U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 4676

National PDES Testbed
Report Series

Validation Testing System Requirements

Katherine C. Morris
Michael McLay
Peter J. Carr

NATIONAL PDES TESTBED

NIST

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 4676

National PDES Testbed

NATIONAL PDES TESTBED

Validation Testing System Requirements

Katherine C. Morris
Michael McLay
Peter J. Carr

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher,
Secretary of Commerce

National Institute of
Standards and Technology
John W. Lyons, Director

September 1991

DEPARTMENT OF COMMERCE
UNITED STATES OF AMERICA

Table of Contents

List of Tables

Validation Testing System Requirements

Abstract

The need to share complex product data across multiple enterprises, using different computing systems and networks is growing. The ISO Standard for the Exchange of Product Model Data (STEP) addresses this need by providing information models which clearly and unambiguously describe this data for different enterprise applications. The validity of these information models is essential for success in sharing data in a highly automated business environment.

This document describes requirements for software to support the testing of information models for validity and correctness. The requirements provide a basis for the software development to support the Validation Testing System (VTS) within the National PDES Testbed. This document describes the software needs as currently known for a complete validation testing system. It identifies the scope of the VTS software, presents functional requirements for the proposed system, describes the data flow, environment, external interfaces, performance, and documentation requirements of the system, and prioritizes the requirements for the purpose of implementing the proposed system.

1 Introduction

This document describes requirements for software to support the automation of the Validation Testing System (VTS). The VTS is under development in the National PDES Testbed[1] at the National Institute of Standards and Technology (NIST) [McLean90] [Mitch90]. The Testbed is used by researchers to test the validity of application protocols [Palmer91], or application models[2], which are being proposed for STEP[3]. Conformance testing of STEP-based application systems and prototype application systems are not addressed in this document.

The requirements address automation for a complete VTS and will be used as the basis for software development within the Testbed. However, the requirements could also be used by others planning a validation testing system. The intended audience for this document is program sponsors, project managers, and software developers. This document represents comprehensive requirements for validation testing. In the Testbed, software to support these requirements will be acquired or developed in stages. Appendix A describes the priorities for software implementation.

Requirements for the VTS software are based on experience in the testing of application models. The requirements were gathered by interviews with users and developers of the current testing software in the National PDES Testbed. The requirements reflected in this document are based on experience with existing systems. Based on this experience, four toolkits, described in this paper, have been identified. These toolkits reflect enhancements to the existing testing process. Other technical requirements are derived from the evolving STEP development methodology.

To consider these requirements, compare the validation testing system to the role of a building code inspector. The VTS examines STEP models in an application context, just as a building inspector examines the components of a new building. Constructing an application model is similar to a building inspector's analysis of a new building to determine what should be tested. The VTS defines tests to perform against the new application model, just as the building inspector must formulate a plan for testing building features. The building inspector must test features of the building for flaws. Likewise, a mechanism for testing features of the application model is identified. Features are tested against known cases, just as a building inspector uses an instrument to check for correct voltages in electrical outlets.

1. Funding for the Testbed has been provided by the Department of Defense Computer-Aided Acquisition and Logistic Support (CALS) Office.

2. The term *application model* will be used throughout this paper to refer to the domain specific conceptual schema which are under evaluation in the VTS. This schema has several forms in the development process: an application protocol (AP), an application resource model (ARM), or an application interpreted model (AIM) which applies application specific constraints to STEP models.

3. The Standard for the Exchange of Product Model Data (STEP) is a project of the International Organization for Standardization (ISO) Technical Committee on Industrial Automation Systems (TC 184) Subcommittee on Industrial Data and Global Manufacturing Programming Languages (SC4).

The process of testing an application model, as described above, can be decomposed into six high-level activities: Scoping the Application Context, Model Construction, Test Definition, Test Case Data Generation, Test Execution and Analysis, and Model Refinement and Improvement. Each of these activities may be performed by a separate group of people. Four integrated software toolkits are required to support these activities. A toolkit consists of the set of software, or tools, which are needed to support the particular activities for a phase of the testing process. Table 1 identifies these activities and the corresponding software toolkits. More information on the activities and toolkits is provided in the following section.

TABLE 1 Activities and Toolkits of the Validation Testing System

Activity	Toolkit
Scoping the Application Context	Model Scoping and Construction
Model Construction	Model Scoping and Construction
Test Definition	Test Definition
Test Case Data Generation	Test Case Data Generation
Test Execution and Analysis	Test Execution and Analysis
Model Refinement and Improvement	Model Scoping and Construction

Each toolkit will consist of the set of software tools necessary to address the tasks of the particular activities associated with a phase of the process. Some tools will be used in more than one toolkit. This reflects the overlap in the requirements which these toolkits will support. Initial versions of the toolkits will address a subset of the requirements for the toolkit and will consist of a partial set of the tools identified. These toolkits will be enhanced as needed. Where available, commercial or other third party tools will be used.

The *Test Case Data Generation*, the *Test Execution and Analysis,* and the *Test Definition* toolkits will each address a single activity. However, the *Model Scoping and Construction* toolkit will service more than one activity because of the overlapping requirements for automation to assist in the processes of model scoping, development, and improvement. This is partly because these activities involve intensive analysis which involves considerable human judgement. Therefore, they allow for a minimal amount of automation in the form of general purpose tools.

The following section describes the functional requirements of each toolkit. The remainder of the requirements specified in this paper apply to all the toolkits. Appendix A describes priorities for implementation of the toolkits within the Testbed.

Detailed implementation considerations for the VTS software are currently under development and are outside of the scope of this document. These considerations will be available in the VTS software design document. Interfaces to external systems rely on standards where available. Further detailed requirements for selecting external systems will be identified as needed.

2 Functional Requirements of the VTS Toolkits

This section describes the four toolkits which make up the VTS. The activities supported by each toolkit are summarized, and the functional requirements for these activities are presented. Based on these requirements, areas of overlap between the toolkits are identified. In the lists of requirements the words in parentheses indicate names given to the requirement area described.

The VTS software will provide a controlled environment which reduces the potential for misleading errors in the model validation process and reduces the need for trained technical personnel. A user interface will be provided for each toolkit to eliminate manual intervention that can introduce errors into the testing environment. All tools which make up the validation system should have the same type of user interface to facilitate the interchange of validation testing personnel across major functional boundaries.

References to tools from the toolkit will be indicated by enclosing the name in parenthesis. For example, the definition of a tool for parsing Express [ISO11] (Express Parser) is found in Table 3.

2.1 Model Scoping and Construction

This section addresses the *Model Scoping and Construction* toolkit. The *Model Scoping and Construction* toolkit supports three activities: *Scoping the Application Context*, *Model Construction*, and *Model Refinement and Improvement*.

The *Scoping the Application Context* activity identifies a formal technical boundary for the application model. This boundary governs what portions of the STEP model are to be tested and for what use. This activity results in 1) an application activity model, 2) a statement of the scope and requirements, 3) Express specifications and definitions, and 4) graphical representations of the model.

During the *Model Construction* activity experts are interviewed to obtain detailed application information requirements and to understand any constraints on the use of this information. This activity results in the detailed application model to be tested. This model must be provided in three formats: 1) a graphical representation, such as in Express-G, 2) a formal specification in Express, and 3) a completely documented prose description including definitions and diagrams. Alternative formats to the Express-G based representations described above are IDEF1X [ICA85] and NIAM [NIJ89] among others.

The *Model Refinement and Improvement* activity resolves issues with the application model which were uncovered during the *Test Case Data Generation* or *Test Execution and Analysis* activities. Alternative solutions are developed and one is selected. Once these issues have been resolved, the model is modified and a new model is released for testing purposes. This activity results in 1) the refined model, and 2) an issue resolution statement describing the solution selected and supporting rationale.

These three activities involve intensive human judgement and, therefore, allow for a minimal amount of automation. The following functionality is required for the *Model Scoping and Construction* toolkit to support these activities:

a) Provide a capability for ensuring the syntactic correctness of the application model. (Express Parser)

b) Provide a capability for examining the application model and STEP model. (Express browser) The ability to display the application model, in both the Express-G and the Express language formats, is needed.

c) Provide a capability for diagramming the application activity model. (Diagramming Tools)

d) Provide a capability for browsing the specification of the application model in a variety of formats. (Express Browser and Other Model Browsers)

e) Provide a capability for indexing the information stored on the file system. (Operating System, future capability by Configuration System)

2.2 Test Definition

The *Test Definition* toolkit supports the *Test Definition* activity. This activity defines requirements and tests which will be used to evaluate the functionality of the application model. This stage consolidates expert interview results into realistic test scenarios. During this activity the application model, represented in a variety of formats, will be referenced. Alternative formats to the Express based representations described above are IDEF and NIAM among others. The outputs of this activity are a test plan, test purposes and test scenarios, usage constraints, expected results, and test product/part profiles.

Support for the *Test Definition* activity will require the following functionality:

a) Provide a capability for illustrating the concepts in an application model. (Diagramming Tools)

b) Provide a capability for preparing and reviewing the test plans, test case documentation, and development of model issues. (Word Processing System) Tools which tailor the word processing system for working with specific types of documentation are desired. Specifically, the documents need to be produced in ISO format. The word processing system should also include or have an interface to a diagramming facility for illustrating the concepts found in the application model.

c) Provide a capability for browsing the specification of the application model in a variety of formats. (Express Browser and Other Model Browsers)

d) Provide a capability for creating, and storing related documentation. (Word Processing System)

2.3 Test Case Data Generation

The *Test Case Data Generation* toolkit supports the *Test Case Data Generation* activity. The *Test Case Data Generation* activity provides product data for the product/part profile for a given test scenario. The primary output of this activity is the data and the abstract test cases to be used to test the application model. During this activity serious flaws in the application models are sometimes discovered and documented as issues. This activity tests the application model's ability to support realistic data. Recommended VTS software enhancements are also documented.

The primary automation of this activity is for the task of providing test data. The most reliable and efficient way to obtain test data is in the form of an Initial Graphics Exchange Specification [IGES] file extracted from a CAD system. This can provide up to 25% of the data needed but covers only a handful of STEP geometric representation entities. In the initial testing experience half of the test product data definitions were built from scratch. This is the most labor intensive and error prone, but potentially most reusable, activity in the entire testing process.

Several sources are available for data. However, the data gathered from these sources is generally not provided in a complete STEP format. The current approach to gathering data is the following:

1. A CAD system is used to produce a geometric model for application models that require geometric data.
2. This data is extracted into a STEP file format.
3. Additional data is entered manually to produce a complete set of data in the STEP exchange file format [ISO21].

Alternatively, the complete set of data is manually generated. Besides being a very costly process, this method is not as reliable as generating data using a commercial CAD system. The commercial systems support necessary constraints on the data with respect to a particular geometric modeling representation. A STEP editor should support the enforcement of constraints found in the application models. Ideally, other commercial computer aided-x systems might be used to generate additional test case data.

There are two aspects of the data which need to be addressed in order to bring the data into a format suitable for use in the testing process. The first aspect addresses the completeness of the data set with respect to the application model. The second aspect addresses the physical format of the data.

The following capabilities are required to support the *Test Case Data Generation* activity.

a) Provide a capability for examining the specification for the application model. (Express Browser)
b) Provide a capability for documenting issues and recommendations. (Word Processing System)
c) Provide a capability for translating an application model into a program work space. (Express Translator)
d) Provide a capability for loading the data into the system. (STEP Exchange File Parser)
e) Provide a capability for merging multiple files in the STEP exchange file format into one file and maintain the appropriate references. (STEP Data Editor)
f) Provide a capability for managing versions of product data to organize test data. (Configuration System)

g) Provide a capability for producing product data. (CAx[4] Systems)

h) Provide a capability for examining, supplementing, and creating data. (STEP Data Editor)

i) Provide a capability for translating data from the IGES exchange file format to the STEP exchange file format. (IGES Translator)

j) Provide a capability for translating data from other CAx formats to the STEP exchange file format. (Other Translators)

2.4 Test Execution and Analysis

The *Test Execution and Analysis* toolkit assists in executing the test cases and analyzing the results. In order to execute the test cases a testing environment needs to be established and the test cases need to be formally specified. Analysis of these cases involves comparing the tests to the model to determine the validity of the tests. The results of *Test Execution and Analysis* activity are 1) a report which describes problems with and needed improvements to the application model, 2) executable test cases for reproducing test results, 3) improved test product/part data, and 4) test reports.

This activity allows for a great degree of automation. The automation necessary to support this activity can be broken down into the following tasks:

1. The application models are represented in the testing system.
2. Data, gathered during the previous activity, is loaded into the testing system.
3. The tests cases are executed against the data to ensure that test purposes as identified in the test plan can be achieved. The results of the execution are logged.
4. The test case executions are analyzed and an evaluation is made as to whether the product data could be correctly used. The results are measured against the expected results described in the test plan.
5. Deficiencies in the application model are written up in an issues document. These issues are resolved within the appropriate ISO/TC184/SC4 project.

The *Test Execution and Analysis* activity uses a database management system. The typical testing scenario is as follows: the application model is represented in the database management system; data is loaded into the system; the tests are specified in the system's query language; these queries are executed; and the results are analyzed. The queries are the executable form of the abstract test cases.

The data collected in the *Test Case Data Generation* activity needs to be brought into the *Test Execution and Analysis* toolkit. The primary means of transferring this data is via exchange files in the STEP exchange file formats. However, the *Test Execution and*

4. CAx is any Computer-Aided operations/processes, including MCAD (Mechanical Computer-Aided Design) e.g. drawing/drafting, ECAD (Electrical Computer-Aided Design), e.g. PCB layout, MCAE (Mechanical Computer-Aided Engineering), e.g. solids modeling, ECAE (Electrical Computer-Aided Engineering), e.g. logic design, CAM and CIM (Computer-Aided Manufacturing and Computer-Integrated Manufacturing), e.g. NC processing and photoplotting.

Analysis toolkit may be designed so that data can be directly transferred into the database by the *Test Case Data Generation* toolkit, thereby by-passing the exchange file.

Once the data has been loaded into the database system, tests, in the form of queries, are performed to verify the correctness of the model for test purposes. These tests range from simple traversal of the data to complex transformations. Although the test queries are prepared in advance, they need to be performed interactively so that the data can be examined throughout the process. This interaction is particularly important in determining the source of a problem when the manipulations do not succeed. A record of this session should also be logged to assist in analysis.

After the execution of the test cases is completed the results are analyzed and documented. The final report and issues document are produced.

The following functionality is required by the *Test Execution and Analysis* activity:

a) Provide a capability for translating application models into an existing database system. (Express Parser)

b) Provide a stable database system with well documented interfaces. (Database System)

c) Provide a capability for loading data into the database system. (STEP Exchange File Parser)

d) Provide a capability for merging multiple files in STEP exchange file format into one file and maintain the appropriate references. (STEP Data Editor)

e) Provide an interactive query and update language for the database system. (Query Language)

f) Provide a capability for browsing data. (STEP Data Browser)

g) Provide a capability for recording the interactive session during which the tests on the data were conducted. (Logging Mechanism)

h) Manage the relationship of testing requirements to specific tests in terms of the entities and attributes in the application model which are being tested. (Cross Referencing System)

i) Provide a capability to document issues and recommendations. (Word Processing System)

2.5 Other needs

While many different models will be tested by the VTS, the testing process is fairly constant and much of it can be automated. The inputs to the validation testing process (the application models and data) need to be controlled, and much of the work flow needs to be automated. The components that change between testing processes are the application models and, therefore, the data. These changes need to be managed.

Another aspect of the transition between testing of application models involves the conversion of data to conform to the new model. Model testing is an iterative process, and data needs to be synchronized with the latest version of the application model.

The following additional functionality is needed to support the VTS:

a) Organize the application models, their respective test cases including the data and results, and the location and versions of tools with which they operate. (Configuration System)

b) Automate the flow of activity through the various tasks of the testing process. (programmable interface to the Configuration System)

c) Transform data to conform to a new application model when the model is modified. (Data Converter)

2.6 Summary of software requirements

The following table shows the composition of each VTS toolkit. The tools identified represent the software required to support the toolkit. These tools are based on the requirements described above. Configuration support for version and workflow management spans all the toolkits.

TABLE 2 Composition of VTS Software Toolkits

Model Scoping and Construction	**Test Definition**
Diagramming Tools	Diagramming Tools
Express Browser	Express Browser
Word Processing System	Word Processing System
Other Model Browsers	Other Model Browsers
Express Parser	

Test Case Data Generation	**Test Execution and Analysis**
Express Parser	Express Parser
Express Browser	Express Browser
Word Processing System	Word Processing System
Express Translator	Express Translator
STEP Data Editor	STEP Data Browser
STEP Exchange File Parser	STEP Exchange File Parser
IGES Translator	Database System
Other Translators	Query Language
Data Converter	Logging Mechanism
CAx Systems	Cross Referencing System

Table 2 above illustrates that there are several requirements which are shared by the four toolkits. Table 3 on the following page summarizes all the tools that comprise the VTS software and provides a brief description of each tool. The software requirements are listed in the order that they appear in the text.

TABLE 3 VTS Tools

Tool	Description
Configuration System	Supports the management of documents and other files, including programs, used by the VTS system.
Diagramming Tool	Supports display and creation of diagrams which illustrate parts of an application model.
Express Browser	Displays the contents of a model written in Express in a “user friendly” format. Provides minimal hypertext like capabilities. The initial version will have an interactive interface and the future version will also have a programmable interface.
Word Processing System	Supports editing functions and provides context sensitivity for standard formats.
Other Model Browsers	Display the application model in a variety of different formats for reference. Do not need to provide capability to modify model. Could be simple drawing packages.
Express Parser	Parses an application model specified in Express to verify syntactic its correctness.
Express Translator	Translates an application model written in Express into a program workspace.
STEP Data Editor	Provides an interactive environment for the display, manipulation, and editing of data which corresponds to an application model.
STEP Data Browser	Provides an interactive environment for the display and manipulation of data which corresponds to the application model. Does not allow the user to change the data.
STEP Exchange File Parser	Parses a STEP file into a working format and/or database system.
IGES Translator	Reads in an IGES file and outputs a corresponding STEP file.
Other Translators	Translate data from a CAx system’s internal format to STEP format.
Data Converter	Translates a data file corresponding to a particular application schema to an updated format based on changes made to the schema.
CAx Systems	Provides a capability for producing product data.
Database System	Provides for persistent storage of and shared access to data.
Query Language	Provides capability to represent data manipulations to be conducted during testing. Dependent on the database system.
Logging Mechanism	Records the session during which the tests on the data were conducted.
Cross Referencing System	Supports management of the relationships between specific tests and relevant sections of the application model.

3 Data Flow Requirements for the VTS Toolkit

Each of the activities of the VTS consumes and produces data. The data produced by one activity is then consumed in the subsequent activity. The degree to which this data is processed by computer varies with the activity. Therefore, the data flow within and between each of the toolkits varies. This section describes the data flow between toolkits. Data flow within toolkits is outside of the scope of this document. Table 4 illustrates inputs and outputs to the four toolkits.

TABLE 4 Data Flow Between Toolkits

	INPUT	OUTPUT
Model Scoping and Construction	Application Requirements STEP Models	Scope & Requirements Statement Application Activity Model Express Specifications and Definitions Graphical Representations (e.g. Express-G)
Test Definition	Application Requirements Scope & Requirements Statement Application Activity Model	Test Plan & Test Purposes Test Product/Part Profile Acceptance Criteria
Test Case Data Generation	Test Plan & Test Purposes Express Specifications Test Product/Part Profile Acceptance Criteria Product Data Artifacts (i.e. documents and IGES files)	Application Test Environment STEP File Abstract Test Cases
Test Execution and Analysis	Test Plan Abstract Test Cases Application Test Environment STEP File: Test Product Data	Model Issues & Testing Software Enhancements Evaluated Executable Test Cases (for retesting) Test Reports

The primary output of the *Model Scoping and Construction* toolkit is the application model in human and computer interpretable formats. These views of the application model are used in the other toolkits. The model is the basis for many of the tools contained in the other toolkits. These toolkits are specifically configured to support the application model produced by the model scoping and construction activities. The data consumed by this toolkit are the existing STEP conceptual models. These models provide a basis for the application model being produced. The input conceptual models are a reference source and are not modified.

The output of the *Test Definition* toolkit is a plan for evaluating the application model. This plan results in test objectives and scenarios, abstract test cases, usage constraints, expected results, and test product/ part profiles. These materials define what parts of the application model are to be tested and provide some guidance to those people formulating and evaluating the automated tests. The outputs of this activity are not machine interpretable, but they are the human interpretable plan for testing.

The initialization of the *Test Case Data Generation* toolkit involves configuring the tools contained in the toolkit to support the application model produced by the *Model Scoping and Construction* toolkit. Once the toolkit has been initialized it consumes product data from external sources and produces data which corresponds to the application model. The data consumed are represented in many formats, while the data produced are available in STEP exchange file format. The data produced may also reside in a database system.

Likewise the *Test Execution and Analysis* toolkit is initialized with the data produced using the previous toolkits. The application model produced during the *model scoping and construction* activities is used to configure most of the tools in this toolkit, and the data produced by the *Test Case Data Generation* activity is incorporated into the system. As the level of sophistication of the toolkits increases, the *Test Execution and Analysis* toolkit will be based on the *Test Case Data Generation* toolkit directly and there will not be a need to initialize it explicitly. The new toolkit will appear as enhancements to the initial toolkit.

The *Test Execution and Analysis* toolkit is used to generate computer interpretable test cases based on the test plans which were developed during the *test case definition* activity. These test cases are then executed using the data which resides in the toolkit. The results are used in the analysis and refinement activities. The data which are produced by this toolkit are the computer interpretable test cases and the results of executing these tests.

In summary the *Model Scoping and Construction* toolkit is used to generate the application model which is the basis for the *Test Case Data Generation* and the *Test Execution and Analysis* toolkits. The *Test Definition* toolkit is used to formulate the plans for testing the application model; this plan guides the users of the *Test Execution and Analysis* toolkit in formulating automated tests. The *Test Case Data Generation* toolkit is used to generate product data. The *Test Execution and Analysis* toolkit is used to generate computable test cases and execute these test cases against the product data. The results of these test cases are then used in model analysis and refinement. The model refinement activity leads to a new application model and may cause refinements to the STEP models. The entire process is repeated using the refined model.

4 VTS Software Environment

The VTS software will eventually operate at multiple sites and on many types of computers. Currently model validation is performed predominantly by PDES, Inc.[5] They use test facilities at the National PDES Testbed at NIST and at the South Carolina Research Authority (SCRA). In the future the VTS software should support the broader community of all STEP Application Protocol developers. To ensure software portability, the VTS will utilize computer standards where possible. In particular, the POSIX [FIPS 151], X Window System [Scheifler89], C [ANSI89], C++ [Stroustrup90] and SQL [ANSI86] standards will be used.

An open systems based computing environment will be available to all users of the Validation Testing System. This will consist of a collection of computer systems running standards compliant operating systems, communications protocols, windowing systems, and data access protocols. Vendor supplied hardware supporting POSIX (Portable Operating System Interface for Computer Environments) compliant operating system, OSI (Open System Interconnect) communications software, and applications using X Windows for the user interface is the basic model for the VTS open systems environment. The initial implementation will use TCP/IP (Transmission Control Protocol / Internet Protocol) for communications and UNIX for the operating system to expedite implementation of the software tools. Features from TCP/IP and UNIX that are not readily portable to the open architecture will not be used. As OSI and POSIX compliant tools become available in the Testbed, software will be modified to work with these standards.

4.1 General Environment

A primary requirement of the VTS software is that it work within the Validation Testing Laboratory of the National PDES Testbed. Described below are the hardware, software, and networking requirements which exist within this environment.

The general testing environment consists of POSIX/X Windows workstations such as the SUN SPARC, DEC DECstation, HP 700, and IBM RS6000. Ideally, all software tools would be made available on all platforms within the Testbed; however, licensing restrictions and limited resources available for porting activities will restrict availability of VTS software to executing on specific machine architecture or on specific workstations (CPU). The minimum requirement is that all software be available on at least one CPU within the Testbed. Remote X Window display access to all Testbed CPU's will allow VTS tools to be used from Testbed workstations other than the workstation on which it is being executed.

The following are requirements of the general environment:

a) The software in the VTS must execute on workstation technology available in the Testbed. The machines are typically 10 MIPS (Million Instructions Per Second) processors with 16 megabytes of core memory. All machines have 1024x900 or larger black and white screens. Several color displays are also available. The Testbed is

5. PDES, Inc. is an industrial consortium which has the charter to accelerate the development of STEP. Members of PDES, Inc. are the primary users of the National PDES Testbed and the VTS.

currently equipped with DECstations 5000, SUN 3, SUN SPARC, IBM RS-6000, and HP 700 series computers. Each VTS tool is required to be executable on one of these platforms.

b) VTS software must be capable of being executed from remote sites through network and modem connections.

c) Access to embedded databases by VTS software will be done through a defined interface. The interface specified in the STEP Data Access Interface Specification (SDAIS) [SIIP91] will be used when it becomes available.

d) Configuration management services are required to manage the versions of software application models and product data.

e) The design of the user interface for the Testbed software will be defined by the needs of the users in terms of computer sophistication, mode of computer access, and familiarity with other software systems.

f) Network access to the VTS must support remote terminal capability, remote shell execution, remote procedure call, remote data access, and remote file copy. The initial network supports connection to a host computer through the TCP/IP protocol. Future network connections will adopt OSI services as they become commercially available.

g) Remote connections via phone lines must support the V.32 standard (9600 baud with data compression and error correction).

h) Multiple dial-in lines will be supported through a single phone number by use of a rotary switch.

4.2 Development Environment

Some of the software tools used in the VTS will be developed in the National PDES Testbed. To support this activity, a generic development environment for creating VTS software tools will be established. The following are requirements for the development environment.

a) The VTS software will be developed on one of the workstations as specified for the general environment (Section 4.1).

b) Software in the development environment will use the operating systems, communications, and user interfaces specified in the general environment (Section 4.1).

c) An environment will be established to facilitate team development of the VTS software. This environment will provide for shared access, configuration and version management, and archival of any source code developed and other programs incorporated into the VTS.

d) Developers of VTS software will use the Testbed generic software development environment. This environment will be configured to supply the developer with direct access to all commands necessary to build VTS software tools.

e) VTS software will be written using the POSIX utilities, ANSI C, and C++. Compilers, symbolic debuggers, and parsers to support these languages are required.

f) The graphics user interface for the VTS will use the X11 R4 protocol. InterViews [Linton91] will be used for building X Window based applications.

g) Initially TCP/IP based remote procedure calls will be used to link client-server based VTS applications and the Internet will be used to support connections from remote sites.

4.3 Operating Environment

An operational environment for executing VTS software will be established to support end users of the VTS. The following are requirements for the operating environment.

a) The VTS software will be available for use on workstations specified in the general environment (Section 4.1). Those tools marked as *initial release* in Table 5 will be provided on the indicated platforms; future releases may include other platforms. The software will be released for a particular platform as it becomes available.

b) Workstation sizing requirements will be based on usage requirements determined by survey of the Testbed users. Additional equipment will be acquired as necessary.

c) Software in the operational environment will use the operating systems, communications, and user interfaces specified in the general Testbed resources (Section 4.1).

d) An X-Window based environment will be established for the users of the VTS software. This environment will be configured to provide easy access to the tools needed in the testing process.

e) Users of the Testbed software must have access to the non-graphics based tools through network based connection or through dial-in access.

f) Networking tools available to operational environment users will include services specified in the general Testbed resources (Section 4.1).

TABLE 5 Minimum VTS Tool Availability by Architecture

Tool	Architecture	Release
Configuration System	SUN	future
Diagramming Tool	SUN or HP	future
Express Browser	SUN	future
Word Processing System	WordPerfect - PC FrameMaker - SUN[6] Emacs - SUN, DECstation	initial initial initial
Other Model Browsers	TBD	future
Express Parser	SUN, DECstation	initial
Express Translator	SUN	initial
STEP Data Editor	SUN	initial
STEP Data Browser	SUN	future
STEP Exchange File Parser	SUN	initial
IGES Translator	DECstation[7]	initial
Other Translators	TBD	future
Data Converter	SUN	future
CAx Systems	TBD	future
Database System	SUN or DECstation	future
Query Language	SUN or DECstation	future
Logging Mechanism	SUN	future
Cross Referencing System	SUN	future

4.4 Maintenance Environment

Support for recompilation of software on multiple workstation architectures within the Testbed requires a facility to support direct sharing of software source code between all machines in the Testbed.

a) The maintenance environment must support the ability to generate and store executable and object code libraries for each workstation architecture simultaneously.

b) The POSIX make facility will be used to build all VTS tools on the Testbed workstations. Makefiles must work independent of the CPU architecture. Setting the CPU POSIX environment variable will direct all machine dependent build operations.

c) The Configuration management system used within the Testbed will be used to control all releases of the VTS software. The current version of all released software will be kept in read-only files in a publicly accessible directory.

6. Usage of WordPerfect and FrameMaker is limited to machines for which they are licensed.

7. The IGES to STEP translator is supplied by PDES Inc. Current plans call for initial support for the DECstation.

5 External VTS Software Interfaces

The VTS will use tools developed outside the scope of the VTS software development. The toolkits must interface with the external systems described in this section. These external systems are divided into three categories: environmental interfaces, software systems, and STEP requirements.

5.1 Environmental Interfaces

VTS software will conform to the following external system interfaces:

a) Development Language

C++ has been selected as the development language for the VTS software. The C++ compiler used for development must be compatible with the VTS database management system. ANSI C may also be used for some components of the system.

b) Operating System

The VTS operating system specification is POSIX. All VTS software will be written to use the program interface and software tools from POSIX. Non-POSIX dependencies will be identified and documented by the developers. The VTS software will also interface to NFS (Network File System), TCP/IP, and OSI.

c) User Interface Systems

The graphics based VTS software must provide a user interface based on the X1 windowing environment. All character based VTS software must be able to operate from an ANSI terminal interface.

5.2 External Software Systems

The VTS software will require access to the following external software tools:

a) Express Parsers and Translators (*Fed-x*[8])

Tools to process Express definitions of the application models are required for the VTS software development. These tools will be used in verifying the Express syntax and for generating a representation of the models in a program workspace. The *Fed-x Toolkit* [Clark90] is available for parsing Express and can be used to generate a representation of the application model for the toolkits.

b) Word Processing System

The VTS requires word processing software capable of supporting ISO conventions for formatting a standard. This includes formatting text and paragraphs, mixing text and graphics on a page, generating indexes, and generating tables of contents.

c) Graphic Display Systems

The VTS requires the capability to generate 2-D and 3-D displays of product information. This is planned as future work.

8. The *Fed-x Toolkit* software has been developed at NIST but not as part of the VTS project.

d) Database Management System

A commercial database management system will be chosen to be incorporated into the VTS software. The system must be capable of using C++ or Express as its schema definition language. A more detailed set of requirements is available in [Morris90a].

e) Configuration Management System

A tool is needed to maintain information relating to the interaction of the different versions of the components of the VTS system. Some components are the executable programs or the software tools, the application models in their various forms, and associated data files in their various formats. The configuration management tool should be accessible to an executing program. Until this software is in place, its function will be performed manually.

5.3 External STEP Interfaces

The VTS software is required to maintain consistency with changes to the following specifications of STEP:

a) Express Language

The VTS software requires automated tools for processing the application models which are written in Express [ISO11]. Periodically a new version of the Express language is made available. The VTS software must be updated to support new versions of Express as it is released.

b) STEP Exchange File Format

VTS software tools are required to maintain consistency with the STEP exchange file format [ISO21].

c) STEP Data Access Interface Specification

This specification is not yet available. When the specification is available, the VTS software will be upgraded to use the STEP Data Access Interface [ISO22].

6 Performance Requirements for the VTS Software

The VTS tools are to be used interactively both within the Testbed and remotely. Performance considerations need to address data access locally, across networks (both local area networks and wide area networks), and over telephone lines. Tool performance should provide reasonable response time for common operations. Further analysis of the application models and data to be tested is needed to provide detailed performance guidelines; however, the following general criteria must be satisfied:

a) Provide reasonable response time to support interactive use.

b) Provide a user friendly interface with multiple methods for triggering commands (menus and command line interfaces) and useful interactive assistance (on-line help).

c) Provide a smooth mechanism for re-configuring toolkits for testing different application models.

d) Provide informative error reporting and graceful recovery for predictable problems with data.

7 Supporting Documentation for the VTS Software

The complete Validation Testing System will include documentation to support the understanding and use of the software within the Testbed and at remote testing sites. The following documents are required to support this activity:

a) Validation Testing System Software User Guide
This document will provide a reader who is familiar with C++ and the STEP specification with background information and an introduction to the VTS software. It will describe how to use it for the development of STEP based applications. Examples of using the software will be provided.

b) Validation Testing System Software System Manuals
This document will provide an application developer with detailed definitions of the classes, functions, constants, structures, macros, and miscellaneous definitions found in the VTS software.

c) Manual Pages for On-line Help with VTS Commands
Each command in the VTS will have a *man* page describing the function of the command and the command line syntax for running the command.

d) UNIX System Manual Pages for On-line Help with VTS Classes and Functions.
Classes and functions will have *man* pages that describe the class or function. Groups of related functions or classes may be combined into a single *man* page.

8 Summary

The Validation Testing System supports the National PDES Testbed. Requirements for the system are driven by the STEP development effort and reflect the needs of Testbed users. The development methodology which the VTS software supports is described in A Proposed Testing Methodology for STEP Validation [Mitch91]. Construction of the tools for the VTS will be dependent on available resources. (See Appendix A for priorities.)

Based on the major activities involved in the development of application models for STEP, four software toolkits have been identified for the VTS: *Model Scoping and Construction*, *Test Definition*, *Test Case Data Generation*, and *Test Case Execution and Analysis*. Data flow between these toolkits corresponds to the flow of activities in the testing methodology. Each toolkit consists of multiple software tools. Many of the tools are shared among the toolkits because the validation process focuses on the validation of a single application model. The common element to all the activities and toolkits is the application model undergoing test. However, the model is represented in a variety of ways throughout the testing process.

The National PDES Testbed provides a development and operating environment which supports the development of the VTS software and the use of the toolkits. The users of the Testbed reside around the country and travel to NIST to use the facility. Less often, the facility is accessed remotely. It is important to provide reliable software and a stable environment. The facility consists of a group of workstations which are networked together. These computers can be accessed locally in the Testbed, remotely using Internet, and via a modem over telephone lines. The VTS software should be designed to be portable to other platforms so that the toolkits ultimately can be installed at other locations. Documentation will be provided to assist software developers in installing, maintaining, and using the software.

The VTS software will interface with several external systems. The primary external systems are the operating system, compilers, the Express language, the STEP exchange file format, and interface display systems. Some of the sources for these external systems are known; others are being investigated. As other implementation specifications are developed within STEP, such as the STEP Data Access Interface, these will also be supported.

The design of the user interface for the Testbed software will be influenced by the needs of the users in terms of computer sophistication, mode of computer access, and familiarity with other software systems. The user interface should facilitate workflow between the VTS tools and toolkits. The interface should provide meaningful error reporting to the users. Documentation will be provided to assist users in using the VTS software.

Appendix A Priorities for Implementation of VTS Software

The VTS software will be developed and released according to a prioritized list of user requirements. The priorities specified below reflect those user requirements. The anticipated funding level will allow for the completion of all tools in Table 6.

Future releases of the toolkits will address those tools not included in the initial release and enhancements to the tools in the initial release. Tools in the initial release are expected to change in response to changes in the external systems to which they interface. As new versions of external systems, such as the operating system or Express, are made available, the tools will be enhanced to support the new versions of these systems.

A.1 Initial Release

The initial version of the *Model Scoping and Construction* toolkit will consist of a parser for checking Express syntax along with text editing capabilities.

The *Test Definition* toolkit will initially be limited to existing word processing capabilities.

The *Test Case Data Generation* toolkit is the first toolkit needed. The minimal system necessary to support STEP testing must allow a tester to read and write a STEP exchange file, and browse and modify its contents.

The *Test Execution and Analysis* toolkit will consist of a data browser and the tools to load the data. The data browser will be driven by the schemas under analysis.

TABLE 6 VTS Toolkits Initial Release Capabilities

Tools	Toolkits - initial release			
	Model Scoping and Construction	Test Definition	Test Case Data Generation	Test Execution and Analysis
Express Parser	x			
Word Processing System	x	x	x	x
Express Translator			x	x
STEP Data Editor			x	
STEP Data Browser				x
STEP Exchange File Parser			x	x

A.2 Future Release

After the initial release of the toolkits, requirements will be reviewed to determine whether additional features for the tools in the initial set are needed for testing purposes. In addition, a release schedule will be set for the tools which have not been provided in the initial release. The priority list for new features in the test system are expected to change with time, so the planning horizon for the VTS tools has been limited to single releases.

TABLE 7 VTS Toolkits Future Release Capabilities

Tools	**Toolkits - future release**			
	Model Scoping and Construction	**Test Definition**	**Test Case Data Generation**	**Test Execution and Analysis**
Configuration System	X	x	x	x
Diagramming Tool	x	x		
Express Browser	x	x	x	x
Word Processing System	X	X	X	X
Other Model Browsers	x	x		
Express Parser	X			
Express Translator			X	X
STEP Data Editor			X	
STEP Data Browser				X
STEP Exchange File Parser			X	X
IGES Translator			x	
Other Translators			x	
Data Converter			x	
CAx Systems			x	
Database System				x
Query Language				x
Logging Mechanism				x
Cross Referencing System				x

X - indicates tools from initial version

x - indicates new tools

References

[ANSI86] American National Standards Institute, Database Language SQL, Document ANSI X3.135-1986.

[ANSI89] American National Standards Institute, Programming Language C, Document ANSI X3.159-1989.

[ICA85] Information Modeling Manual IDEF Extended (IDEF1X), ICAM Project Report (Priority 6201), Air Force Systems Command, Wright-Patterson Air Force Base, OH, USA, December 1985.

[IGES] *Initial Graphics Exchange Specification*, Version 5.0, IGES/PDES Organization, NTIS PB88235452, National Technical Information Service, Springfield, VA, June 1990.

[ISO11] Information Modeling Language Express: Language Reference Manual, working draft N5 ISO TC184/SC4/WG5, Spiby, P., ed., January 1991.

[ISO21] Part 21 ISO TC184/SC4. Industrial Automation Systems -- Product Data Representation and Exchange -- Part 21: Clear text encoding of the Exchange Structure.

[Linton91] Linton, M,. InterViews Reference Manual Version 3.0-alpha, Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Silicon Graphics, January 1991.

[McLean90] McLean, C.R., National PDES Testbed Strategic Plan 1990, NISTIR 4438, National Institute of Standards and Technology, Gaithersburg, MD, October 1990.

[Mitch90] Mitchell, Mary, Validation Testing Systems Plan, NISTIR 4417, National Institute of Standards and Technology, Gaithersburg, MD, October 1990.

[Mitch91] Mitchell, M., A Proposed Testing Methodology for STEP, NISTIR, National Institute of Standards and Technology, Gaithersburg, MD, forthcoming.

[NIJ89] Nijsen, G., and Halpin, T., Conceptual Schema and Relational Database Design: A Fact Oriented Approach, Prentice Hall, Englewood Cliffs, NJ, 1989.

[Palmer91] Palmer, M., Gilbert, M., Guidelines for the Development and Approval of STEP Application Protocols, ISO/TC184/SC4 N1 Version 0.7 working draft, January 25, 1991.

[PDES90] PDES, Inc. Program Technical Development Plan for Phase II, PDES, Inc. PMG001.01.02, February 20, 1990.

[Scheifler89] Scheifler, R.W., X Protocol Reference Manual for Version 11, O'Reilly and Associates, Inc., Sebastopol, CA, 1989.

[SIIP91] An SDAI Subset for PDES, Inc. Prototyping and SDAI C++ Binding for PDES, Inc. STEP Implementation Prototype Package, Stephen Clark, internal report, April 1991.

[Stroustrup90] Stroustrup, B., ANSI X3J16/90-0020, C++ Language System Reference Manual.