CEC TEN-T ATM Task UK/96/94

ACCESS

ATN Compliant Communications

European Strategy Study

Define Interoperability Test Tools

Document Reference	: ACCESS/DFS/264/WPR/032
Author	: Ian Nicholls
Revision Number	: 1.1
Date	: 25 November 1998
Filename	: D032I1-1.DOC

DOCUMENT CONTROL LOG

Revision Number	Date	Description of Change
0.1	24 April 1998	first draft
1.0	28 October 1998	minor revision of Version 0.1, presented for acceptance by the Consortium
1.1	25 November 1998	revision of Version 1.0 as a result of comments from NATS and discussions at Meeting 12. In addition to minor editorial corrections, the following sections have been modified: 1.2 (2 new paragraphs at end) 2.1 (new section) 2.2 (third bullet extended) 2.3.3 (new paragraph at beginning) 3.1.3 (new paragraph at end) 3.2.3 (sentence extended) 4.1.1 (new paragraph at end) 4.1.2 (second paragraph modified) 4.2 (additions to “reasons” column in table) 4.2 (new paragraph at end) 4.3 (new paragraph at end) 4.5 (new paragraph at end)

EXECUTIVE SUMMARY

ACCESS Phase 2, Part 2, elaborates in detail the strategy, methodology and scenarios for AMHS interoperability testing together with test specifications and the test schedule. Within the scope of this extensive testing exercise, the question as to the usefulness of hardware and software test tools arises, in order to make the testing more efficient and of a better quality. This question is analysed in this document and a number of recommendations are made.

First, the benefits which might be gained through the use of test tools are described. It is concluded that, from the viewpoint of efficient and reliable communications, automated test tools could be highly beneficial in the AMHS interoperability testing environment.

In order to progress the analysis, the services which might be provided by test tools are described and categorised without any consideration of the implementation of the test tools themselves. This is done so that, in the light of the effort which might be justifiable, recommendations can be made on an appropriate subset of services to be implemented in test tools. The recommended subset of services consists of:

- interact with test database
- support the test operators in controlling test execution
- communicate with remote test operators
- generate test data from test cases
- support test operator in recording test results
- maintain test case database
- maintain test results database
- generate test reports.

Concerning the implementation of these services, it is recommended that the test tools consist of one central computer installation with remote access by test operators. A list of functional and non-functional requirements placed on the system is given.

TABLE OF CONTENTS

1. INTRODUCTION 1

1.1 Scope 1

1.2 Purpose of Document 1

1.3 Document Structure 2

1.4 References 2

2. POSSIBLE BENEFITS OF THE USE OF TEST TOOLS 3

2.1 General Benefits of Test Tools 3

2.2 Possible Benefits for AMHS Interoperability Testing 4
2.2.1 Testing Interfaces 4
2.2.2 Distribution of Systems in the Test Configurations 5
2.2.3 Test Case Definition and Maintenance 5
2.2.4 Test Documentation to be Produced 5

3. POSSIBLE SERVICES PROVIDED BY TEST TOOLS 6

3.1 Test Operator Interface 6
3.1.1 interact with database 6
3.1.2 control test execution 6
3.1.3 communicate with remote operators 6

3.2 Interface to IUT and other Systems 6
3.2.1 interface for active control of IUT or other system 7
3.2.2 interface for retrieving information from IUT or other system 7
3.2.3 cause changes in network 7

3.3 Test Execution 7
3.3.1 generate test data from test cases 7
3.3.2 execute logic of the test cases 7
3.3.3 record results 7

3.4 Database Maintenance 7
3.4.1 test case maintenance 7
3.4.2 results data maintenance 7
3.4.3 report generation 8

4. STRATEGIES FOR TEST TOOL PROCUREMENT 9

4.1 General principles 9
4.1.1 Justifiable Development Effort 9
4.1.2 Basic Approach 10

4.2 Recommended Test Tool Services 10

4.3 Centralised versus Decentralised Tool Configuration 11

4.4 Functional Requirements **12**

4.5 Non-functional Requirements **13**

1. Introduction

1.1 Scope

The document discusses and makes proposals for the test tools, including computer hard- and software as well as communication facilities, for supporting AMHS interoperability testing. As one of the deliverables of ACCESS Phase 2 Part 2, it is concerned only with AMHS *interoperability* testing. Although the project plan, as revised, also contains Work Packages dealing with conformance[1] testing, these are not addressed here.

1.2 Purpose of Document

The contents of the document are, for a number of reasons, rather exploratory and speculative in nature:

- methods for interoperability testing are not subject to standardisation and depend highly on the protocols / interfaces to be tested and on the test configuration;
- to the authors' knowledge, no general, (industry) standard equipment is available for this purpose;
- the subject of test tools is not exhaustively treated in other protocol testing environments.

On the other hand, the conditions under which interoperability testing is to be performed within the ACCESS context are well defined, so that it should be possible to make definite statements on the use of tools.

The purpose of the document is therefore to identify the possible benefits of the development and use of tools in AMHS interoperability testing and to recommend a strategy, if applicable, for this.

Because of the exploratory nature of the topics discussed here, no concrete work package description had been produced during the ACCESS scoping phase for this WP. With the benefit of experience gained in other ACCESS Part 2 WPs, "Define Interoperability Test Tools" was interpreted to mean a definition of *requirements* to be placed on test tools which might be employed during AMHS interoperability testing *as defined elsewhere in the Part 2* WPs. This requirements definition would, for example, be sufficient to commence a procurement process for the tools.

AMHS interoperability testing as defined in ACCESS Part 2 does not consider dynamic and quantitative aspects of the systems under test such as throughput times, behaviour under load, maximum message capacities etc. While recognising that these are important features of network components which should be tested and for which appropriate test tools would be necessary, it is not feasible to investigate requirements placed on the tools without a definition of the tests to be carried out. For this reason, test tools which would support the dynamic and quantitative testing of AMHS components are not considered here. Their complexity and cost would certainly be considerably more than that of the tools envisaged in

[1] For the distinction made within ACCESS between interoperability and conformance testing, see [A270].

this WP. This is due to the fact that real-time testing is much more ambitious than the testing of the logic of communications software.

1.3 Document Structure

The document is structured as follows.

Chapter 2 identifies the possible benefits to be gained from the development and use of tools.

Chapter 3 considers the functions which it might be useful to support by means of tools.

Chapter 4 analyses and makes recommendations on strategies which could be employed for this purpose.

1.4 References

Reference	Title
[A260]	WP260 Define Trials Objectives
[A261]	WP261 Define Operating Scenarios
[A262]	WP262 Produce Test Specification
[A263]	WP263 Produce Test Schedule
[A264]	WP264 Define Interoperability Test Tools
[A265]	WP265 Configure Trials Scenario
[A266]	WP266 Conduct ATSMHS Trials
[A270]	WP270 Conformance Test Requirements
[A271]	WP271 Conformance Test Specification
[ICAO1]	ICAO, Aeronautical Telecommunications Network (ATN), Standards and Recommended Practices (SARPs), Sub-Volume 3, Ground-Ground Applications, Version 2.2, January 1998
[ICAO2]	Guidance Material on [ICAO1]
[ICA16]	ATSMHS SARPs
[ICA17]	ATSMHS Guidance Material

2. Possible Benefits of the Use of Test Tools

This chapter looks at the benefits which could possibly be gained through the use of tools in AMHS interoperability testing. It does not consider the functions themselves or their implementation and deployment - see the following chapters.

2.1 Definition

In the context of this document, “test tools” are taken to mean systems consisting of hard- and software which are implemented, procured and operated for the purpose of supporting the test operators and making the testing process more reliable and efficient. Test tools are normally distinct from the systems being tested and are not normally used outside of the interoperability testing phase of network implementation.

2.2 General Benefits of Test Tools

This discussion is restricted to a consideration of possible benefits in a general context and not specifically within the context of AMHS interoperability testing.

The following features could be used to characterise interoperability testing in general.

- the distributed environment in which parties involved in the testing are not located at one site and need to communicate efficiently and accurately and in a fashion which allows the recording of their interactions;
- the need to repeat sets of interoperability tests when new (releases of) protocols have to be tested or when new network configurations make this necessary (“regression testing”);
- test specifications which are subject to continual extension and modification, for example in the case of when, during operations, new difficulties are encountered which should have been identified and excluded in interoperability testing (testing is, by definition, always incomplete!);
- possibly large data volumes (test sequences, test results and their summaries);
- the need to produce concise, consistent and accurate documentation.

Corresponding to these points above, the following benefits might be achieved by the use of tools:

communication in the distributed environment: standardisation and automisation of the interactions among operators performing the tests so that dependencies among interactions in the test cases can be reliably executed;

repeatability of tests: provision of means for exact repetition, automatically comparing result with former results;[2]

[2] Originally AMHS interoperability testing was planned to be a one-time activity. However this is not likely to be true, with tests being repeated over a long period of time.

maintenance of test specifications: performing of version control on test specification databases;

administration of data: storing and manipulation (by means of dedicated database programmes) of test data, including (possibly) the automatic capture of raw test data from the test execution;

document preparation and maintenance: automatic creation of reports from the test input and results.

Overall, interoperability testing is a rather "expensive" operation in terms of manpower and elapsed time for co-ordination, test execution and result analysis. However it is essential before taking new systems and protocols into operation. Any reduction in effort made possible by the use of support tools should be seriously considered in order to reduce overall costs.

A further important aspect is the improved overall quality of tests and their documentation to be expected when tools are employed: tools have the effect of formalising the definition and execution of tests.

2.3 Possible Benefits for AMHS Interoperability Testing

The general benefits identified in the previous section are now considered specifically in the context of AMHS testing whose objectives and scenarios are defined in [A260] and [A261] respectively.[3]

2.3.1 Testing Interfaces

"Testing interface" is used here to mean an interface at which an operator (or possibly a system external to the test configuration) interacts with the IUT or other system in the test configuration. Each test scenario involves interactions at two (or more[4]) interfaces. Such interactions can be classified as "active" (the operator or external system initiates an action at the interface in the IUT or other system) or "passive" (the operator or external system expects to see specific results at the interface).

Active interactions need to be scheduled, e.g. in the case of sending responses confirming message receipt. Passive interactions need to be analysed and correlated with the active interactions and with each other in order to yield the result of the test.

One such active interaction is with the network (OSC-GW-14, Network Failure and Recovery).

[3] [A261] lists in its Section 3.1 the basic equipment necessary for the execution of the test scenarios as specified there. The "test tools" considered in this present document are possible additional, but not essential equipment.

[4] Interfaces for message submission/retrieval and those e.g. for retrieving log information should be considered separate here.

Whereas this number of interfaces involved here is rather restricted and the complexity low,[5] it is nevertheless considered that automated test tools could provide useful benefits in increasing the effectiveness and quality of test execution.

2.3.2 Distribution of Systems in the Test Configurations

The scenarios defined are not specific about the location of systems contained in the test configurations. In the case of AMHS interoperability testing however, it is highly likely that the systems involved will be distributed across Europe, i.e. in each configuration (with the possible exception of configuration 5 - one MS with two UAs) at least one system will be remote from others.

It is considered that, from the viewpoint of efficient and reliable communications, automated test tools could be highly beneficial in the AMHS interoperability testing environment.

2.3.3 Test Case Definition and Maintenance

This is an off-line activity[6] by means of which the test case definitions (produced in ACCESS, for example, with word processors and documented on paper) are entered into the test tools and maintained there in a machine-processible form.

The logic of AMHS interoperability test cases is rather straightforward with few branches etc. It is not expected that support tools for the maintenance of test case definitions and e.g. the generation of messages to be sent and those to be expected would bring about a significant increase in efficiency.

2.3.4 Test Documentation to be Produced

[A261] lays out a structure for test reports. This could be usefully derived from test results maintained in a more extensive database in an automatic way.

[5] Much more complex configurations could have been envisaged!

[6] by comparison, for example, with the test execution, which would be “on-line”.

3. Possible Services Provided by Test Tools

In this section, the “interoperability test tools” are considered as a black box with no regard to their implementation. The possible services supplied by them at the boundary of the black box are identified and structured into four groups. A possible subset of these which might usefully be implemented is recommended in the following chapter. The logical model and the nomenclature used is shown in Figure 1.

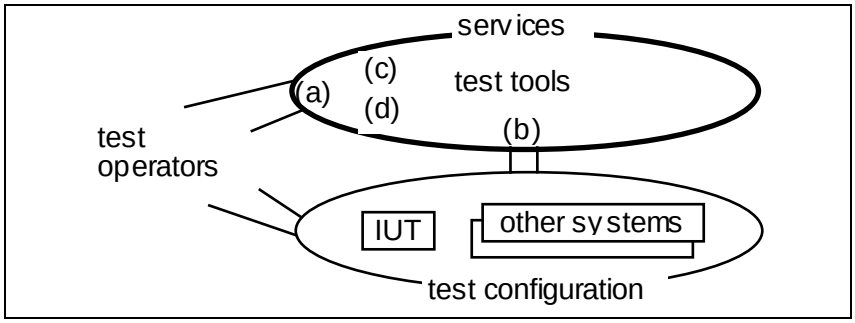

Figure 1: Logical model and nomenclature. For (a) - (d) see following text.

3.1 Test Operator Interface

This set of services involves the interface of the test tools with the test operators. On this interface three sets of services could be provided. See (a) in Figure 1.

3.1.1 interact with database

A major part of the test tool functions could be a database containing test cases, results, configuration data etc. - see section 3.4. On an appropriate interface these functions could be made available to test operators.

3.1.2 control test execution

By means of an appropriate interface, the test operators could have the possibility of controlling test execution on the test configuration, either automatically or manually with support functions provided by the test tools - see section 3.3. This would involve, for example, the proper sequencing of interactions derived from the test cases.

3.1.3 communicate with remote operators

Since the test configuration and its operators are, in general, distributed across more than one location, there is a need for the operators to communicate with each other. This service could be provided by the test tools.

This communication function is central to the testing process and needs to be discussed in the context of test tools: interactions among test operators need to be recorded and coupled with actions performed at the test locations.

3.2 Interface to IUT and other Systems

It could be possible for the test tools to have direct, machine interfaces to the test configuration as shown in Figure 1. These interfaces could be for message submission and

retrieval, inspection of log information, etc. Three types of services could be provided by the test tools. See (b) in Figure 1.

3.2.1 interface for active control of IUT or other system

On this interface, the test tools could directly and actively control the test configuration according to the test cases, e.g. by submitting and retrieving messages.

3.2.2 interface for retrieving information from IUT or other system

As part of the test execution, the test tools could retrieve information such as logs which is stored in the test configuration via this interface.

3.2.3 cause changes in network

A machine interface to the network could be used for the testing of failure modes which can be controlled by software.

3.3 Test Execution

The carrying out of the tests could be supported in three different ways. See (c) in Figure 1.

3.3.1 generate test data from test cases

Test data such as messages and their contents and expected results could be generated from the test scripts and made available to test operators or used directly via the interface described in section 3.2.

3.3.2 execute logic of the test cases

The dependencies inherent in the test case logic could be made available to test operators or used directly via the interface described in section 3.2. A scheduling function could also be considered here.

3.3.3 record results

Logs of test case execution could be recorded in the test tools.

3.4 Database Maintenance

The database contains data on which the tests are based and data resulting from the tests. See (d) in Figure 1.

3.4.1 test case maintenance

Part of the database could be the test scripts themselves in various versions together with their update history. Provision could be made for the comfortable editing of test scripts.

3.4.2 results data maintenance

Part of the database could be the sets of results obtained in executing tests correlated with the test scripts and test configurations themselves.

3.4.3 report generation

Test reports as specified in [A260] could be generated automatically or partially from test results maintained according to the service described in section 3.4.2.

4. Strategies for Test Tool Procurement

On the basis of the discussions in previous chapters, possible strategies for the implementation of test tools are analysed in this chapter and recommendations are made.

4.1 General principles

4.1.1 Justifiable Development Effort

In order to estimate the effort which could sensibly be spent on the development and deployment of AMHS interoperability test tools, it is necessary to have a feel for the frequency and intensity of use which such tools may expect.

AMHS interoperability testing within the ACCESS Project was originally intended to be a one-time activity. With the actual test execution now postponed to take place outside the project, the estimates of effort still remain valid. However the possible repetition of tests becomes more likely.

Document [A261] contains[7] estimates of the effort necessary for developing test scripts and configuration files amounting to 22 man-days per IUT The existence of test tools is likely to have little impact on these activities[8] so that it is necessary to look at the effort involved in actually performing the tests.

It is estimated that, for each IUT, each of the 25 test cases would require approximately one half of one day for its execution plus an overhead of 5 days for setting up, report preparation etc. During these 17.5 days, the presence of 2 people can be assumed, one at each of two locations. This yields an order-of-magnitude estimate of 35 man-days for the testing of each IUT. This estimate does, of course, not take into account repetitions which become necessary when tests are not successfully completed.[9]

Assuming that 6 different IUTs would have to be tested during the introductory stages of the AMHS, an total effort in the vicinity of 200 man-days appears realistic.

For the purpose of discussion,[10] a basic reduction in effort of, say, 30% which is due to the use of appropriate test tools can be assumed. This analysis leads to the conclusion that, simply from a cost/benefit point of view, an effort of 70 man-days for the development and procurement of test tools could be justified.[11]

---

[7] Section 3.2

[8] It is assumed that some machine support would be used in these activities anyway.

[9] As a side-effect, the existence of test tools would encourage the execution of interoperability testing.

[10] This discussion does not take into account that test tools can, in addition, increase the quality of testing.

[11] Part of this effort would possibly have to be assigned to the purchase of hard- and software. However this is likely to be small in comparison with the manpower effort.

The figure of 30% for the reduction in effort through test tools is, as for the other figures used here, important for the conclusions drawn in the following sections. It is based on the authors' experience for a medium-scale testing activity.[12]

4.1.2 Basic Approach

The allowable effort derived in the previous section is not large, by any means, in the context of systems development. It follows that, simply from a cost point of view, the approach for organising the use of test tools needs to be pragmatic.

One necessary conclusion is that the development, procurement and provisioning of the tools should be in the hands of one of the test participants rather than being contracted out entirely to an organisation which is responsible for providing test support.[13] This is necessary in order to reduce and simplify the human interfaces between users and suppliers of the test tool support in connection with specification, modification, training etc. It is reasonable to assume that this designated test participant would also be willing to provide the support even when the organisation is not the owner of the IUT or of other systems involved.

The small amount of effort justifiable for implementation also requires that a simple approach to the technical development, procurement and deployment, using as many standard components as possible, is necessary. This is in view of the fact that the tools to be implemented would only be applicable to the concrete task at hand, i.e. to AMHS interoperability testing, and not to a wider context.

4.2 Recommended Test Tool Services

In the light of the discussion in the previous sections, recommendations are made in the following table on whether services identified in chapter 3 should be implemented or not. Note that implementation matters are not yet being considered here.

refer -ence	service description	implement? (yes/no)	reasons
3.1.1	interact with database	yes	essential service, simple to implement with standard software
3.1.2	control test execution	yes (restricted)	only instructions are given to test operator: no direct control
3.1.3	communicate with remote operators	yes	important because efficiency and quality can be improved significantly
3.2.1	interface for active control of IUT or other system	no	expensive to implement and dependent on type of IUT, i.e. a general implementation is not possible
3.2.2	interface for retrieving	no	expensive to implement and

[12] The more extensive and repetitive the testing activity, the higher this figure becomes.

[13] Of course, this does not restrict the possibilities of the test participant to contract work out.

refer -ence	service description	implement? (yes/no)	reasons
	information from IUT or other system		dependent on type of IUT, i.e. a general implementation is not possible
3.2.3	cause changes in network	no	necessary in only few test cases and therefore not justifiable
3.3.1	generate test data from test cases	yes	can be of great assistance to test operator
3.3.2	execute logic of the test cases	no	not possible if IUT and network interfaces not available
3.3.3	record results	yes (restricted)	done manually by test operator; not possible automatically if IUT and network interfaces are not available
3.4.1	test case maintenance	yes	important core service
3.4.2	results data maintenance	yes	important core service
3.4.3	report generation	yes	database function, easily implemented

One result which follows from the recommendations made in the above table is that fully automatic testing will not be supported by the test tools. This would have necessitated, in addition to the other services, all of the services in the table for which a "no" has been given in the "implement?" decision. Expressed differently, the test tool strategy requires that test operators remain in the "test loop" during testing.

4.3 Centralised versus Decentralised Tool Configuration

One important conclusion from the analysis in the previous section is that no machine interfaces between the test tools and the IUT or other systems (as shown by (b) in Figure 1) should be implemented The test operators always remain in the "test loop" and make use of the support provided by the test tool. This opens the possibility of implementing the test tools at one central location with remote access from the participating locations.[14] A comparison between a centralised and decentralised test tool configuration is shown in Figure 2.

[14] Of course machine interfaces to IUTs and other systems would also be possible from a central location but this is considered to introduce too much complexity and would not be justifiable.

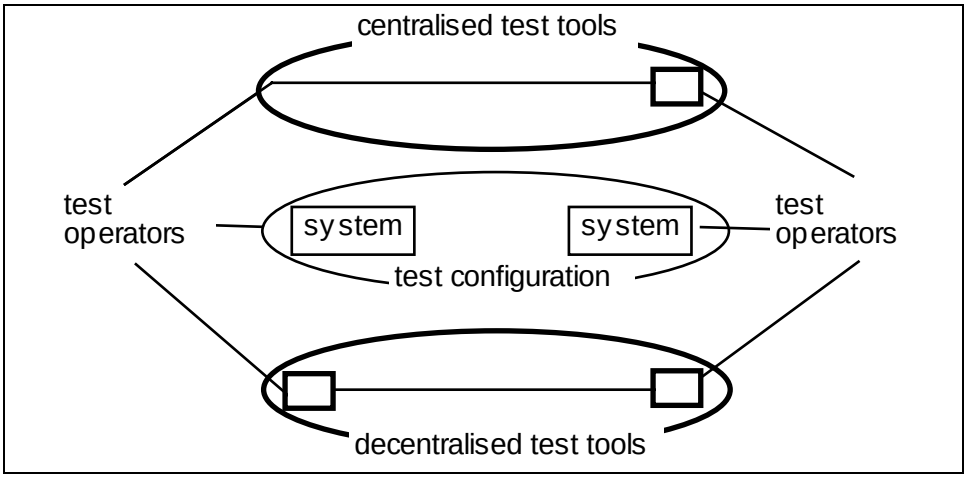

Figure 2: Comparison between centralised and decentralised configurations

In terms of simplicity, the centralised approach clearly has a number of advantages without causing increased communication costs. It is therefore **recommended** that the test tools be implemented centrally in one computer system allowing access from local and remote test operators.

For the networking between test operators and the centralised tool system a number of options are possible:

- ATSO packet switched WANs,
- the same communication infrastructure as is used by the systems in the test configuration,
- the Internet, e.g. by means of the Telnet protocol (security problems are not considered to be important here).

Availability and ease of use should be the criteria used to make a choice among these options.

Assuming the adoption of the centralised approach to the implementation of test tools, the portability of the tools would not bring any significant advantages. However because of the small scale of the implementation (see the following sections), the tools can be considered to be portable in any case.

4.4 Functional Requirements

This section lists some high-level functional requirements placed on the test tool. According to the recommendations derived in preceding sections the tool. consists of a central server which is accessed from terminals over a data network. Only requirements placed on the server (and not the terminals) are discussed here and the following sections. They are derived from chapter 3 and the results of section 4.2.

The core of the tool functionality is a database application. This is responsible for administering the three sets of data listed in section 3.4, together with data on the test configuration. For the implementation, industry standard database systems should be considered. In addition, transient sets of data such as messages to be sent, instructions to the test operator will exist during the preparation and execution of tests.

Support software shall exist for the creation and maintenance of test scripts in different versions, including plausibility checks.

On request of the operator, the steps to be carried out by him and expected events shall be derived by the test tool from test scripts and configuration data. The operator shall be able to enter the current state of a test and receive an updated step sequence.

for each test executed, acknowledged test steps, with optional comments from the operator shall become part of a test log (results database).

Reports on test execution shall be generated on the request of the operator in a format which can be flexibly adapted to future needs.

The possibility of communication with remotely located test operators shall be integrated into other functions, e.g. transferring the current test status. In addition there shall be the possibility of communication free text messages via the test tool. There is only a need for communication within Europe to be foreseen.

This functionality shall be made available at a test operator terminal interface which is connected to the server by a data network. Administrative functions shall control the access of operators to the tool by means of passwords.

4.5 Non-functional Requirements

The performance requirements placed on the test tool are minimal.

The number of terminals/workstations accessing the server application at any one time shall be restricted to one per test location and the total number of transactions per second is likely to be far less than one per second on average. The data volumes involved are likely to amount to only a few megabytes.

As a result of this, no dedicated hardware is necessary and the creation of a new (set of) application(s) on existing or shared hardware will suffice. Availability requirements are relatively high so that tests are not interrupted by the failure of the tool. However the availability normally provided by non backed-up servers will suffice.

A functional and quantitative extensibility of the system shall be implemented.

Note: A consideration of these requirements and those of the previous section leads to the conclusion that the test tools can probably be implemented on commercially available PCs with sufficient memory, processor speed, communication ports, standard software etc. It appears unlikely that the resources of bigger systems such as workstations, for example, would be needed.