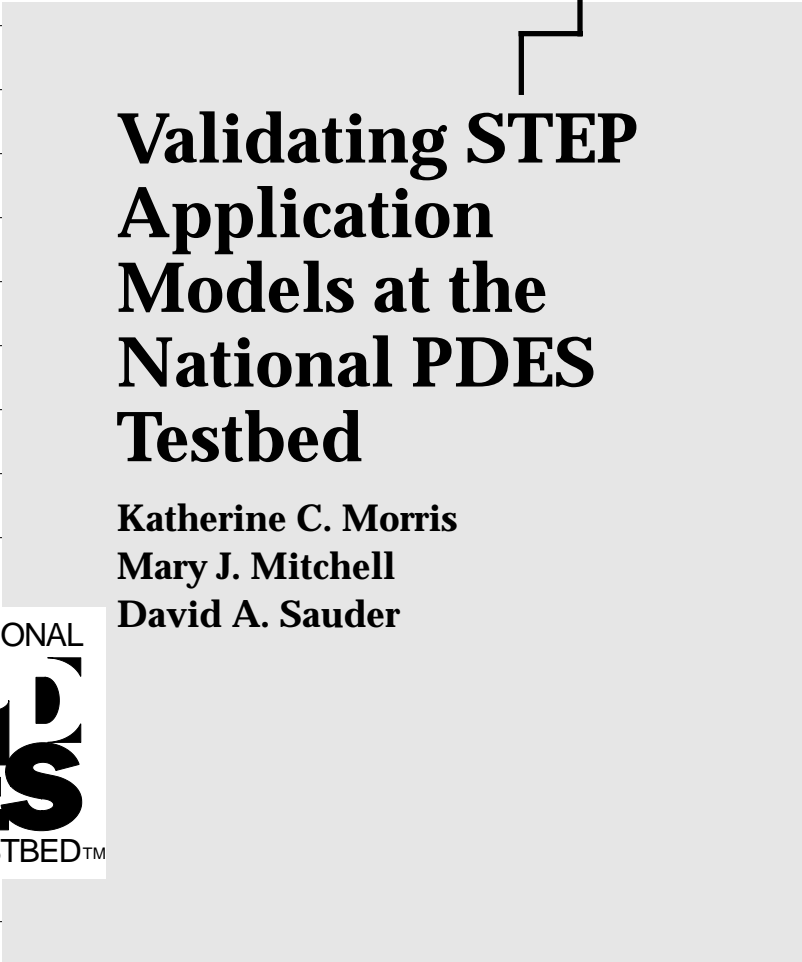


**National PDES Testbed  
Report Series**



**Validating STEP  
Application  
Models at the  
National PDES  
Testbed**

**Katherine C. Morris  
Mary J. Mitchell  
David A. Sauder**



National PDES Testbed



**Validating STEP  
Application  
Models at the  
National PDES  
Testbed**

**Katherine C. Morris  
Mary J. Mitchell  
David A. Sauder**

U.S. DEPARTMENT OF  
COMMERCE

Robert A. Mosbacher,  
Secretary of Commerce

National Institute of  
Standards and Technology

John W. Lyons, Director

December 1991



---

# Validating STEP Application Models at the National PDES Testbed

---

**Katherine C. Morris**  
**Mary J. Mitchell**  
**David A. Sauder**

---

## **Abstract**

This document is part of the National PDES Testbed Report Series and is intended to complement the other reports of the Validation Testing System (VTS) project. Specifically, other documents are available which fully describe the model validation methodology used in the Testbed, software requirements for the VTS, and details of the software which automates that methodology. These documents are referred to throughout this report.

The problem of sharing data has many facets. The need for the capability to share data across multiple enterprises, different hardware platforms, different data storage paradigms and systems, and a variety of network architectures is growing. The emerging Standard for the Exchange of Product Model Data (STEP), a project of the International Organization for Standardization (ISO), addresses this need by providing information models which clearly and unambiguously describe data. These models are organized into *application protocols*. An application protocol addresses the data sharing needs for a particular application area. STEP integrates the information requirements from all the application protocols. The validity of these information models is essential for success in sharing data in a highly automated environment.

This document describes how application models will be validated in the National PDES Testbed at the National Institute of Standards and Technology. (PDES, Product Data Exchange using STEP, is the U.S. effort in support of the international standard.) Application model development and testing is a complex process which involves synthesizing, analyzing, and manipulating large amounts of diverse information. Most of the process relies exclusively on human capabilities for analysis, judgment, and interaction; however, part of this process can and should be automated. A strategy for automation is based on an analysis of the information flow needed for the development and testing process and initial experiences with automation for validation testing at the National PDES Testbed.



---

---

## Table of Contents

---

	Abstract .....	iii
	Table of Contents .....	v
	List of Figures .....	v
	List of Tables .....	v
1	Introduction .....	1
2	Overview of Validation Testing .....	2
3	Activities of the Validation Testing Methodology .....	5
4	Automation for the Validation Testing Methodology .....	9
5	Validation Testing at the National PDES Testbed .....	15
	5.1 Experiences with the Interim System .....	15
	5.2 Future Directions for the Validation Software .....	17
6	References .....	19
APPENDIX A	VTS Document Series .....	21

---

## List of Figures

---

FIGURE 1	Process Model of the Validation Testing Activities .....	4
FIGURE 2	Data Flow Between Toolkits .....	13

---

## List of Tables

---

TABLE 1	Activities and Toolkits of the Validation Testing System .....	3
TABLE 2	VTS Tools .....	10
TABLE 3	Composition of the VTS Software Toolkits .....	11
TABLE 4	Information Flow Between Toolkits .....	12



## 1 Introduction

---

Confidence in a standard by its user community is absolutely essential for a standard to gain acceptance. Proof that the standard is properly defined and that it can be used successfully will help achieve user confidence. A rigorous testing program is the foundation for any useful standard. Appropriate testing before standardization can ensure that a draft specification indeed meets the functional requirements for the standard.

The Standard for the Exchange of Product Model Data, commonly referred to as STEP<sup>1</sup>, is an emerging international standard designed to provide a complete, unambiguous, computer-readable definition of the physical and functional characteristics of a product throughout its life cycle. Information model specifications contained in STEP are implementation independent, though distinct implementation interface techniques are defined to support applications relying on file exchange or shared databases.

An *application protocol* (AP) is a specification for a portion of the data described in STEP. This specification of information entities and corresponding usage constraints describes the product data requirements of a given application [Palmer91]. The whole of the STEP conceptual models consists of the logically integrated union of all STEP AP models. Thus the STEP AP specifications permit product information to be unambiguously exchanged or shared between implementations on dissimilar systems.

The National PDES Testbed<sup>2</sup> is used to test the validity of application protocols, or other application models<sup>3</sup>, which are proposed standards within STEP. The term *application model* refers to the component information model of an AP or a similar information model. Earlier in 1991, Mitchell [Mitch91] proposed a practical methodology for validating STEP APs. The Validation Testing System (VTS) at National PDES Testbed realizes that methodology. This document shows how the VTS applies software based on the proposed methodology to support AP validation. The methodology and software build on previous experience with testing of application models for STEP.

---

1. 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 (TC184) Subcommittee on Industrial Data and Global Manufacturing Programming Languages (SC4). For an overview of the standard refer to *Part 1: Overview and Fundamental Principles* [ISO1].

2. The National PDES Testbed is located at the National Institute of Standards and Technology. Funding for the Testbed Project has been provided by the Department of Defense's Computer-Aided Acquisition and Logistic Support (CALs) Office. The work described in this document is funded by the United States Government and is not subject to copyright.

3. The term *application model* will be used throughout this paper to refer to the domain specific conceptual schema which is under evaluation. The schema can be the component models of an AP, e.g., an application reference model and an application interpreted model, or a precursor to an AP such as a context driven integrated model (CDIM). While the validation methodology is applicable to this general class of application models, the first priority at the National PDES Testbed is to support the requirements for validation of Application Interpreted Models (AIM).

The intended audience for this document includes

- participants of AP projects who use the National PDES Testbed for validating their application models,
- organizers of other projects investigating methodologies and tools for development and validation of APs, and
- other organizations which intend to provide validation testing services and/or tools.

This document is part of the National PDES Testbed Report Series. Other reports expand on different aspects of the comprehensive view of the VTS illustrated in this document (see Appendix A.) Section 2 of this document gives an overview of the validation testing process for those readers who are unfamiliar with AP validation. Section 3 describes the activities which comprise the validation testing methodology. This section correlates the steps, described in the *Proposed Testing Methodology for STEP Application Protocol Validation* [Mitch91], with the activities described in other document in the VTS series [Mitch90][Morris91]. Section 4 describes the automation to support validation testing. Requirements for automation of the VTS are fully described in *Validation Testing System Requirements* [Morris91]. And Section 5 forecasts future directions for the Validation Testing System software based on experiences with the interim software being used at the National PDES Testbed. These experiences provide a basis for the VTS software architecture described in the document *Architecture for the Validation Testing System Software*.

---

## 2 Overview of Validation Testing

---

Application protocol development and testing is a complex process; it involves synthesizing, analyzing, and manipulating large amounts of diverse information. Most of the process relies exclusively on human capabilities for analysis, judgment, and interaction; however, part of this process can and should be automated. A strategy for automation is based on an analysis of the information flow needed for the development and testing process and initial experiences with automation for validation testing at the National PDES Testbed. This strategy is discussed throughout this document. This section describes the validation process in general.

Validation testing of STEP determines whether a developing application protocol does what it is intended to do, i.e., meets the requirements that led to its development. The proposed approach is to validate APs by simulating the behavior of the relevant applications. Since APs will be used for data sharing, the specific behavior that must be verified relates exclusively to data access requirements for the application areas.

Validation tests are identified by examining the functions of the applications in a particular area. The data required to perform each activity in the application processes is specified in detail. Realistic data is associated with each of the validation tests. The data needed to perform a specific activity is then mapped into the structures defined by the application model. The application model with its associated data is examined to determine if it can support the generation of the outputs needed by the applications. This approach to validation essentially simulates the behavior of an application system interacting with a user of the system.

This methodology for validating an application model can be decomposed into the six high-level activities shown in **Table 1**: Scoping the Application Context, Model



Construction, Test Definition, Test Case Data Generation, Test Execution and Analysis, and Model Refinement and Improvement [Mitch90]. Each of these activities may be performed by a separate group of people. Four software toolkits have been identified to support these activities. A toolkit consists of the set of software which automates or supports the particular activities for a phase of the AP development and testing process. Separate toolkits for each activity are not needed due to overlaps in the automation needs for the activities. **Table 1** identifies how the software toolkits correspond to the activities.

---

**TABLE 1**

---

**Activities and Toolkits of the Validation Testing System**

---

<b>Activity</b>	<b>Toolkit</b>
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

---

The validation process can be compared to the activities of a building code inspector. An AP developer or tester uses the VTS toolkits to examine STEP models in an application context, just as a building inspector uses a blueprint to examine aspects of a new building at its site. Constructing an application model is similar to an inspector's analysis of a new building to determine what should be tested. The AP tester defines tests to perform against the new application model, just as the inspector must formulate a plan for testing the building's features. The inspector identifies the appropriate building codes needed to test different features of the building for flaws. (For example, the building inspector would test the chimney flue on a house with a fireplace.) Likewise, product data for testing features of the application model is collected. Features are tested against known cases, just as a building inspector uses an instrument to check for correct voltages in electrical outlets. The VTS software is similar to the inspector's voltage meter -- it allows the tests to be conducted in a consistent manner, but the meaning of the tests is dependent on human analysis of what the instrument reports.

Together the activities and the toolkits which support them comprise the Validation Testing System, just as inspecting a building requires the process the inspector uses, as well as his instruments. The activities and toolkits of the VTS are discussed in the following sections.

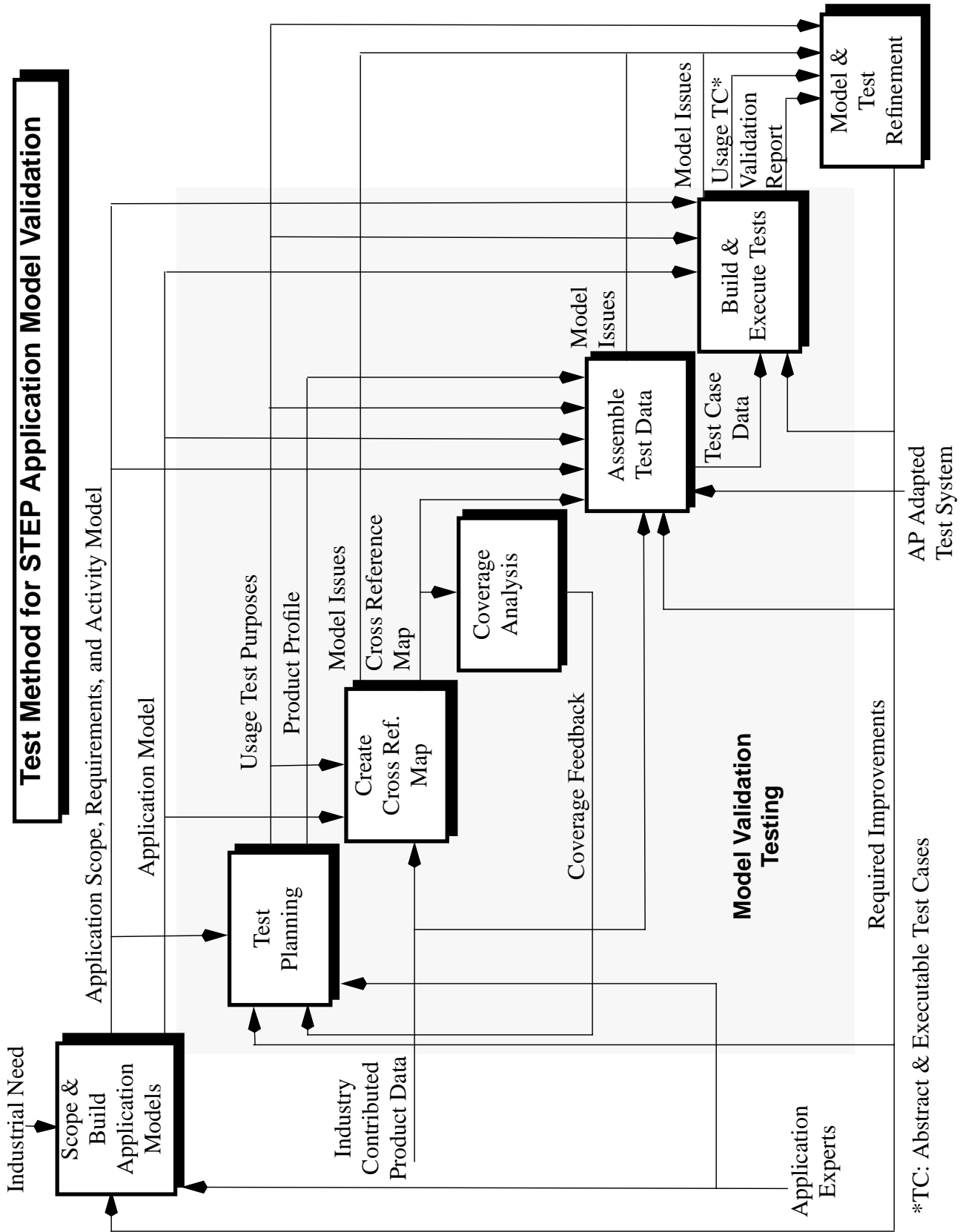


FIGURE 1

Process Model of the Validation Testing Activities

### 3 Activities of the Validation Testing Methodology

---

This section summarizes the validation testing methodology and the flow of information within it. The methodology is fully described in [Mitch91]. An overview is provided here as a background for the reader.

Application models are validated by simulating the data access patterns of the relevant applications. This process was originally decomposed into the six activities shown in **Table 1**. As the methodology for validation testing developed, the decomposition was refined to the seven steps shown in **Figure 1**. The six original activities directly reflected needs for automation of the validation testing system and are used as a basis for discussion in this document.

The IDEF0 [ICAM82] activity model in **Figure 1** depicts the seven steps in the validation testing methodology and shows the flow of information in the validation process. The first step corresponds to activities 1 and 2 below. In this step the application scope and model are constructed and released for testing. The scope controls what is in the application model and, therefore, guides what needs to be tested. (See the AP development plan [Stark91] for a detailed discussion of these activities.) Once the application model has been constructed, it is typically turned over to a different set of people: the testers.

The testing activities, which fall into the grey area in **Figure 1**, verify the correctness of the model that has been developed during the previous activities. The next three steps in **Figure 1** are discussed below as part of the Test Definition activity. In the second step, *test planning*, the testers decide what will be tested. In the third and fourth steps, *create cross reference map* and *coverage analysis*, further refinement and additional detail are provided to the set of tests previously defined. In these two steps, the testers acquire a detailed understanding of the application model. Then they gather and inventory industry contributed product data. Separate activities are not defined for these steps because they do not generate new requirements for software tools to automate the process. These steps involve expanding the testers' understanding of the problem and include interaction with the potential contributors of product data. The remaining three steps in **Figure 1** correspond one-to-one with the activity descriptions given below.

#### 1. Scoping the Application Context

The activity of Scoping the Application Context identifies a formal technical boundary for the application model by examining the functions of the application. The boundary is defined by analyzing the general processes the application performs. The manner in which the application model will be used (e.g. exchange files or shared use) may constrain these processes. A decision is made on whether each process input or output will be supported by the AP. A study of the information needs of the application processes establishes categories for the information. The categorization clarifies what capabilities the application model should support. The resulting activity model, which illustrates the scope of the application model, is reviewed by experts in the application area to ensure that it reflects common business practices. The results of Scoping the Application Context are the following:

- an activity model which defines the application processes and their interfaces, and
- a statement of the scope and requirements.

## 2. Model Construction

During the Model Construction activity detailed information models are constructed and integrated with existing standard information models. Interviews with experts in the application area provide detailed information requirements and an understanding of any constraints on the use of this information. The requirements are driven into a detailed conceptual data model of the application information. The detailed data model is called an Application Reference Model (ARM). Next, segments of the existing STEP integrated resource model specifications [ISO41, ISO1], which meet the needs of the ARM, are identified. Any application specific restrictions and preferences are also specified at this time. An Application Interpreted Model (AIM), the application model to be tested, is produced. The AIM supports the requirements specified in the ARM but is based on the information structures specified in the integrated resource models. For an AP the AIM is to be provided in three formats:

- a graphical representation, such as Express-G [ISO11],
- a formal specification in Express [ISO11], and
- a completely documented description including definitions and illustrations.

The result of activities 1 and 2 is a complete description and understanding of the problem that the application model is addressing. Analyzing and integrating the information requirements into a single information model, the ARM, contributes to the testers' understanding of the application area. The further integration of this model with other information models within STEP produces an information model which is specific to an application area and also consistent with other phases in a product life cycle.

These results -- the understanding of the domain and integration with the existing information models -- are represented in **Figure 1** as the output *Application Scope, Requirements, and Activity Model*. This result involves intensive analysis and judgment which could benefit from computer-aided support. The other output in **Figure 1**, *Application Model*, is the computer-interpretable representation of that information model. This result is the portion of the output which is used to automate the testing process.

## 3. Test Definition

The primary result of the Test Definition activity is a plan for validating the application model. The information produced by this activity is informative and guides the rest of the testing process but is not computer-processible. The test plan describes how typical application data access requests will be satisfied using representative data. The test plan provides the organization to manage the complexity of the required tests. The plan consists of tests based on the usage requirements and includes an understanding of the types of and sources for data needed to conduct the tests. These activities are depicted in three steps of **Figure 1**: *Test Planning*, *Create Cross Reference Map*, and *Coverage Analysis*.

During Test Definition activity the testers define testing requirements, develop a test plan to be used to evaluate the functionality of the application model, and identify and gather test data. The testers consolidate the results of interviews with experts into realistic usage scenarios that describe the data access needs for the application area. The usage scenarios identify significant combinations of information and organize the details of how that information is used. Each significant combination of information,

that identifies non-redundant and realistic test conditions, is called a test purpose. Test purposes are organized into test groups based on functionality.

In addition to the test plan, this activity includes three outputs for analyzing the characteristics of the test data: a product profile, a cross reference map, and coverage feedback. A product profile describes characteristics of the information and is used for gathering representative test data. The cross reference map indicates the correspondence between the application model and the test data. The creation of the cross reference map frequently uncovers major structural flaws in the application model. The coverage analysis of the representative test data reveals unused segments of the application model. If the AP project cannot identify data which corresponds to these segments, then the application model needs to change. Either the model was misunderstood, in which case better documentation is added to clarify the model and appropriate data is found; or the search to find corresponding data was exhausted, in which case the unused segments are ultimately removed from the application model.

This activity produces six outputs:

- an overall test plan with test groups,
- test purposes with usage constraints,
- a product profile,
- identification of representative test data,
- a cross-reference map to correlate the test data with the application model, and
- a report which describes model issues and needed improvements to the application model.

#### 4. Test Case Data Generation

During the Test Case Data Generation activity, represented in **Figure 1** as the *Assemble Test Data* step, the testers assemble and build the product data based on the product profile for a given usage scenario. The objective is to identify where in the application model the representative product data will reside. Each piece of industry-contributed data should have a single, logical place in the model. In the initial testing experience [PDES90] less than half the data needed for simulating the application was available in electronic form. More than half of the test product data was generated by hand. This was the most labor-intensive and error-prone, but potentially the most reusable, output of the entire process.

The computer-processible output of this activity is the test case data. Another important output of this activity is the detailed description of the usage test purposes, which are fully documented as abstract test cases<sup>4</sup>. This activity results in the following output:

- detailed test data in a format suitable for processing by the VTS software, e.g. STEP exchange file format [ISO21],
- abstract test cases, and
- a report which describes model issues and needed improvements to the application model.

### **5. Test Execution and Analysis**

The Test Execution and Analysis activity involves the development, execution, and analysis of the test cases against the application model. In order to execute the test cases, a computerized testing environment needs to be established and the test cases need to be formally specified with respect to the testing environment. Analysis of the test cases involves comparing the test results to the expected results to determine the validity of the application model. In addition, general statements, about what any implementation of the AP must support, are documented. This activity produces the following results:

- test reports,
- additional abstract test cases,
- improved test product data for reproducing test results,
- executable test cases for reproducing test results, and
- a report which describes model issues and needed improvements to the application model.

### **6. Model Refinement**

The Model Refinement activity resolves issues that were uncovered during the testing process. Alternative solutions are proposed and the best solution is selected. Once there is agreement on how to resolve an issue, the model is modified and a new model is released for validation testing. This activity results in the following information:

- the refined application model,
- an issue resolution statement describing the solution selected and supporting rationale, and
- refined test purposes and abstract test cases.

Model validation testing is an iterative process. The end result of the process is an application model suitable for inclusion in STEP as part of an AP. The model must be both useful and usable to be part of the standard. The involvement of a variety of application experts in the validation process ensures that the model is useful. There should also be

---

4. An abstract test case is the complete, implementation-independent specification of the actions required to achieve a specific test purpose. This includes the plan of what aspect of the model to test, the data to be used to test this aspect, and the expected results of the tests. These test cases may ultimately contribute to the abstract test cases needed for conformance testing. See *Part 31: Conformance Testing Methodology and Framework: General Concepts* [ISO31] for more information about the needs for conformance testing.

reviews by application experts independent of the AP project. To ensure that the model is usable, validation testing should be repeated until the model satisfactorily supports the information needs identified in the test plan. Once the utility of the application model has been demonstrated, the next component of an AP can be developed. (I.e., once the AIM has been validated, the specification of conformance requirements<sup>5</sup> can proceed.)

The validation of an application model is dependent on the application area under consideration, but the validation process itself is constant and many aspects of it can be automated or supported by automation. Due to the nature of the standard being developed, it is mandatory that some parts of the process are automated. The standard will enable the automatic sharing of data. Therefore, the ability to automatically access data using the application model needs to be verified. The following section discusses how this is accomplished.

---

## 4 Automation for the Validation Testing Methodology

---

This section describes the four software toolkits which support the validation testing methodology and the data flow between them. The software simulates the information access requirements for the application area being tested. The VTS software will provide a controlled environment for validation testing, thereby reducing the potential for introducing errors into the process. The VTS toolkits and the control of the supporting environment will also reduce the level of computer sophistication and interaction needed so that the users of the system will be able to concentrate on validating the application models.

**Table 1** from the introduction identifies the four toolkits and their correspondence with the validation testing activities. The four toolkits are 1) *Model Scoping and Construction*, 2) *Test Definition*, 3) *Test Case Data Generation*, and 4) *Test Execution and Analysis*. All the toolkits support the validation testing process. The primary requirement of this process is the capability to manipulate and represent a particular application model in a variety of ways. Therefore, many functional requirements are common among the toolkits [Morris91]. **Table 2** summarizes and briefly describes the software requirements of the validation process. Some of these tools -- such as the CAx<sup>6</sup>, database, and word processing systems -- are available as commercial systems. For these tools commercial systems will be used in the VTS software. Other tools are either available from related projects or will be developed for the VTS.

---

5. The specification of AP conformance requirements is evolving. The relationship between AP validation and conformance testing is being defined within STEP.

6. 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.

**TABLE 2**

**VTS Tools**

<b>Tool</b>	<b>Description</b>
CAX Systems	Provides capability for producing product data.
Configuration System	Supports the management of documents and other files, including programs, used by the VTS system.
Cross Referencing System	Supports management of the relationships between specific tests and relevant sections of the application model.
Data Converter	Translates a data file corresponding to a particular application schema to an updated format based on changes made to the schema.
Database System	Provides for persistent storage of and shared access to data.
Diagramming Tool	Supports display and creation of diagrams which illustrate the concepts represented in an application model.
Express Browser	Displays the contents of a model written in Express in a “user friendly” format (e.g. Express-G.) Provides minimal hypertext-like capabilities.
Express Constructor	Assists in the construction of Express descriptions of an application model. May be graphical, such as Express-G or context sensitive textual editing for Express.
Express Parser	Parses an application model specified in Express to verify syntactic correctness.
Express Translator	Translates an application model written in Express into a program work space.
IGES Translator	Reads an IGES file and outputs a corresponding STEP file.
Logging Mechanism	Records results of a session during which the tests on the data were conducted. Provides extensive error reporting.
Other Model Browsers	Display the application model in a variety of different formats for reference. Do not provide the capability to modify model. Could be simple drawing packages.
Other Translators	Translate data from a CAX system’s internal format to STEP format.
STEP Data Editor	Provides an interactive environment for the display, manipulation, and editing of data that corresponds to an application model. Reads and writes data in STEP exchange file format.
STEP Data Browser	Provides an interactive environment for the display and manipulation of data that 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.
Query Language	Provides capability to represent data access requests in a format that can be executed during testing. Dependent on the database system.
Word Processing System	Supports editing functions and provides context sensitivity for standard formats.



The decomposition of tools depicted in **Table 3** represents the automation needed to support the activities covered by the individual toolkits. Several of these tools are shared by all four toolkits. The implementations of these toolkits within the National PDES Testbed may be limited by available resources.

**TABLE 3**

**Composition of the VTS Software Toolkits**

<b>Model Scoping and Construction</b>	<b>Test Definition</b>
Diagramming Tool	Diagramming Tool
Express Browser	Express Browser
Word Processing System	Word Processing System
Other Model Browsers	Other Model Browsers
Express Parser	
Express Constructor	
<b>Test Case Data Generation</b>	<b>Test Execution and Analysis</b>
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
Logging Mechanism	Logging Mechanism
IGES Translator	Database System
Other Translators	Query Language
Data Converter	Cross Referencing System
CAX systems	

Each of the activities of the VTS consumes and produces data. The information produced by one activity is required in the subsequent activities. A subset of the information produced is directly processible and is used to automate the activities. This automation parallels the flow of information between steps illustrated earlier in **Figure 1**. **Table 4** illustrates information inputs and outputs of the four toolkits; the entries in **bold** represent the computer-processible portion of the information which flows between the toolkits. The remainder of this section focuses on that information. Mitchell [Mitch91] discusses the more general information flow of validation testing.

**TABLE 4**

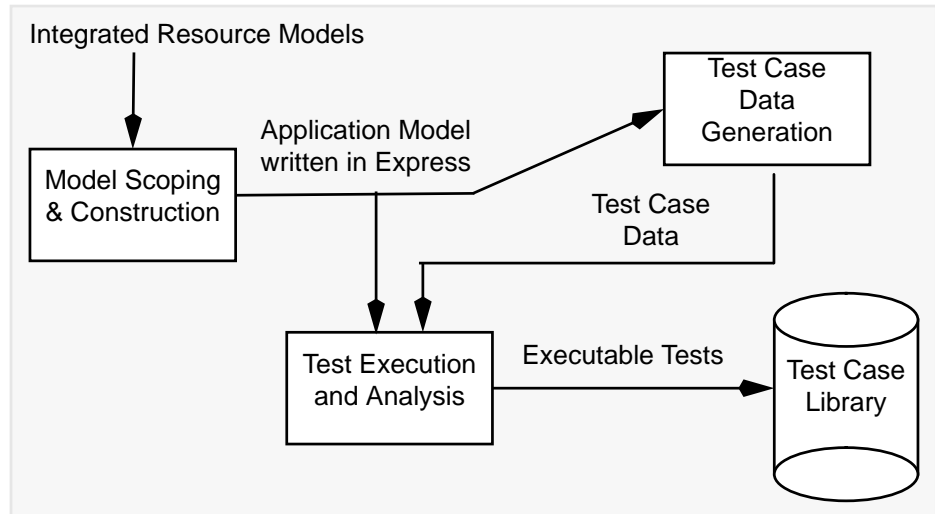
**Information Flow Between Toolkits**

	<b>INPUT</b>	<b>OUTPUT</b>
Model Scoping and Construction	Application Requirements from Experts <b>STEP Integrated Resource Models</b>	Scope & Requirements Statement Activity Model <b>Application Models Written in Express</b> Graphical Representations (i.e. Express-G)
Test Definition	Application Requirements from Experts Scope & Requirements Statement Activity Model Application Model (for reference)	Test Plan with Test Purposes Product Profile Cross-Reference Map Sources for Product Data Model Issues
Test Case Data Generation	Scope & Requirements Statement Activity Model <b>Application Models Written in Express</b> Test Plan with Test Purposes Product Profile Contributed Product Data (e.g., <b>IGES files</b> )	<b>Test Case Data</b> Abstract Test Cases Model Issues
Test Execution and Analysis	Scope & Requirements Statement Activity Model <b>Application Models written in Express</b> Test Plan with Test Purposes Abstract Test Cases <b>Test Case Data</b>	<b>Executable Test Cases</b> Validation Reports Testing Software Enhancements Refined Abstract Test Cases Model Issues

As shown earlier in **Table 1**, the *Model Scoping and Construction* toolkit supports three activities of the validation testing process: *Scoping the Application Context*, *Model Construction*, and *Model Refinement and Improvement*. These activities involve intensive analysis and judgment and, therefore do not allow for much direct automation. This toolkit assists in referencing and constructing application models and preparing documentation but does not provide more complex automation. **Figure 2** illustrates the flow of data between toolkits, as represented by the items shown in bold in **Table 4**.

FIGURE 2

## Data Flow Between Toolkits



The output of the *Model Scoping and Construction* toolkit is the application model in both human- and computer-interpretable formats. These formats include documentation describing the application model, a graphical representation of the model (Express-G), and the definition of the model in Express. While these different views of the application model are used in the other toolkits, only the Express version of the model is directly used as a basis for the other toolkits. The data consumed by this toolkit is the existing STEP conceptual models, which provide a basis for the application model being produced. The input conceptual models are a reference source and are not modified.

The *Test Definition* activity, which is supported by the *Test Definition* toolkit, also involves a great deal of human interaction and analysis and is the least automatable activity. As with the earlier activities, the automation is limited to assistance in referencing the application and activity models and in the preparation of documentation. None of the outputs of this toolkit are directly used by the other toolkits.

The primary automation for the *Test Case Data Generation* activity, which is supported by the *Test Case Data Generation* toolkit, is for assistance in preparing test data. The toolkit is initialized by configuring the tools to support the application model produced by the *Model Scoping and Construction* toolkit. This toolkit consumes product data from external sources and produces data which corresponds to the application model. The data consumed is represented in many formats, while the data produced is available in STEP exchange file format and may also reside in a database system.

A reliable and efficient way to obtain a subset of the required test data is in the form of an Initial Graphics Exchange Specification (IGES) [IPO91] file extracted from a CAD system. The IGES file can be translated into the format of the application model in the *Test Case Data Generation* toolkit. This IGES file represents a geometric model of a product and can provide up to 25% of the data needed depending on the application requirements; however, it covers only about a dozen of STEP geometric representation entities. Additional data needs to be provided manually to complete the information required for the application model.

The *Test Execution and Analysis* toolkit assists in the activity of generating executable test cases, executing the test cases, and analyzing the results. This activity allows for a great degree of automation: The tests can not be manually conducted -- they must be automated to validate the application models. In order to execute the test cases, a software environment for testing needs to be set up, and the test cases need to be formally specified in that environment. A database management system is used for the execution of the test cases.

The typical testing scenario is as follows:

1. the application model, provided in Express by the *Model Scoping and Construction* toolkit, is represented in the database system;
2. data, prepared using the *Test Case Data Generation* toolkit, is loaded into the database;
3. the executable test cases are specified in the system's query language;
4. these queries are executed to simulate the typical application data access requests against the data in the database; and
5. the results are analyzed.

Currently the primary means of transferring data into the database is via files in the STEP exchange file format [ISO21]. However, the VTS software will be designed so that data can be directly stored in the database by the *Test Case Data Generation* toolkit. The *Test Execution and Analysis* toolkit will use that same database, thereby avoiding the transfer of data between the two toolkits. In this case the *Test Execution and Analysis* toolkit will appear as enhancements to the *Test Case Data Generation* toolkit.

The *Test Execution and Analysis* toolkit is used to generate executable test cases based on the test plan and abstract test cases developed earlier. These test cases are then executed using the data which resides in the toolkit. This toolkit produces the results of executing the tests and the computer-interpretable test cases. Both of these products are available for re-testing of the model after it has been improved. The results are also used in the analysis and refinement activities.

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 testers, when they use the remaining two toolkits. The *Test Case Data Generation* toolkit is used to assemble product data to be used for testing. The *Test Execution and Analysis* toolkit is used to prepare executable test cases and to execute these test cases against the product data. The results of the tests are used for model analysis and refinement. The test cases themselves are saved for future testing activities and will contribute to the AP documentation for the standard.

The model refinement activity leads to a new application model and may contribute to refinements to the STEP models. Throughout the testing process, any deficiencies in the application model, the test cases, or the test environment are documented, and appropriate enhancements are made. The entire validation process is repeated using the refined model, test data, and testing environment.

The information flow between the toolkits reflects the information flow in the application area outside of the testing environment. For example, consider the application area of configuration control of product designs:

The application model represents a data model used for configuration control within an organization. The application model is developed in the testing environment by interviewing experts in the application area. A software designer would interview experts in configuration control and in design applications to develop a data model for the organization.

The data and its usage in the validation testing process directly reflect the needs of the application area. The data model is populated with data, using the same access paths that an organization would need to use to enter new products or designs into their system. Much of the data used for validation testing is contributed from industrial sources involved in the application area. The tests performed on the data ensure that it is accessible as required based on the usage scenarios developed in the test definition activity. The usage scenarios describe the information requirements for configuration control of product designs. The library of executable test cases is based on reports or queries that are commonly used by configuration control applications.

---

## **5 Validation Testing at the National PDES Testbed**

---

The National PDES Testbed has been used for STEP validation testing since 1989. The software currently in place in the National PDES Testbed, described in detail in [Breese91], provides some of the automation desired for the Model Construction, Test Case Data Generation, and Test Execution activities. This section summarizes the direction for future improvements and additions to the validation testing software at the Testbed. These plans are based on past experiences with validation testing in the Testbed, which helped clarify the needs for validation testing software.

### **5.1 Experiences with the Interim System**

The current software which supports the testing process consists of a set of independent tools collected from a variety of sources. As a result of their origins they operate in a variety of hardware and software environments. The current method of sharing data throughout the testing process is by exchanging data files between these tools. This environment requires data translation, which introduces the potential for errors or inconsistencies, every time data is moved between activities in the testing process. Moreover, the translation process and the associated, manually activated, process of importing and exporting data is time-consuming.

Automation for the validation testing process is currently provided by a set of software tools which translates both the application model and the test data among a variety of formats [Clark90a]. The existing software includes: a data editor; a relational database management system; an Express compiler and translators for representing the application model in the data editor and database system; an exchange file parser and loaders for populating the data editor and database; data export facilities for extracting data from the editor, database, and a commercial CAD system; an IGES to STEP translator, which handles a very limited subset of STEP data, for providing some testing data; and

a visualizer and geometric modeler for displaying and manipulating a limited set of data.

The current system lacks some necessary functionality and provides unacceptable performance in other areas. Significant improvements can be made in the following areas:

- performance, in terms of both computation time and reliability;
- workflow automation, in order to minimize the need for manual intervention between automated parts of the testing process;
- adoption of a data representation paradigm which more closely resembles that of Express; and
- expansion of the functionality to cover the requirements of model scoping and construction.

The most significant improvements to the existing system can be made by replacing the existing data editor and database management system with an improved, integrated editor and database system. Both of these tools have suffered from significant performance problems with large sets of data [PDES90]. The integration of the two will assist in reducing the performance problems, help in automating the workflow, and align the representation paradigm with that of Express.

The current data editor, QDES [Clark90b], was developed as a prototype; it was not designed to be used as a production system. Performance problems with the database loader are magnified by the fact that the mapping for each application model has not been optimized for the relational database management system being used. Instead, a general-purpose mapping was applied so that new database schemas could be automatically generated.

Another significant improvement to the existing system can be made by providing a more integrated work environment. The objective of integrating the software is to eliminate the need for manual intervention at a number of points in the validation process. The current system requires that the test case data developed on a CAx system to go through multiple translations or processing steps before it is usable for executing a test. Each translation or processing step requires manual intervention which creates an opportunity for introducing new errors or using the wrong version of a file.

In addition, many necessary functions are currently supported by separate utilities. The use of some of these utilities requires that the data is exported from the data editor or database, run through the utility, and imported back into the system; other utilities must be run before data can be loaded into a system.

The data editor and database system in the interim system use different data representation paradigms for representing the information involved in the testing process. This places a burden on the testers, the users of the validation software, to understand the different representational formats for the information model and corresponding data and the relationship between these formats. This is one of the more difficult problems facing the testers.

Express uses a representation paradigm which allows a hierarchical decomposition of information as well as the representation of a semantic network of information. This representation is much richer than that provided by the relational paradigm [Date90] used in the interim database implementation. The relational paradigm reduces all infor-

mation to flat table structures and does not support any of the semantics of the associations between the tables. The data editor used in the current software uses an object-oriented paradigm [Gold85], which more closely resembles Express.

The testers are faced with learning not only the method of representing the application model in Express but also the format for representing the information in the various tools used in the testing process. The relational database system in the interim system uses the most different paradigm for representing data. The meaning of the representation in the relational system must be inferred, based on the individual tester's interpretation of Express and the tester's understanding of the mapping of the model into the relational database system. As with any interpretation, this leaves room for ambiguity and misunderstanding.

## 5.2 Future Directions for the Validation Software

The software needs for the STEP AP validation process can be divided into two categories:

- automation *of* the validation process, by simulating the data access requirements of the application area; and
- automation *to support* the validation process, through assistance for preparing documentation and for referencing and browsing the application model.

The first category -- automation of the process through simulation of the data access requirements for the application area -- is mandatory for validation testing. The testing process is not complete if the tests are not computer-processible and repeatable. These tests reflect the intended usage of the application model. This is the only need addressed by the interim system, and improvement of this software is the first priority for future implementation efforts.

The second category -- automation to support the validation process -- includes assistance in document preparation and for referencing and browsing the application model in any of the various formats. This need is currently met by commercial word-processing and drawing packages. These packages, however, provide only limited support for these functions. Support for this functionality will contribute to all of the activities of the validation process; however, it will not be addressed until the first need for automation *of* the process is satisfied.

Two tools are most important for supporting the first category of automation -- simulation of the information requirements for the application area:

1. a structured data editor and
2. a database system with query language.

The tools being used in the interim system for these functions do not satisfy performance and reliability requirements. Furthermore, since the tools are not integrated, problems arise related to data translation, model and data configuration, and clarity due to the use of different data representation paradigms. These problems with the interim system are discussed in the previous Section.

Initial efforts for the VTS software will focus on developing a data editor. The existing editor is the weakest component of the interim system. The replacement will provide

better performance, be more reliable, and support a broader range of functions. Many of these functions are currently supported by separate utilities.

The new data editor will be developed so that it can be integrated with a database system. However, use of the editor will not depend on having a database system. Furthermore, the integration of the database into the system will not cause the end-user interface to change, although additional operations may be available for the user as a result of adding the database to the system.

The VTS software will provide an integrated set of functions for addressing the four toolkits. The integration of the functions will provide a more efficient environment and will better simulate the data access needs of the applications areas being tested. The VTS software will:

- reduce the frequency of data translation and human intervention which will reduce the number of errors;
- provide better and more extensive error checking which will reduce the potential for errors;
- improve performance and automation of the workflow, which will reduce the amount of time needed for the testing process;
- provide more sophisticated support for data editing and creation, which will reduce the inconsistencies in the data;
- provide a single interface to the software, which will reduce the time needed for learning to use the different tools.

The future environment will allow the users to operate on the same data set throughout the testing process without the need for translations to share data between toolkits.



## 6 References

---

- [Breese91] Breese, J. Newton, Michael McLay, and Gerard Silvernale, Validation Testing Laboratory User's Guide, NISTIR 4683, National Institute of Standards and Technology, Gaithersburg, MD, October 1991.
- [Clark90a] Clark, S. N., An Introduction to The NIST PDES Toolkit, NISTIR 4336, National Institute of Standards and Technology, Gaithersburg, MD, May 1990.
- [Clark90b] Clark, S.N., QDES User's Guide, NISTIR 4361, National Institute of Standards and Technology, Gaithersburg, MD, June 1990.
- [Date90] Date, C. J., An Introduction to Database Systems: Volume 1, Fifth Edition, Addison-Wesley, 1990.
- [Gold85] Goldberg, A. and D. Robson, Smalltalk-80: The Language and Its Implementation, Addison-Wesley, Reading, MA, July 1985.
- [IPO91] The Initial Graphics Exchange Specification (IGES), Version 5.1, IGES/PDES Organization, NCGA, Fairfax, VA, September 1991.
- [ISO1] ISO 10303 Industrial Automation Systems -- Product Data Representation and Exchange -- Part 1: Overview and Fundamental Principles, Working Draft, ISO TC184/SC4/WGPMAG/N43, Mason, H., ed., October 7, 1991.
- [ISO11] ISO 10303 Industrial Automation Systems -- Product Data Representation and Exchange -- Part 11: Description Methods: The EXPRESS Language Reference Manual, Committee Draft, ISO TC184/SC4/WG5/N14, Spiby, P., ed., April 29, 1991.
- [ISO21] ISO 10303 Industrial Automation Systems -- Product Data Representation and Exchange -- Part 21: Clear Text Encoding of the Exchange Structure, Committee Draft ISO TC184/SC4, Van Maanen, Jan, ed., March 12, 1991.
- [ISO31] ISO 10303 Industrial Automation Systems -- Part 31: Conformance Testing Methodology and Framework : General Concepts . Committee Draft, ISO TC184/SC4/WG6, January 14, 1991.
- [ISO41] ISO 10303 Industrial Automation Systems -- Product Data Representation and Exchange -- Part 41: Integrated Generic Resources: Fundamentals of Product Description and Support, Committee Draft, ISO TC184/SC4/WG4/N16, McKay, A., ed., September 6, 1991.
- [Mitch90] Mitchell, M., 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 Application Protocol Validation, NISTIR 4684, National Institute of Standards and Technology, Gaithersburg, MD, September 1991.
- [Morris91] Morris, K.C., McLay, M., and Carr, P. J., Validation Testing System Requirements, NISTIR 4676, National Institute of Standards and Technology, Gaithersburg, MD, September 1991.
- [Palmer91] Palmer, M., Gilbert, M., and Foster, J., Guidelines for the Development and Approval of STEP Application Protocols, ISO TC184/SC4/WG4/N15 Version 0.9 working draft, September 17, 1991.
- [PDES90] Test Report for Context-Driven Integrated Model (CDIM) Application A1, Skeels, J., ed., PDES, Inc. internal report PMG012.01.00, SCRA, Charleston, SC, April 1990.
- [Stark91] Stark, C., and Mitchell, M., Development Plan: Application Protocol for Mechanical Parts Production, NISTIR 4628, National Institute of Standards and Technology, Gaithersburg, MD, July 1991.

---

## APPENDIX A VTS Document Series

---

This document complements others in the National PDES Testbed Report Series which provide detailed technical information relating to the Testbed software. Those documents which specifically address aspects of the Validation Testing System are described below.

*Validation Testing Systems Plan* lays out the tasks and the overall approach for the initial implementation of the Validation Testing System. (NISTIR 4417)

*Proposed Testing Methodology for STEP Application Protocol Validation* describes the complete process used to develop and validate application protocols. This methodology document focuses on the analysis of application models and planning for validation testing. (NISTIR 4684)

*Validating STEP Application Models at the National PDES Testbed* describes a strategy for automation based on an analysis of the information flow in the application protocol development and testing process, and on initial experiences with automation for validation testing at the National PDES Testbed. (NISTIR 4735)

*Validation Testing System Requirements* describes functional requirements for automation of the VTS. This document also provides an overview of the VTS software environment. Requirements for the VTS system are driven by the STEP development effort and reflect the needs of the National PDES Testbed users. (NISTIR 4676)

*Architecture for the Validation Testing System Software* describes an architecture for software which supports the testing of information models for validity and correctness. The architecture provides a basis for software development within the National PDES Testbed. (NISTIR 4742)

*Validation Testing System Software Design* provides guidelines for the implementation of the VTS software. The document describes an architecture for creating and integrating software libraries to support the tools for the VTS. Designs for the components of the software are also provided. (forthcoming)