4

TSE-P - a highly flexible tool for testing network management applications using simulation

Dr. Winfried Geyer, Stephan Hesse, Dr. Norbert Newrly
Telenet GmbH
Rhine-Main Office, Marburger Straße 14
D-64289 Darmstadt
phone +49 6151 733-361, -334, -360
fax +49 6151 733-325
email geyer@telenet.de , hesse@telenet.de, newrly@telenet.de

Abstract

In this article we describe TSE-P, a tool for testing network management applications developed by Telenet by order of the Deutsche Telekom AG. TSE-P is specially designed for supporting tests of management applications based on CMIP, but also other interfaces as FTAM and RPC have been integrated. Series of operations to be sent to or expected from the IUT can be formulated in a special test script language. TSE-P holds and maintains the same MIB as the IUT. This makes it possible to generate realistic operations automatically without specifying every detail.

Several information models have been implemented, e.g. customer administration information models, information models for managing ATM, etc. A special feature of the TSE-P are gateway tests, where the reaction of the IUT at an interface is tested when an operation is sent to another interface. Different information models at those interfaces are supported simultaneously.

New information models for CMIP applications can be implemented very quickly by using specially developed ASN.1 and GDMO compilers. For the implementation of behaviour a new prolog-like language called RDL (rules description language) is used, which supports nondeterminism in its decisions.

B. Baumgarten et al. (eds.), *Testing of Communicating Systems*

1 TESTING OF NETWORK MANAGEMENT APPLICATIONS

Within the area of OSI network management the functionality of the communicating systems is partially described by the OSI or national communication standards, i.e. protocol specifications etc. The communication standards describe methods how a managing system (*manager*) queries and modifies a data base (*Management Information Base — MIB*) in the managed system (*agent*) and how the agent notifies the manager of spontaneous changes in its information structure.

The behaviour of the agent when performing operations on the data base is described together with the structure of the data base in a document called the '*Information Model*' or '*Object Model*'. Since this behaviour must describe real world implementations it can be very complex. For example, operations on an object are allowed to have side effects on other objects.

This potentially complex application behaviour requires testing facilities that concentrate on the application and use the communication protocols only as a vehicle to access the implementation under test. (The term 'implementation under test' — IUT — is used to reference that part of the 'system under test' — SUT —, that is the target of the test. For example, the communication protocol stack is part of the SUT, but not of the IUT. See also [X.290] for a definition of these terms.)

1.1 Testing by Simulation

The testing strategy of TSE–P is based on simulation. That means that the behaviour of the test system is mainly controlled by a simulator of the real application. This is in contrast to other testing approaches where the behaviour of the test system is only controlled by a sequence of (automatically or manually generated) test scripts. While this seems appropriate for testing low level protocols, its weakness becomes evident when testing complex behaviour of applications:

- In some cases, the described behaviour can be achieved in different ways. For example changing two attributes in an object can either be done with a single SET-operation giving a new value to both attributes or with two SET-operations, each changing the value of one attribute. Traditional testing based on [X.290] (ISO 9646) requires all possible combinations to be specified in the test scripts.
- The effect of an operation depends on the state of the MIB. When performing tests using a fixed sequence of test scripts, any possible state of the MIB has to be taken into account. Since the MIB is a 'potentially infinite' finite state machine, much effort has to be spent to describe the PDUs (protocol data units) to configure the MIB correctly. In contrast, testing by simulation keeps track of the state of the MIB by holding and

maintaining a copy of the data base. This also enables the TSE-P to automatically generate test operations based on the current state of the MIB.

- The international standards describe methods for event reporting and log control. TSE–P has a module for generating appropriate event reports and log records in accordance to the current MIB. The expected behaviour of a IUT regarding event reporting and log control doesn't need not to be specified in appropriate test scripts.

The role of test scripts in TSE–P is reduced to influencing the behaviour of the simulator. This leads to a significantly reduced number of test scripts and to less complex test scripts. Nevertheless, if necessary TSE–P test scripts may be detailed up to a level which is used by other test approaches.

2 TEST CONFIGURATIONS

The test system can be used in three different configurations: test of a manager, test of an agent and gateway test.

2.1 Testing a Manager

The principal test configuration for testing a manager is shown in figure 1. In this configuration the test system contains a component which simulates the agent upon which the IUT works.

The generating capabilities of the TSE–P can be used to specify (under test script control) valid operations to be executed by the test operator on the IUT's user interface.

This test configuration may also be used to simply execute operations sent by the manager. This application is particularly useful for implementation support (or basic interoperability testing) where the manager only needs a communications partner ('sparring partner') without the need to control the whole test flow by the test system.

All operations are checked against the specifications for protocol and application and — after passing all the checks — are used to modify the state of the agent.

In general, testing comprises the following steps:

- The test script specifies an action that the manager is expected to perform on the agent. This action is potentially complex so that the manager needs more than a single operation to perform the action. In most cases the exact sequence of operations is not well-defined and may be chosen by the manager.
  The action specification in the test script may be incomplete in that some or all of the

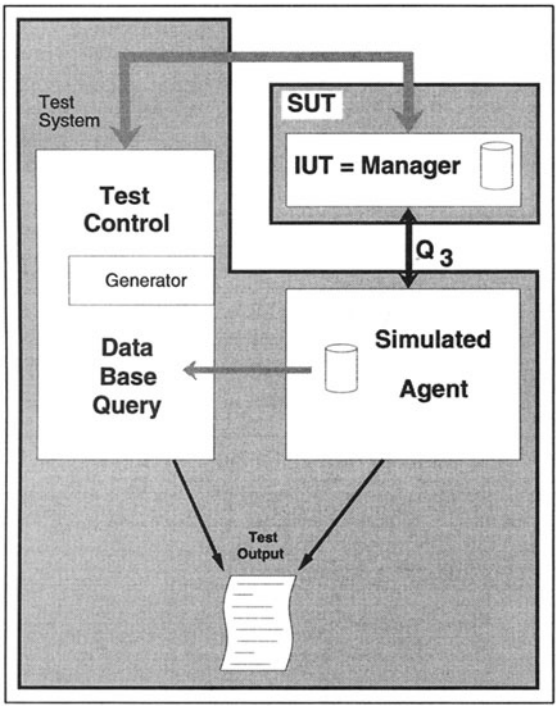

Figure 1 Test configuration for testing a manager.

parameters can be left unspecified. A lookup in the data base of the simulated agent is used to complete such action specifications.

- The request to perform the action is communicated to the manager. This may be done — if technically possible — by data communications (online) or through the help of an operator.
- The request is transformed by the IUT into a sequence of operations which is transferred via the Q_3-Interface to the simulated agent (Q_3 is a reference point between manager and agent in OSI network management).

- The simulator checks and executes the operations, which causes a change on its state represented in its data base.
- The simulator generates appropriate results and sends them back to the IUT. Test scripts may be used to verify the receipt of special operations or to predefine special results.
- On receipt of some trigger event like the success message from the manager the test control component initiates a query on the data base of the simulator to verify that the changes in the data base correspond to the expected ones.
- Differences between the expected and the actual state of the data base as well as errors detected during the check of the operations are flagged in the test output as potential errors.
- At the end of a test a summary of the encountered errors and warnings is generated.

2.2 Testing an Agent

In the test configuration for testing an agent, TSE-P acts in the role of a manager (see figure 2). Although it looks quite different from the test configuration given in figure 1, the main components are the same.

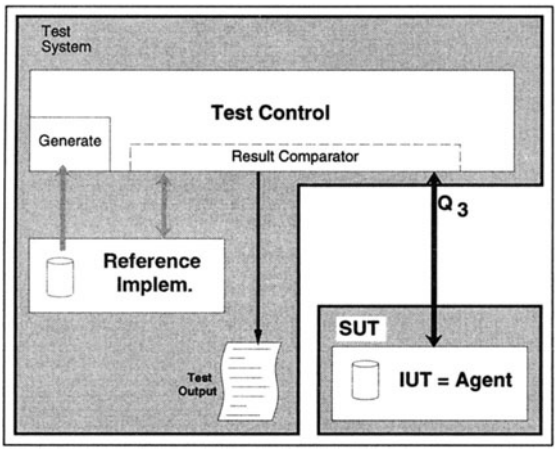

Figure 2 Test configuration for testing an agent.

The simulated agent is used in this test configuration as a reference implementation. It keeps track of the real agent's state information. This is accomplished by executing each operation on the real agent and on the reference implementation.

Testing an agent is always controlled by a test script. The elements in the test scripts are essentially the same, but they are interpreted in a slightly different way:

- The test script specifies an action which has to be performed on the agent. This action may be realized as sequence of operations on the agent's data base.
  The action specification in the test script may be incomplete in that some or all of the parameters can be left unspecified by the test script author. A lookup in the data base of the simulator is used to complete such incomplete action specifications.
- The action is converted into a sequence of operations. These operations are executed on the IUT via the Q_3-Interface and on the reference implementation using an internal interface.
- Both agents generate results to the operations which are compared. Differing results indicate deviations of the behaviour of the IUT from the specification. Such differences are flagged as potential errors in the test output.
- At the end of a test a summary of the encountered errors and warnings is generated as test result.

2.3 Gateway Testing

In the gateway test configuration, both TSE–P and the IUT have to play simultaneously the role of a manager on one interface and the role of an agent on the other interface. In general, a test comprises the following steps:

- First, TSE-P plays the role of a manager, sending operations to the IUT at interface 1. The IUT receives these operations in the role of an agent.
- The IUT transforms the received operations into other operations, which are sent in the role of a manager to the TSE-P at interface 2.
- The TSE-P receives these operations in the role of an agent, processes them, and sends results to the IUT.
- The IUT receives the results, and, taking them into account, decides what to do with the original operations which it received as an agent. It processes these operations and sends the results to TSE-P on interface 1.

When deciding if the IUT acts correctly, the TSE-P could simply generate the operations it expects from the IUT and compare these with those actually received. However,

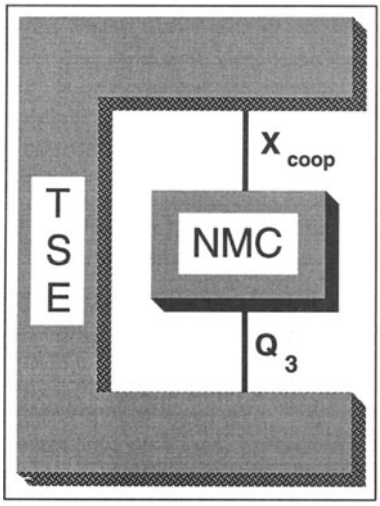

Figure 3 Example of a gateway test configuration (see chapter 4).

these operations as well as their order are not explicitly specified. It depends on the implementation of the manager and not on the interface specification. Hence, it was decided not to compare the operations, but the effects they have on the MIB.

The gateway test is therefore performed like this:

- When the TSE-P sends the operations to the IUT in the role of a manager, it generates operations which the IUT is meant to generate and send to the TSE-P at interface 2. These operations are processed on a so called "reference MIB", which is kind of a copy of the current MIB reflecting the MIB of the (interface 2) agent.
- The operations sent by the IUT on interface 2 are processed on the current MIB.
- After all results have been sent and received, the reference MIB and the current MIB are compared. Relevant differences are displayed.

3 RUNNING A TEST

Telenet TSE–P runs on a UNIX machine with a colour graphics terminal. It uses X window and OSF/Motif for presentation of the user interface.

After startup the TSE–P main window is presented (see figure 4). It allows various

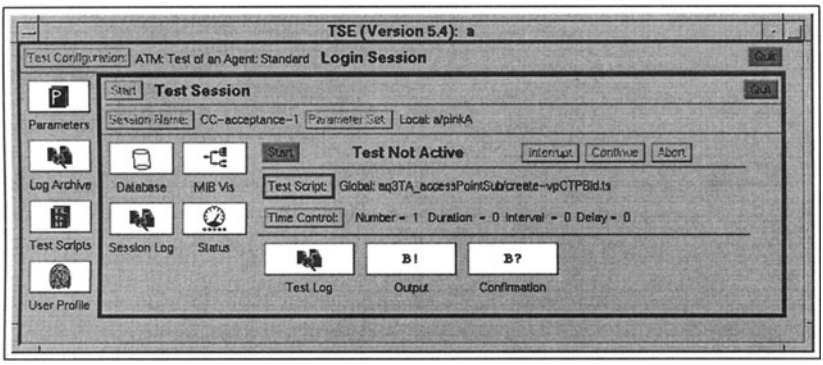

Figure 4 Main window of TSE–P application.

manipulations on different objects in the test system like test scripts, data bases, test documentation archives etc.

3.1 Test preparation

Before starting a test it has to be verified that the MIB of the TSE-P and the MIB of the SUT are equivalent. If necessary, an appropriate data base has to be loaded or generated. The address parametrisation has to be adjusted to the address of the SUT by using either a text editor or the TSE–P parameter set generator.

Furthermore a test script has to be selected from the set of predefined test scripts or has to be written in the user's work area. TSE–P test scripts are normally quite simple since the user may (and normally will) choose to specify only very few parameters. All unspecified parameters will be filled in during the execution.

Figure 5 shows a typical TSE–P test script. The header serves to document the test script. The body simply says 'create an object of class vpCTPbidirectional with any name

```
HEADER {
  PURPOSE =
        "Create a vpCTP-Bidirectional object "
        "         MANAGED OBJECT CLASS = vpCTPbidirectional "
        "         MANAGED OBJECT INSTANCE will be generated "
        "         Attribute list will be generated ";
  KONFIG  = AGENT;
};


STEP "create vpCTP-bidirectional" (REPEAT=0) {
  BLOCK () {
    CMISE () CREATE {
      MANAGED OBJECT CLASS = "vpCTPbidirectional";
      MANAGED OBJECT INSTANCE = *;
      ATTRIBUTES = *;
    };
  };
};
```

Figure 5 Typical TSE–P test script.

fitting to the current MIB and with any attributes that are valid in the current state of the IUT'.

Test scripts may either be edited using a normal text editor or using a special interactive tool, the TSE test script generator (see figure 6).

3.2 Test execution

During test execution the test system processes the test script and generates valid values for all incompletely specified parameters (e.g. an asterisk means 'any or omit'). The generated operation is transmitted to the IUT and also processed by the test system itself. The result received from the IUT is compared with the own result.

Figure 7 presents an example of the produced test output. It consists of a basic output window with a one-line informative message per event. More detailed information is available on every test output line. It can be viewed by clicking on it. A special navigation tool aids in quickly locating the relevant test output lines.

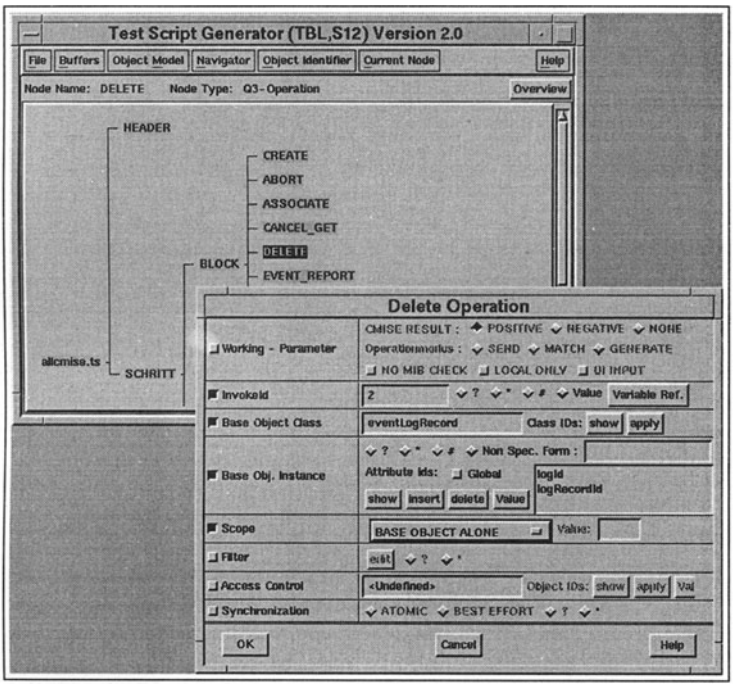

Figure 6 The TSE test script generator.

3.3 Interactive MIB browsing

During or after the execution of a test it is possible to browse interactively through the MIB. The MIB visualisation (see figure 8) presents the tree structure of the MIB and allows the user to focus on specific objects and to display their attributes.

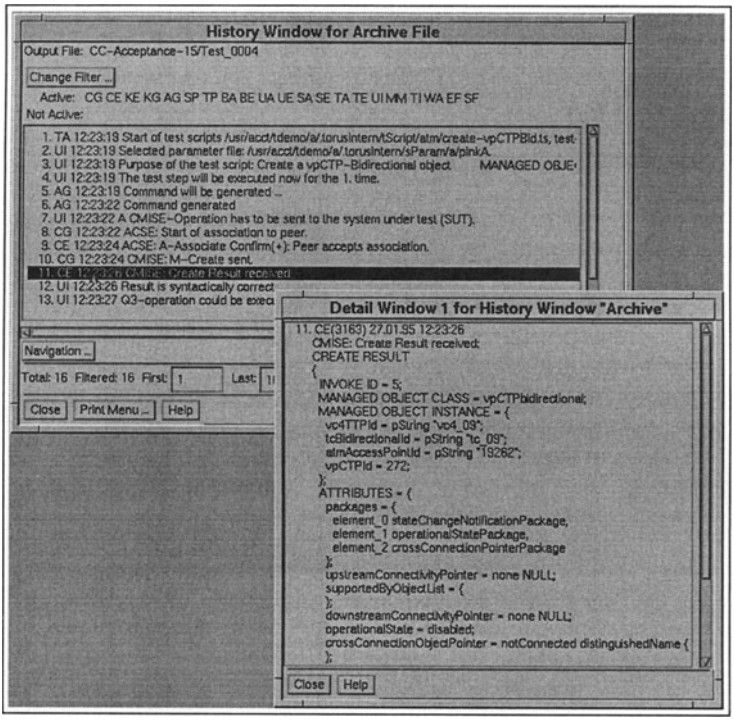

Figure 7 Sample test output of TSE–P.

4 TECHNICAL DEVELOPMENT, USAGE, PRACTICAL EXPERIENCES

Development of the TSE-P started in November 91, when Telenet was asked to support the Deutsche Telekom AG (Deutsche Bundespost at that time) to specify an information model for customer administration, embedded in a project called SuB ("System-unabhängige Betriebsführung"). An appropriate test system, a first version of the TSE–P, was implemented until June 92. In addition to the facility of testing agent or manager, it

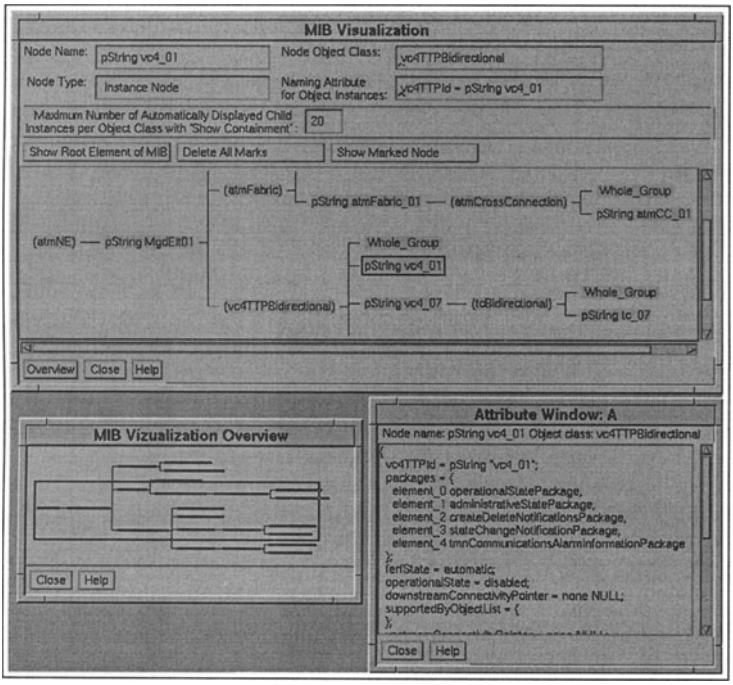

Figure 8 TSE–P MIB visualisation window.

already had the capability to perform gateway tests. Customer administration requests were sent in a customer-specific format to the IUT and the transformation into correct Q_3-operations was checked.

The test system was not only used by the Deutsche Bundespost for acceptance testing, but also by the providers of the manager and the agent systems.

Both the information model and the behaviour of the information model were implemented hard coded in C++.

For the next information models to be implemented, the new prolog-like programming language RDL (rules description language) was designed to implement the behaviour. In contrast to prolog, RDL supports nondeterminism for generating test cases

randomly. The information model itself (supported classes and their attributes, etc.) still was coded in C++. The information models delivered with this new version in November 93 were TBL ("Technik für Betriebslenkung"), MML ("man machine language") and VMZ ("Verkehrsmeßzentrum"), dealing with configuration and performance management.

In April 94, Telenet was asked to develop a test system for network management within a bilateral ATM pilot project of the Deutsche Bundespost and France Telecom. For flexibility reasons, it was decided to provide the knowledge about the information model not as compiled C++ code, but as precompiled input to be interpreted during runtime. These interpretable files are generated by specially developed ASN.1 and GDMO compilers.

NMCs (network management centers) and ATM-NEs (network elements) managed by them were to be tested. The NMCs communicate via an CMIP interface called Xcoop, and with the ATM-NEs managed by them via a Q_3 interface. The information models mainly deal with the reservation of bandwidth. Besides testing an ATM-NE and each interface of an NMC separately, also a gateway test of an NMC was implemented.

TSE-P for ATM was delivered in July 95. Using ASN.1 and GDMO compiler and parts of the already coded behaviour, different versions of these information models could be implemented very quickly.

The TSE-P for ATM was ordered in a very early stage of development of the information model. While implementing th information model for the test system, many errors, questions and obscurities have been detected within the specifications, which led to changes in the involved interface specifications. So the TSE-P not only served as a test tool or as reference system for the real system, but also as a pilot implementation of a new information model.

New versions for customer administration information models were implemented, now also using the new mechanisms described above. Further developments also deal with other interfaces, as FTAM and RPC.

While implementing the information models mentioned above, several standards had to be supported, such as ISO 9596/ITU X.711 CMIP, ISO 10040/ITU X.701 Systems Management Overview, ISO 10164-1/ITU X.730 Object Management Function, ISO 10164-2/ITU X.731 State Management Function, ISO 10164-4/ITU X.733 Alarm Reporting Function, ISO 10164-5/ITU X.734 Event Management Function, ISO 10164-6/ITU X.735 Log Control Function, ISO 10164-13/ITU X.738 Summarization Function (partially), ISO 10164-11/ITU X.739 Work Load Monitoring Function (partially), ISO 10165-2/ITU X.721 Definition of Management Information, ISO 8650/ITU X.227 ACSE, ISO 9072-2/ITU X.229 ROSE, ISO 8823/ITU X.226 Presentation, ISO 8826/ITU X.225 Session, ISO 8073/ITU X.224 OSI Transport Class 0, ISDN S_0, ISDN S_{2m}, X.25, etc.

As described above, ASN.1, GDMO and RDL compilers have been developed. There is a large set of RDL rules which covers behaviour which is contained in every information model following [X.711]. Moreover, there is a mechanism to generate RDL rules auto-

matically, which cover the part of information model knowledge formulated in GDMO. New rules only have to be written for information model specific informally formulated behaviour (see figure 9).

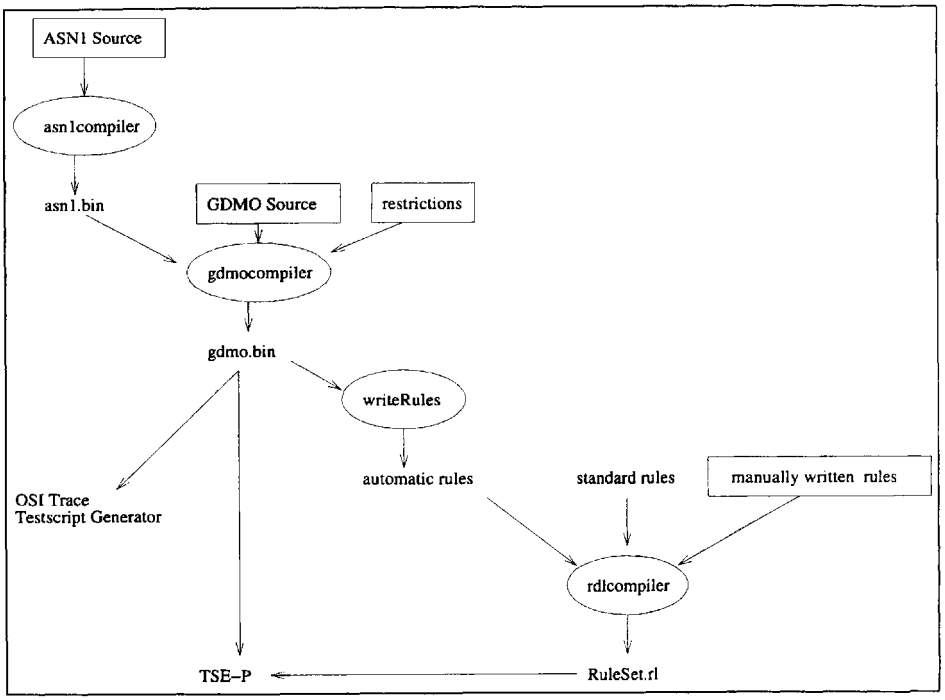

Figure 9 Implementing new information models.

For the implementation of a new information model (see chapter 5) these tools and mechanisms permit an easy and quick implementation of the formal part of the GDMO description. After this implementation step formal tests are already possible. In a following implementation step RDL rules have to be written representing the informally described behaviour. Some work has to be done here, but once this behaviour is implemented, it

does not need to be formulated in eventually many test scripts (which also can lead to mistakes), but is tested and simulated automatically by the test system.

5 IMPLEMENTING NEW INFORMATION MODELS

Implementing an new information model means to adjust the TSE–P to all application specific informations. This information may be divided into three different parts:

- abstract syntax information (usually described in ASN.1),
- MIB data base schema layout (usually described in GDMO) and
- MIB behaviour specifics (usually described in the non-formal parts of GDMO).

TSE–P represents the behaviour description as a set of rules expressed in RDL. These rules are used for generating valid behaviour as well as validating behaviour of the IUT.

TSE–P encompasses tools that read ASN.1 and GDMO information and generate the relevant internal descriptions. The rule set for an information model consists of three parts:

- General rules for every information model (e.g. a CMISE-operation should have an invoke id),
- automatically generated rules from the formal GDMO description (e.g. there is no SET operation allowed on a specific attribute),
- and manually written rules in which the behaviour specifics are encoded (e.g. the creation of an object instance assumes the existence of another instance with specific properties).

A compiled rule set can be generated without having to write rules manually. This rule set only covers the behaviour as described in the standard rules and in the rules generated automatically. It can be seen as a prototype of the desired implementation. New rules can be added describing parts of the informally described behaviour, until the implementation is complete. In this way, rapid prototyping is possible.

REFERENCES

[X.25] CCITT (1988), Interface between DTE and DCE for terminals operating in the packet mode and connected to public data networks by dedicated circuit (ISO 8208).
[X.208] CCITT (1988), Specification of Abstract Syntax Notation One (ASN.1) (ISO 8824).
[X.209] CCITT (1988), Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) (ISO 8825).
[X.224] CCITT (1988), Transport protocol specification for open systems interconnection for CCITT applications (ISO 8073).
[X.225] CCITT (1988), Session protocol specification for open systems interconnection for CCITT applications (ISO 8327).
[X.226] CCITT (1988), Presentation protocol specification for open systems interconnection for CCITT applications (ISO 8823).
[X.227] CCITT (1988), Association control protocol specification for open systems interconnection for CCITT applications (ISO 8650).
[X.229] CCITT (1988), Remote operations: protocol specification (ISO 9072-2).
[X.290] CCITT (1988), Conformance testing (ISO 9646-1).
[X.291] CCITT (1992), Abstract test suite specification (ISO 9646-2).
[X.292] CCITT (1992), TTCN (ISO 9646-3).
[X.701] CCITT (1992), Systems management overview (ISO 10040).
[X.711] CCITT (1991), Common management information protocol specification for CCITT applications (ISO 9596-1).
[X.721] CCITT (1992), Definition of Management Information (ISO 10165-2).
[X.722] CCITT (1992), Guidelines for the definition of managed objects (GDMO) (ISO 10165-4).
[X.730] CCITT (1990), Object Management Function (ISO 10164-1).
[X.731] CCITT (1990), State Management Function (ISO 10164-2).
[X.733] CCITT (1992), Systems Management: Alarm Reporting Functions (ISO 10164-4).
[X.734] CCITT (1992), Event Report Management Function (ISO 10164-5).
[X.735] CCITT (1992), Systems Management: Log Control Function (ISO 10164-6).
[X.738] CCITT (1993), Summarization Function (ISO 10164-13).
[X.739] CCITT (1993), Workload Monitoring Function (ISO 10164-11).
[Prolog] W.F.Clocksin and C.S.Mellish, Programming in Prolog, Springer Verlag, 1984