the CS test method is usually preferred because it allows automated testing with minimal restrictions on the IUT. (The CS test method is used to test the transport and session layer protocols.)

**RS Test Method.** As mentioned above, both the CS and DS test methods require the IUT's service interface to be directly manipulated during testing. Since not all protocol implementations have the open service interfaces required by these test methods, ISO is standardizing a third external test method—the RS test method. The architecture of the remote single-layer test method is illustrated in Figure 5. This test method requires only one PCO to exist above the $(N-1)$ service provider at a SAP remote from the SUT. All tests developed for the RS test method are specified in terms of the $(N-1)$-ASPs or $N$-PDUs that can be observed by the Lower Tester at this PCO.

Though the RS test method does not assume an accessible service interface—and thus does not use an Upper Tester—the SUT may be required to perform some Upper Tester functions to implement the test coordination procedures specified in the $N$th layer protocol test suite. These functions, if needed, typically are performed by the $(N+1)$–$(N+n)$ protocols in the SUT. For example, to establish a connection between the Lower Tester and IUT, the OSI protocols that use the services of the IUT—the $(N+1), (N+2), \cdots, (N+n)$ protocols—may have to respond to a connection indication service primitive issued by the IUT. Although the higher layer protocols give the testing device some control over the IUT's service interface, they do not allow activity at the interface to be externally *observed* (either by the Lower or Upper Tester) as in the coordinated and the distributed test methods.

Because the RS test method does not use an Upper Tester, only certain types of user stimulus can be provided at the IUT upper service interface. Thus, it is not possible to test all states of the protocol machine. Consequently, because the RS test method does not require IUT service interface access, it cannot provide the same level of testing as the CS and DS test methods.

To date, the RS test method has been used primarily to test protocols that do not need many user-initiated events to be executed during testing. Protocols such as the X.25 packet and link layer protocols can be tested extensively without needing an Upper Tester to provide user stimulus.

The COS uses the RS method to test the responder mode of the file transfer, access, and management (FTAM) application service element (ASE). This test method also is currently being used by AT&T's X.25 conformance test system (AXCTS) to test the X.25 packet and link layer protocols (see the section on the X.25 protocol).

**Embedded and Multi-Layer Test Methods.** Embedded versions of the coordinated, distributed, and remote test methods also have been defined by ISO. The embedded versions of these methods allow protocol implementations to be tested while embedded under one or more higher layer protocols. The only differences between the single-layer test methods described earlier and the embedded test methods defined by ISO are the complexity of the Lower Tester, and the location of the PCOs and Upper Tester (if applicable).

The configuration used by the coordinated single-layer embedded (CSE) test method is shown in Figure 6. Note that the CS test method (see Figure 3) specifies that the Upper Tester resides at the service interface *directly* above the IUT, while the CSE test method dictates that the tester resides at a service interface *n layers* above the IUT. As in the CS test method, the Lower Tester controls the activity of the Upper Tester using a TMP.

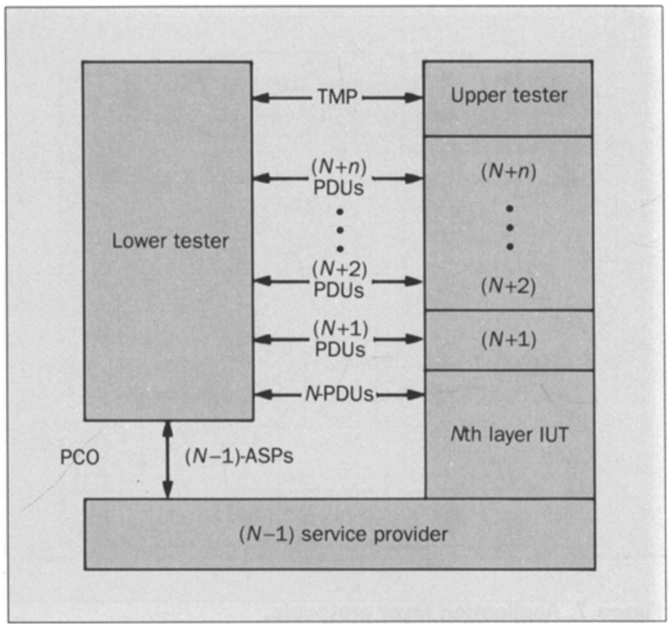


**Figure 6. Coordinated single-layer embedded (CSE) test method.**

The CSE test method indirectly controls the activity of the *N*th-layer IUT's service interface of the protocol *n* layers above the IUT. This test method requires the $(N+n)$ service interface to be exposed so it may interface to the Upper Tester. The other service interfaces within the SUT—e.g., the $(N+1)-(N+(n-1))$ interfaces—are not directly used during testing ***even if they are accessible.*** Note that to test the *N*th-layer IUT, the Lower Tester must act as the peer entity of the SUT's $N-(N+n)$ protocols, and therefore must be able to send and receive $N-(N+n)$ PDUs. The test suite that tests the *N*th-layer IUT is written in terms both the PDUs that must be exchanged between the Lower Tester and SUT at each of the $N-(N+n)$ layers, and the TMPDUs exchanged between the Lower and Upper Testers.

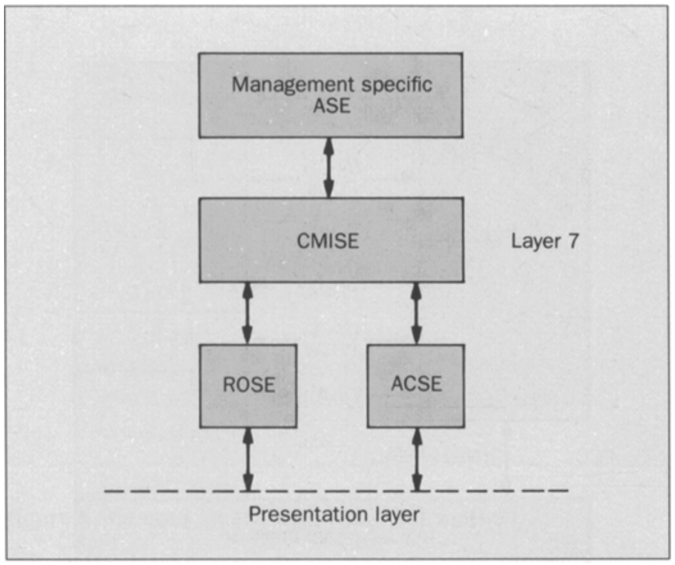


**Figure 7. Application layer protocols.**

The embedded variants of the RS and DS test methods are architecturally similar to those used by the CSE test method (i.e., the Lower Tester acts as the peer entity of the $N-(N+n)$ protocols). The primary difficulty with embedded test methods is that they require complex test specifications and they require general purpose. For example, ISO currently is preparing a test suite for the session protocol embedded under the presentation and FTAM application protocols. This test suite must be written in terms of the session, presentation, and FTAM PDUs exchanged between the Lower Tester and SUT. Thus, it requires specifying the protocol activity at three different layers.

If a different application layer protocol (e.g., the directory access protocol) is used, a new session test suite specified for session, presentation, and directory PDUs will be required. Thus, testing the session layer embedded under the application and presentation layer protocols could require developing a session test suite for testing under each application layer protocol. This is a major flaw of the ISO conformance testing standards. Several ISO and CCITT working groups currently are devising convenient parameterization techniques to allow test cases developed for the embedded test methods to be generically specified. The goal of these groups is to specify embedded test suites that are independent of the application layer protocols by allowing the application layer PDUs to be dynamically read into the test cases.

In addition to the embedded test methods previously described, ISO has defined multi-layer versions of the coordinated, distributed, and remote test methods. These variants allow multiple protocols to be tested simultaneously. Although multi-layer test methods have merit, they have not yet been adopted for use by the standards bodies or by OSI conformance test centers worldwide. Only the National Institute for Standards and Technology (NIST) appears to have done significant research into multi-layer test methods.[7] (For more information about the test methods discussed in this paper as well as other test methods being standardized by ISO, consult ISO DIS 9646-1 and 9646-2.[8,9])

## Choosing a Test Method

The first section uses the CMIP to clarify some of the issues that must be considered when choosing a test method for a protocol.

**A Case Study—The Common Management Information Protocol.** The CMIP is an ISO-standardized application layer protocol that services different network management applications (i.e., fault management, configuration management). Figure 7 shows the relationship between CMIP and other ASEs defined by ISO.

The characteristics of the CMIP are as follows:

- It is a general purpose protocol supporting a variety of network management applications.
- It is user-interactive.
- It supports both confirmed and nonconfirmed services.
- It is an asymmetric, capable of sending and/or receiving information.

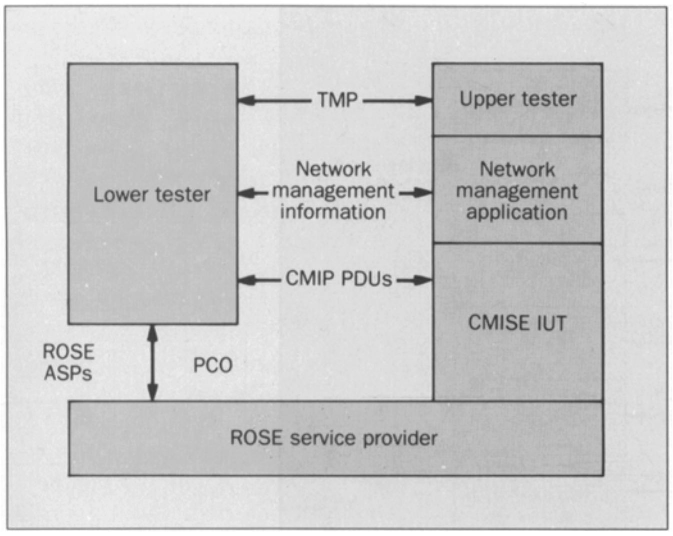


**Figure 8. CSE test method for CMIP.**

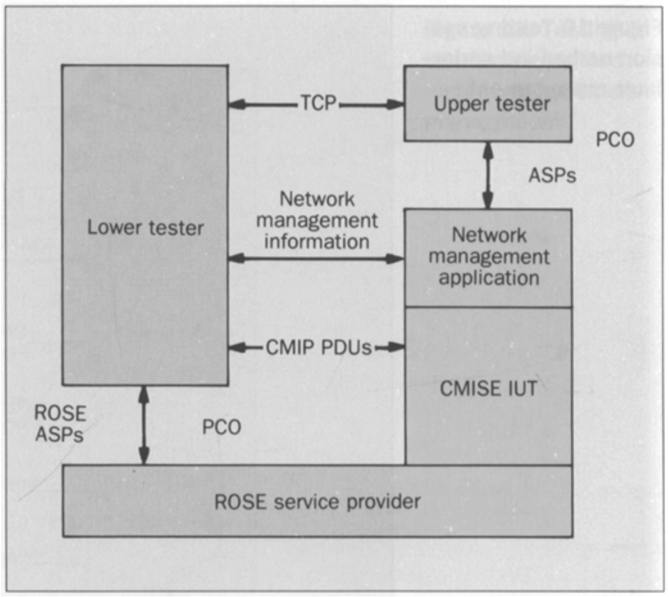


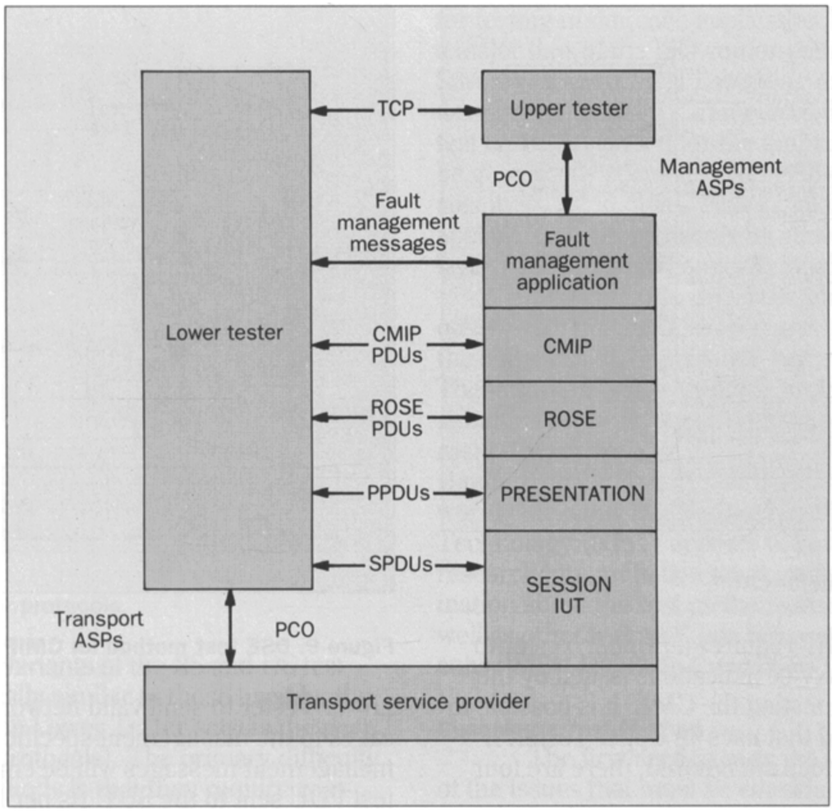
**Figure 9. DSE test method for CMIP.**

Testing the CMIP requires an Upper Tester to observe unconfirmed service indications issued by the CMIP IUT. Thus, when testing the CMIP it is possible to use any OSI test method that uses an Upper Tester. If the multilayer test methods are ignored, there are four test methods:
- Coordinated single-layer (CS) test method
- Coordinated single-layer embedded (CSE) test method
- Distributed single-layer (DS) test method
- Distributed single-layer embedded (DSE) test method, used to test the CMIP.

The remote test methods (i.e., RS and RSE) would be impractical because they do not use an Upper Tester that can observe unconfirmed service indications.

OSI protocol implementors tend to prefer embedded methods because these methods do not need the protocol service interface to be exposed for testing. However, as Figures 8 and 9 show, embedded tests of CMIP using either DSE or CSE test methods require the Lower Tester to send valid network management messages to the management-specific ASE. These network management messages will be embedded in each CMIP test PDU sent to the SUT. As per ISO 9646-3, any such user data sent while executing a CMIP test must be specified in the CMIP test case. Since each network management application may need different information in each type of CMIP message (e.g., the eventData parameter of the M-EVENT-REPORT message is used differently by each network management application), separate CMIP test suites would have to be specified to test the CMIP under each different network management application. Each of these test suites would be written in terms of the CMIP PDUs and the network management messages exchanged between the Lower Tester and the SUT. Consequently, testing the CMIP embedded under a network management application (e.g., fault, configuration, accounting) would require a

**Figure 10. Testing session embedded under fault management.**



different CMIP test suite for testing under each network. Furthermore, since different organizations (e.g., T1M1, OSI/Network Management Forum [NMF], ISO) are defining message sets for network management applications, each CMIP test suite developed using an embedded test method might have to be revised as these message sets are changed and harmonized.

These problems are magnified because the session and presentation layer protocols usually are tested while they are embedded under the application protocols. (Currently, both ISO and COS are planning to provide test suites to test the session and presentation layer protocols embedded under the FTAM protocol. Within the United States, American National Standards Institute [ANSI] representatives agreed that session and presentation layer protocols always will be tested while embedded under the application layer protocols.) Figure 10 shows the configuration used to test the session protocol embedded under a fault management application. As the figure shows, this form of session protocol test requires the test suite to be written in terms of the session, presentation, and CMIP/Remote Operations Service Element (ROSE) PDUs and the *fault management messages* exchanged between the Lower Tester and SUT. If the network management application is changed to a configuration management application (see Figure 11), a

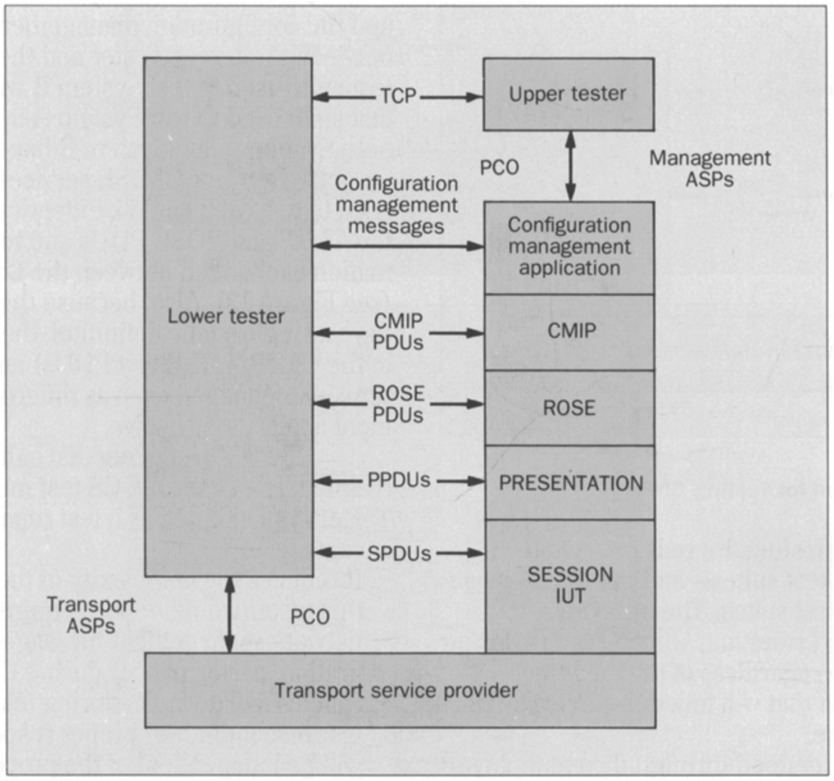


**Figure 11. Testing session embedded under configuration management.**

different session test suite, specified for the session, presentation, and CMIP/ROSE PDUs and *configuration management messages* exchanged between the Lower Tester and SUT, would be required. Similarly, multiple presentation test suites would have to be developed to test the presentation protocol embedded under different network management applications. In addition, each test suite would have to be separately modified and maintained as the various network management applications evolve. Thus, though protocol implementors prefer embedded test methods, it is impractical to use the embedded test methods to test the CMIP.

Because embedded test methods are impractical when testing the CMIP, either the CS or DS test methods should be used. For CMIP testing, the CS test method is preferred, because the test specification must specify an ASE defining the CMIP parameters having Abstract Syntax Notation One (ASN.1) type ANY. Using the CS test method requires that the Upper Tester be implemented as a test responder situated directly above the CMIP IUT (see Figure 12). The Upper Tester substitutes for the network management application in the SUT, and functions as the user of the SUT's common management information service element (CMISE) services during testing. The test responder used in the CS test method uses TMPDUs specified in ASN.1 to simulate the network management application's functions.

By defining an Upper Tester as user of the SUT's

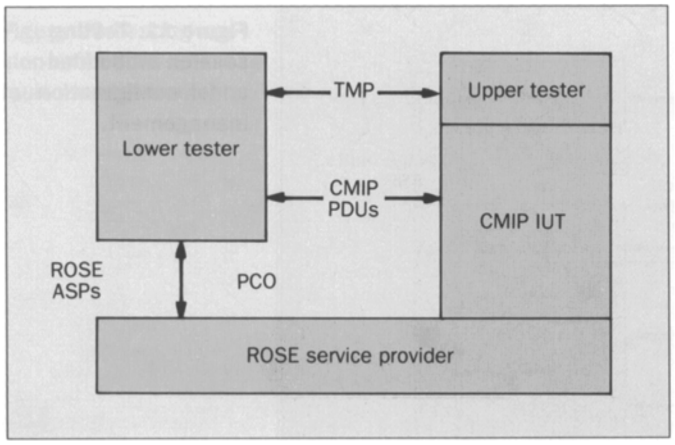


**Figure 12. CS test method for testing CMIP**

CMISE services during testing, there is no need to develop multiple CMIP test suites—and multiple session and presentation layer test suites. The messages exchanged between the Lower and Upper Testers during testing will be the same regardless of the network management application that will function as the CMISE user once testing is done.

This is best understood through the example in Figure 13, where an NMP implementation is shown supporting a fault management application undergoing CMIP testing using the DSE test method. This figure shows the Lower Tester sending fault management messages to the SUT (System A). The CMIP test specification thus defines the CMIP/ROSE PDUs and fault management data exchanged between the Lower Tester and the SUT. If the network management application is changed to a configuration management application (see Figure 14), the messages exchanged between the Lower Tester and SUT (System B) would be CMIP/ROSE PDUs containing information found only in the configuration management application. Here, the CMIP test suite would define the CMIP/ROSE PDUs and the configuration management messages exchanged between the Lower Tester and the SUT. Consequently, test suite used to test System B will be different than the test suite used to test System A in Figure 13. However, if both System A and System B have the same Upper Tester acting as their CMISE services user, the test suite used to test each could be identical, and would consist of the CMIP and ROSE PDUs and test management information exchanged between the Lower Tester and SUT (see Figure 12). Also, because the Upper Tester can have a single, static definition, the TMPDUs (information in the CMIP/ROSE test PDUs) and CMIP test suite can remain unchanged even as different network management applications evolve.

Besides reducing test suite development and maintenance costs, the CS test method with the Upper Tester implemented as a test reponder has the following advantages:

- It reduces the complexity of the Lower Tester, because this tester would not be required to generate all types of network management messages (e.g., fault, configuration, performance) during CMIP testing.
- It helps isolate faults during testing because, by using a test responder, the proper response to a CMIP PDU will be independent of the proper operation of the network management application. Instead, it will require the use of a much simpler test responder.
- It allows the complex task of testing the network management messages and objects to be done separately from CMIP testing.
- It allows protocol stack suppliers to have the stack tested before delivering it to customers supporting network management applications.
- The Upper Tester can be used as a message simulator to do stack-to-stack interoperability testing once conformance testing has been completed.

The primary disadvantages of the CS test method on the CMIP are that it requires vendors to support a special Upper Tester during testing, and that it

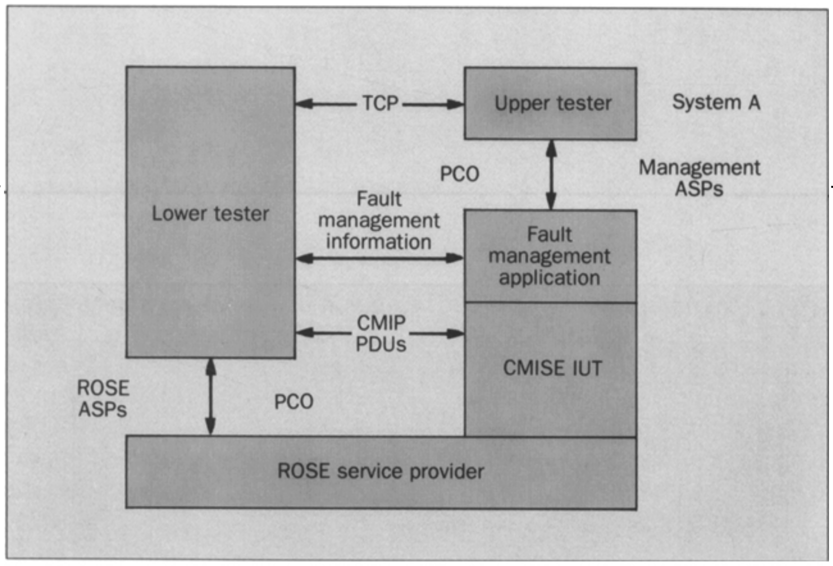


**Figure 13. Testing CMIP embedded under a fault management application.**

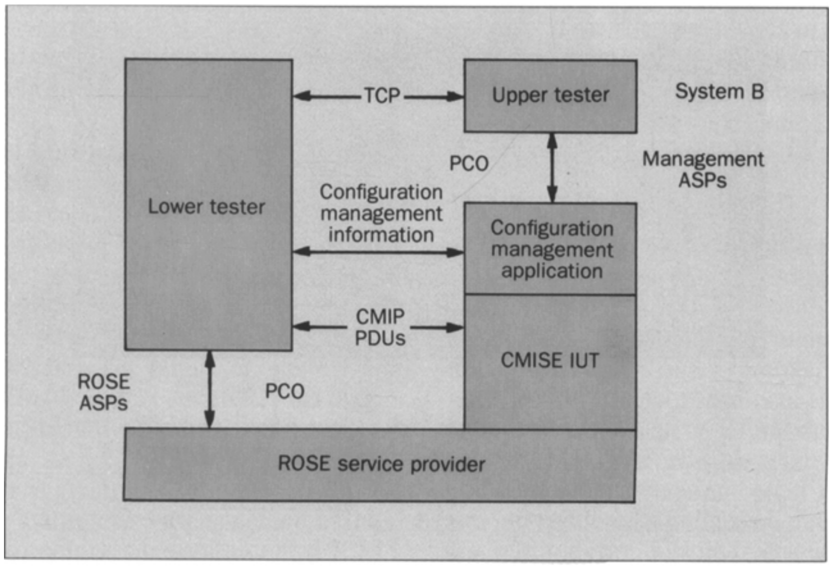


**Figure 14. Testing CMIP embedded under a configuration management application.**

requires developing a TMP and procedures to verify the TMP is valid and sufficient for the purpose of providing synchronization between the Upper and Lower Testers during test execution. In addition, the CS test method assumes the CMIP IUT will be able to receive CMIP PDUs from the Lower Tester. These CMIP PDUs will contain information (TMPDUs) that will allow the Lower Tester to control the Upper Tester's activity. However, problems will arise if the CMIP IUT is only able to send CMIP PDUs. Under these conditions the Lower Tester will not be able to send TMPDUs to the Upper Tester. Therefore, it will be unable to exercise control over the Upper Tester. To use the CS test method, then, CMIP implementors must agree to build the CMIP to function as both the data sender and receiver.

At present, the OSI/NMF has adopted the CS test method for testing the CMIP. A test system using this method is being developed for the OSI/NMF by COS and the Standards Promotion And Applications Group (SPAG) in collaboration with the Fraunhofer Insti-

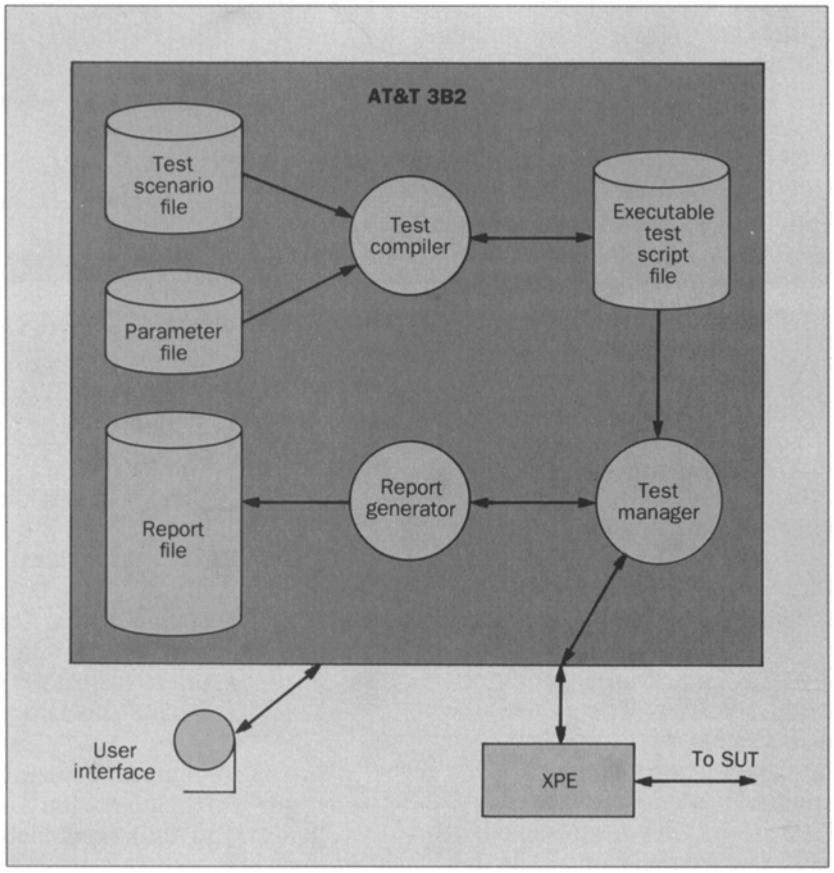


**Figure 15. AXCTS architecture.**

tute. (SPAG is a Brussels-based organization that provides, among other things, conformance test systems and testing services, primarily to European vendors.) The system includes definitions of preliminary architecture and interface specifications for the Upper Tester.

**AT&T's X.25 Conformance Test System**

An example of an OSI conformance test system using the remote single-layer (RS) test method described above is the AXCTS used to test the X.25 Layer 2 and Layer 3 protocols. The RS is especially suitable for testing X.25 implementations because many of these IUTs only execute the X.25 protocols and not those of the higher OSI layers. Since this system uses the RS test method, it consists solely of a Lower Tester.

The AXCTS consists of two hardware components: a UNIX® system-based AT&T computer supporting the test system software, and an X.25 protocol emulator (XPE) providing the X.25 connectivity to the SUT. The test system software consists of test scripts, test compiler, test manager, report generator, and menu-driven user interface. Figure 15 illustrates the relationship between the various modules of the AXCTS.

The AT&T X.25 conformance testing process is achieved by exchanging PDUs between the AXCTS and the SUT. The collection of protocol commands used to drive these exchanges are called "test scripts." These test scripts contain the test cases run by AXCTS. AXCTS provides test scripts for X.25 link layer and packet layer (both permanent virtual circuit and switched virtual circuit) implementations. A test script brings the SUT from the disconnected state to the state under test by executing a sequence of protocol commands known as a *preamble*. It then subjects the IUT to a singular stimulus. All tests for a given protocol state use the same preamble. AXCTS specifies when to transmit a particular frame or packet, and what types of responses are expected in each case. The test scripts are able to check for alternative responses; and they systematically test normal conditions, as well as error conditions of the X.25 system being tested. AXCTS test scripts are executed in the order they appear in a test file. The failure of one test causes the next test in the file to be executed. This process continues until all tests in the file have been run.

To accurately test an X.25 SUT, a customized set of executable test scripts is generated for each SUT. SUT-dependent protocol parameters are specified in the protocol implementation conformance statement (PICS) and protocol implementation extra information for testing (PIXIT) proformas, then stored in files by the test administrator. The test compiler then produces executable test scripts by incorporating the parameter values into the parameterized test scenarios.

The test manager performs the overall test management and I/O control functions for the AXCTS. It sends PDUs specified in the test scripts to the SUT, receives trace information from the SUT, then determines the PASS/FAIL verdict of the test script execution.

The report generator formats the trace output of a test script execution into a report file together with its PASS/FAIL verdict. The PASS/FAIL summary statistics of the entire test file execution is also provided at the end of the report. It gives summary statistics on:

- The number of tests executed
- The number and percentage of tests that passed
- The number and percentage of tests that failed
- The list of tests that failed.

After a test session, the report generator also may be used to condense the failed tests into a separate report, and calculate statistics of an entire protocol layer test suite. The AXCTS also automates the verdict assignment of a test by comparing responses from the IUT with the expected responses encoded in the test script. It issues a "PASS" verdict only if a match is found.

## Conclusion

For OSI to succeed as a truly "open system" standard, conformance test suites and test systems must be developed. The AT&T X.25 conformance test system has shown its value by verifying many of the X.25 interfaces implemented in AT&T products. Conformance test systems for the transport protocol and for OSI application layer protocols such as FTAM and message handling service (MHS) already are available from independent OSI conformance testing centers. In addition, national and international standards bodies are striving to define test methods and test suite specifications for other OSI protocols. As new protocols and their associated tests are standardized, and as implementations appear in the industry, developing conformance test systems to perform these tests will be essential to ensure a cost-effective way to achieve multivendor interoperability.

## References

1. "Proposed Transport Test Management Protocol Specification," ISO/TC 98/SC6/WG4 N336, United Kingdom Contribution to ISO SC6/WG4, Ad Hoc Meeting on Transport Conformance Testing, Paris, France, 16-20 November 1987.
2. "Requirements for a Layer Independent Test Management and

Ferry Control Protocol," ISO/IEP DP 10168-4, ISO/IEC JTC1/SC21 N3689, Association Francaise de Normalisation (AFNOR) Contribution to ISO SC21/SWG on Upper Layer Conformance Testing, 12 July 1989.

3. Darrell Hubbard, "Deterministic Execution Testing of FSM-Based Protocols using the TTCN Model," *AT&T Technical Journal*, Vol. 69, No. 1, January/February 1990, pp. 119-128.
4. H. X. Zeng and D. Rayner, "The Impact of the Ferry Concept on Protocol Testing," *Protocol Specification, Testing, and Verification*, Vol. V, 1986, pp. 519-531.
5. "Requirements for a Layer Independent Test Management and Ferry Control Protocol," ISO/IEC JTC1/SC21/WG1 N680, Canadian Contribution to ISO SC21/WG1 Meeting, Sydney, Australia, December 1988.
6. "Working Draft Answer for Q59—Test Management and Ferry Control Protocols," ISO/IEC JTC1/SC21/WG1 N3243, Output of ISO SC21/WG1 Meeting, Sydney, Australia, December 1988.
7. Richard J. Linn, Jr., Jean-Philippe Favreau, Len Gebase, and Akria Iwabuchi, "An Overview of Formally Specified Multi-Layered Test Systems," National Bureau of Standards, Institute for Computer Sciences and Technology, Gaithersburg, Maryland, January 1988.
8. "OSI Conformance Testing Methodology and Framework Part 1: General Concepts," ISO DIS 9646-1, December 1988.
9. "OSI Conformance Testing Methodology and Framework Part 2: Abstract Test Suite Specification," ISO DIS 9646-2, December 1988.

Biographies (continued)

*products, including defining protocol testing approaches consistent with domestic and international standards bodies. He joined AT&T in 1982 with a B.S.E.E. from the Polytechnic Institute of New York, Brooklyn, and an M.S. in computer engineering from the University of Lowell, Lowell, Massachusetts.* ***Fai Wong*** *is a member of technical staff in the International Operations Systems Engineering Department of the Red Hill facility. She is responsible for systems engineering of the Multi-Function Operations System (MFOS) for the international Postal, Telegraph & Telephone (PTT) market. She joined AT&T in 1983 after earning B.S. and M.S. degrees in mathematics, the former from Davidson College, Davidson, North Carolina, the latter from Carnegie Mellon University, Pittsburgh, Pennsylvania.*

*(Manuscript received July 6, 1989)*