Equipment Automation Advanced Software Tester Product Description
SEMATECH and the SEMATECH logo are registered service marks of SEMATECH, Inc.

International SEMATECH Manufacturing Initiative and the International SEMATECH Manufacturing Initiative logo are registered service marks of International SEMATECH Manufacturing Initiative, Inc., a wholly-owned subsidiary of SEMATECH, Inc.

Product names and company names used in this publication are for identification purposes only and may be trademarks or service marks of their respective companies.
Abstract: This product description document (PDD) from the MFGM028M project describes the usage environment and required capabilities of an Advanced Software Tester (AST) to validate the conformance of the integration and automation capabilities of semiconductor manufacturing equipment primarily to SEMI standards and industry guidelines. The principal audience for this document is software suppliers who develop advanced equipment automation testers and IC makers and equipment suppliers that are the target market for such a product.

Keywords: Equipment Testing, Standards Conformance, Automation, Equipment Specifications

Authors: Charisse Rosier and Steve Fulton

Approvals: Steve Fulton, Project Manager
Brad Van Eck, Program Manager
Joe Draina, Associate Director, ISMI
Laurie Modrey, Technology Transfer Team Leader
Table of Contents

1 INTRODUCTION..................................................................................................................1
  1.1 Background ......................................................................................................................1
  1.2 Purpose and Intended Audience .....................................................................................1
  1.3 Objectives and Scope ......................................................................................................1
    1.3.1 Initial Scope of Application ......................................................................................2
    1.3.2 Eventual Use ...........................................................................................................2
    1.3.3 Out-of-Scope Items and Other Non-requirements ..................................................3
  1.4 User Community ............................................................................................................3
  1.5 Product Overview – The “Big Picture” ..........................................................................5
  1.6 Document Style and Organization ................................................................................6
  1.7 Definitions, Acronyms, and Abbreviations ....................................................................6

2 PRODUCT DESCRIPTION......................................................................................................9
  2.1 Environment ...................................................................................................................9
    2.1.1 Users ......................................................................................................................9
    2.1.2 Other Systems .......................................................................................................10
    2.1.3 Platform ...............................................................................................................10
  2.2 Capabilities...................................................................................................................10
    2.2.1 External Interfaces ...............................................................................................10
    2.2.2 Specific Product Features ...................................................................................11
    2.2.3 Priority of Standards Categories to be Tested .....................................................23
  2.3 Attributes .....................................................................................................................23
    2.3.1 Performance .........................................................................................................23
    2.3.2 Ease of Use ..........................................................................................................23
    2.3.3 Backwards Compatibility .....................................................................................24
    2.3.4 Flexibility ............................................................................................................24
    2.3.5 Reliability, Availability, and Fault Tolerance .......................................................24
    2.3.6 Safety and Security .............................................................................................24
    2.3.7 Maintainability ....................................................................................................25
    2.3.8 Software Portability .............................................................................................25
  2.4 Design Constraints .........................................................................................................25
    2.4.1 Cost ......................................................................................................................25
    2.4.2 Transportability .....................................................................................................25
    2.4.3 Use of Additional Third-Party Software ................................................................25
    2.4.4 Multiple User Support .........................................................................................26
    2.4.5 Co-existence With Other Software ......................................................................26

3 REFERENCES AND RELATED MATERIAL........................................................................26
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Advanced Software Tester User/Stakeholder Community</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>AST Product/Usage Overview</td>
<td>6</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Relating Test Requirements to Source Documents</td>
<td>12</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Data Model of Key AST Test Entities</td>
<td>14</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Requirements Traceability</td>
<td>16</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Data Model of Key AST Test Run Entities</td>
<td>20</td>
</tr>
<tr>
<td>Figure 7</td>
<td>User Security Model</td>
<td>24</td>
</tr>
</tbody>
</table>
Acknowledgments

Jackie Ferrell
Jim Hollister
Lance Rist
Alan Weber
Harvey Wohlwend
1 INTRODUCTION

1.1 Background
Validating the conformance of semiconductor manufacturing equipment to the industry’s integration and automation requirements is a complex endeavor. The relevant SEMI standards and their implementation interactions result in hundreds of potential requirements and behaviors. The availability of a first generation commercial test product, CCS Envoy, enabled quantitative testing and analysis of equipment software implementations, accelerating and improving the product delivered to production fabs. However, the scope of the tester was limited to the static implementation and interaction of the GEM 300 standards functions and requirements (i.e., one carrier/front opening unified pod [FOUP]). Currently no commercial test product can emulate a semiconductor factory’s intended use environment. Furthermore, emerging requirements and standards, such as Equipment Data Acquisition (EDA) and Recipe and Parameter Management (RaP), represent imminent needs far beyond the capabilities of current commercial testers.

1.2 Purpose and Intended Audience
The purpose of this product description document (PDD) is to define the requirements for a software product that can meet International SEMATECH Manufacturing Initiative’s (ISMI’s) view of the industry’s testing needs for the next 8–10 years. The PDD describes the usage environment and required capabilities of an Advanced Software Tester (AST). The document does not specify the tester’s architecture, implementation technologies, or development process to be used; however, it does identify several design constraints that may affect these items.

The principal audience for this document is 1) software suppliers that are interested in creating an advanced equipment automation tester and 2) IC makers and equipment suppliers that form the target market for such a product. Readers of this document should have a working knowledge of the SEMI equipment integration and automation standards to fully understand the intent and capabilities described herein.

1.3 Objectives and Scope
The basic purpose of the AST is to validate the conformance of the integration and automation capabilities of semiconductor manufacturing equipment to a specific set of requirements. Most of these requirements will be derived from SEMI standards and related industry guidelines, but others may be defined by the customers of the equipment (e.g., ISMI member companies).

Within this context, the AST must achieve four major objectives:

1. Create a general-purpose extensible test product that supports a wide variety of automation test requirements and technologies (i.e., not limited to specific semiconductor industry standards)
2. Enable users to create their own test cases to validate their particular requirements (while ensuring that these requirements do not compromise the integrity of the industry standards)
3. Support the traceability of test cases and test results to the underlying test requirements
4. Support a testing process that is more representative of the actual production environment (e.g., multiple load ports/control jobs, concurrent application client interactions, etc.)
To achieve these objectives, the AST will enable users to 1) capture and store basic information about the test requirements, 2) create and manage explicit test cases that address these requirements, 3) execute sets of these test cases in the context of a specific piece of semiconductor manufacturing equipment to produce test results, and 4) analyze and communicate these results. In addition, it will allow the creator of a test case to lock that information so that other users can run the test but not edit it. The tester will generate and track a variety of metrics about the testing and support traceability of the requirements throughout the process.

The AST will serve different purposes depending on the user’s perspective. For example, equipment suppliers who want to use it for testing individual requirements during equipment control system development will need a lot of flexibility in creating and running test cases. On the other hand, IC makers who use the AST to ensure the equipment meets the contracted specifications, which may go well beyond compliance to industry standards, may need support for custom automation requirements and execution of very specific manufacturing scenarios as well. Therefore, operation of the AST should be highly constrained when used in the equipment acceptance process.

Finally, the AST should be extensible so that it can test new communication protocols in the future. It should be delivered with a base set of test cases to demonstrate its fitness for this task. However, the definition of specific test requirements and the creation of test case content to address them are not in the scope of AST development.

Section 2 provides further detail about the environment and capabilities of the AST.

### 1.3.1 Initial Scope of Application

The driving force behind the AST is the need for testing the requirements that relate to the SEMI EDA and Generic Equipment Model (GEM) 300 standards (see Section 2.2.3 for an explicit list of the standards that are covered). Moreover, the AST will allow individual users to define and test their own requirements in addition to testing the base set of scenarios for the SEMI standards.

The tester is primarily intended to test requirements that can be validated by connection to the equipment’s communications interface(s), but in some cases, basic operator interaction with the equipment may also be required. This includes an analysis of the syntax, sequence, and semantics of messages that flow between the tester and the equipment and may also involve a deeper look at the structure of the information communicated (e.g., for E120, verifying that the schema of nodes in the Common Equipment Model [CEM] are properly constructed).

### 1.3.2 Eventual Use

The longer-term goal is to provide a flexible system that is extendable to test new automation requirements, standards, and capabilities without having to rework the application. As such, the tester should also provide a method for incorporating additional communication interfaces, technologies, and content as the standards evolve.

In addition to validating the equipment’s conformance to standards, the tester should support a certain degree of scale and performance testing. This will involve testing limits on file and message sizes, data transfer rate, tests under various load conditions, “Ironman” life tests, etc. (i.e., creating a test environment that reflects the expected production setting as closely as possible).
1.3.3 Out-of-Scope Items and Other Non-requirements

The tester is targeted at equipment-oriented standards and not intended to test host-level application standards such as Interface C (remote e-Diagnostics) and Interface B (e.g., Process Control System Interfaces). However, support for some of these may provide useful product differentiation for a tester supplier. The value of these potential extensions has not been quantified or prioritized.

A particular product implementation of the AST requirements set is not required to run test cases created using a different product implementation.

The tester does not need to support a great deal of explicit “error injection”; rather, it should be able to test the normal processing scenarios and as many of the equipment’s error codes and exception scenarios defined in the standards (and related requirements documents) as possible without compromising the safety of the equipment or the test environment.

1.4 User Community

A broad community of users and other stakeholders will be impacted by the AST. The needs of each have been carefully analyzed and included in the development of this document. The key relationships within this community are shown in Figure 1.

![Figure 1 Advanced Software Tester User/Stakeholder Community](image)

Specific roles, activities, and priorities within this community include the following:

- **AST Supplier**
  Develops, sells, and supports tester to support the needs of the semiconductor industry.

- **Automation Engineer**
  Specifies the automation requirements for semiconductor manufacturing equipment and related factory systems; will want to know these requirements are testable.
• **Equipment Engineering Manager (at OEM)**
  Relies on the AST to validate conformance of the equipment’s automation capabilities to industry standard test requirements.

• **Equipment Engineering Manager (at IC maker)**
  Specifies which equipment is to be tested; relies on the AST to validate conformance of the equipment’s automation capabilities to the full set of purchase specification.

• **Equipment Operator**
  Responds to prompts during an AST test run to modify the state of the equipment, fixtures, product material, or other elements of the test environment as necessary.

• **OEM QA Engineer**
  Maintains quality of the equipment supplier’s products; will use the AST to verify that the communications interfaces are working properly on every piece of equipment that is shipped.

• **Service Provider**
  Uses the AST when providing testing services to OEMs and IC makers; may also develop test content to support standard or custom test requirements.

• **Solution Provider**
  Develops standard communications interface products for use by OEMs and IC makers to reduce the time and cost of incorporating industry standards into their equipment and factory systems.

• **Test Developer**
  Inputs requirements information; builds and debugs test cases to address specific requirements; links test cases to requirements; provides technical support to other users regarding the content and behavior of test cases.

• **Test Engineer**
  Assembles test cases into test collections and test sequences to address a specific set of requirements; analyzes automation requirements and specifies requirements for additional test cases as needed; configures test sequences for a particular equipment type; validates requirements coverage of test sequences; interprets test results and generates summary reports for management review.

• **Test Operator**
  Configures test sequences for a specific piece of equipment and test run; runs test sequences to generate test results.

• **Test System Administrator**
  Installs AST, manages system configuration, administers user privileges.
1.5 Product Overview – The “Big Picture”

Modern semiconductor manufacturing equipment are miniature factories in themselves. They handle millions of dollars of product at a time, simultaneously processing multiple lots, responding to production operators, monitoring their own performance and health, and communicating with other factory systems. These capabilities are provided by the equipment’s embedded controller, which is usually a complex distributed system of industrial and commercial computing components and a lot of custom software.

Using hundreds of such pieces of equipment in an automated, high-volume wafer fab adds another layer of requirements, many of which are specified in SEMI standards. Individual manufacturers invariably extend these requirements with their own in pursuing competitive advantage and differentiation in factory productivity.

To properly test conformance to these requirements, the capability and complexity of the AST must reflect that of the equipment being tested. This is a major conceptual change from the current paradigm and will require a new architecture and a “bottoms-up” design. Specifically, in an AST environment, the user must be able to create individual test cases that are powerful, flexible, even autonomous to adequately exercise the equipment’s automation interfaces. While the choice of implementation technology is left to the AST supplier, the creation of test cases is basically a software development task. Test cases must be capable of multi-channel communications, human interaction, arithmetic calculations, conditional logic, looping, branching, event generation and handling, multi-threaded asynchronous operation, error handling, time management, and many of the other functions of on-line software systems.

The other crucial aspect of the AST paradigm is requirements traceability. The system will support the explicit linkage of test requirements all the way from their original source documents to the test cases that validate them to the coverage and conformance statistics of a specific piece of equipment. It also provides the bookkeeping functions to maintain these linkages and derive useful insights from them.

The rest of the capabilities described in the PDD build on this foundation.

Figure 2 provides a conceptual block diagram of the AST and shows how its key users interact with the system in a typical test configuration. Although people in other roles may also use the system, this constitutes the core group.

Requirements conformance testing is specifically intended to evaluate and quantify the degree to which end use requirements have been met, rather than the intrinsic quality of the software. This distinction is subtle, but profoundly influences the basic approach for conformance testing. To ensure the integrity and traceability of test results and outcomes, ISMI believes that a structured cumulative approach must be used. First, basic static or unit tests that determine that a requirement has been implemented, is properly structured and formed with appropriate messages and responses. This will answer questions like “Does the equipment support this message, that error code, this set of events, and that list of status variables?” and forms the statistical basis for the sampling of complex scenario coverage. Secondly, dynamic or “round trip” test, that exercise the function in operating interacting with other functions and requirements, ensuring it supports the necessary states and transition events, job processing, recipe downloading, remote parameter adjustment, and so on. Finally, scenario tests that address the requirements that stem from the complex interaction of multiple standards and that represent the use cases that will be found in the target production environment or the acceptance criteria for installation or contractual obligation.
1.6 Document Style and Organization

Insofar as possible, each requirement is expressed in a separate paragraph and assigned a unique requirement identification (ID) number and name. The use of “shall” and “should” corresponds to mandatory requirements and recommended/optional requirements, respectively.

1.7 Definitions, Acronyms, and Abbreviations

The following are definitions of terms and concepts used throughout the product description document (PDD). Note that special meanings have sometimes been assigned to otherwise common words/phrases in the context of the Advanced Software Tester (AST). This usage is not intended to constrain an AST supplier’s choice of terms in a product implementation.

Analysis Equipment
The AST conceptual modules used to evaluate and interpret results of a test run and assess conformance of the equipment’s automation software to the test requirements.

AST
Advanced Software Tester; the future product described in this document.

AST Entity
Any one of the identifiable data elements used by the AST (includes test requirements, test cases, test utilities, test collections, test sequences, test results, test logs, etc.).

AST Module
Any one of the major subsystems of the AST.
Attended/Unattended Operation  Presence (or not) of people during a test run; usually refers to the test operator and the equipment operator; the process performed by a test operator includes periods of both interactive, attended operation and unattended operation.

Categories  User-defined fields in the description of a test requirement, test case, or other AST entities that can be useful for browsing, displaying, forming test collections, linking test cases to requirements, etc.; example categories may include a type of requirement, shared source, common purpose, or designation of automation level.

Equipment  The piece of semiconductor manufacturing equipment under test.

Export/Import (an AST entity)  Conversion of AST entities into/from a form that can be used by other popular software packages (XML, .csv/.xls formats, etc.).

GEM 300  The suite of standards updated to support 300 mm automated manufacturing.

HSMS  High Speed Message Service; defines TCP/IP network communication used by GEM for host/equipment communication.

Interface A  The suite of SEMI Diagnostic Data Acquisition (DDA) standards, also referred to as Equipment Data Acquisition (EDA); at this writing, the suite includes E120, E125, E132, and E134.

ISMI  International SEMATECH Manufacturing Initiative

Lock  Prevention of an AST entity from being modified by anyone other than the author; locked entities may still be executed.

OEM  Original Equipment Manufacturer; another term for equipment supplier.

Parallel I/O  The communication mechanism (as defined in E84) used between semiconductor manufacturing equipment and automated material handling equipment to ensure secure handoff of material carriers between them.

Post-conditions  The description of the expected state of the equipment after a particular test case has been run.

Pre-conditions  The description of the state that the equipment must be in before a particular test case can be run.

Private  The content of an AST entity that could be considered IP cannot be viewed, copied, edited, exported (i.e., test case algorithms, test collection logic, etc.). The author sets the privacy of AST entities, which thereafter can be viewed only by the author or a user with AST system administration privileges.
<table>
<thead>
<tr>
<th><strong>Requirements Coverage</strong></th>
<th>A measure of how completely (usually in percentage) a particular set of test requirements can be validated using a test sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Results Repository</strong></td>
<td>The AST conceptual module that stores test results and other test run-specific information needed to interpret the test results.</td>
</tr>
<tr>
<td><strong>Run Specification</strong></td>
<td>See “Test Run Specification.”</td>
</tr>
<tr>
<td><strong>Scenario</strong></td>
<td>An end-to-end sequence of interactions that corresponds to a complete manufacturing operational cycle.</td>
</tr>
<tr>
<td><strong>SECS/GEM</strong></td>
<td>SECS Equipment Communications Standard/Generic Equipment Model; communications protocols between a host computer and the semiconductor manufacturing equipment. SECS (E4/E5) describes the format and content of individual messages while GEM (E30) defines how specific messages (and message sequences) are used to achieve certain automation capabilities. SECS/GEM is used to communicate with virtually every type of semiconductor manufacturing equipment (process, metrology, material handling, assembly, test).</td>
</tr>
<tr>
<td><strong>Source Document</strong></td>
<td>The document from which test requirements are derived; requirements may be explicitly identified at a level sufficiently granular for direct entry into the AST or may be implied in the text of the source document.</td>
</tr>
<tr>
<td><strong>Test Requirement</strong></td>
<td>An expression of a required capability for a piece of semiconductor manufacturing equipment or system; in the context of this document, it may describe the format, content, and sequence of messages that must be communicated; the structure of certain types of information; the implications of satisfying combinations of test requirements, such as precedence or ordering constraints, etc. Source documents for test requirements include SEMI standards, related ISMI documents, standards specifications from other industries, individual IC maker automation specifications, and perhaps others.</td>
</tr>
<tr>
<td><strong>Test Case</strong></td>
<td>A unit of software that validates the equipment’s conformance to a test requirement (or portion thereof) by communicating with the equipment and comparing its response to the expected result.</td>
</tr>
<tr>
<td><strong>Test Collection</strong></td>
<td>A group of individual test cases and/or test utilities and/or test collections that are run together in a prescribed order; a typical scope for a test collection might be one of the scenarios shown in the ISMI EDA Usage Scenarios document.</td>
</tr>
<tr>
<td><strong>Test Coverage</strong></td>
<td>See “Requirements Coverage.”</td>
</tr>
<tr>
<td><strong>Test Editor</strong></td>
<td>The AST conceptual module used to enter test requirements, create test cases, group them into test collections and test sequences, and manage the test repository.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Test Engine</td>
<td>The AST conceptual module that runs test collections to generate test results.</td>
</tr>
<tr>
<td>Tester</td>
<td>An instance of the AST.</td>
</tr>
<tr>
<td>Test Log</td>
<td>An ordered list of related messages generated by AST modules during its operation; there will be several log types.</td>
</tr>
<tr>
<td>Test Outputs</td>
<td>Information generated during a test run; includes test results and logs.</td>
</tr>
<tr>
<td>Test Repository</td>
<td>The AST conceptual module that stores test requirements, test cases, test collections, and test sequences.</td>
</tr>
<tr>
<td>Test Results</td>
<td>Information generated during a test run by the test cases regarding the equipment’s conformance to test requirements.</td>
</tr>
<tr>
<td>Test Run</td>
<td>The act of executing a set of test sequences against a specific piece of equipment.</td>
</tr>
<tr>
<td>Test Run Specification</td>
<td>The information required to execute a specific test run; includes static equipment configuration information, run-specific equipment set-up information, test sequence parameters, etc.</td>
</tr>
<tr>
<td>Test Sequence</td>
<td>A set of test collections that is run by the AST to generate test outputs.</td>
</tr>
<tr>
<td>Test Utility</td>
<td>A unit of software that performs set-up/clean-up functions before/after a test case is run to ensure the required pre-conditions and post-conditions are met; does not validate conformance to any test requirements or generate test results.</td>
</tr>
<tr>
<td>Transfer (an AST entity)</td>
<td>Moving an AST entity from one instance of the tester to another, usually to share that entity with another AST user.</td>
</tr>
<tr>
<td>Transfer Container</td>
<td>Single file used to transfer related sets of AST entities from one tester to another.</td>
</tr>
<tr>
<td>UUID</td>
<td>Universally Unique Identifier; a computer-generated identifier that is guaranteed to be unique.</td>
</tr>
</tbody>
</table>

## 2 PRODUCT DESCRIPTION

### 2.1 Environment

#### 2.1.1 Users

The users of the AST will generally have a technical background in computers. They will have an understanding of the SEMI standards and other requirements that they will be testing. On the other hand, the people most interested in the actual test results may *not* have a technical background and often will not be present during a test run.
2.1.2 Other Systems

- **Test Requirements Database**
  External (to the AST) source for base set of test requirements.

- **Platform Security Sub-system**
  Provides basis for AST security features.

2.1.3 Platform

2.1.3.1 Hardware
The AST will most often be used on a conventional laptop computer. At a minimum, it will be connected to the GEM and EDA ports during a test run and therefore should support network, USB, and parallel and serial port hardware interfaces.

2.1.3.2 Software
The AST will run in a Microsoft Windows environment. Standard email and ftp utilities will be used to transfer AST entities among testers. Export/import functions may use .xml, .csv, and .pdf formats (among others); therefore, software packages that support these formats will likely be present.

2.2 Capabilities
This section summarizes the major functions that the AST shall perform. Unless otherwise stated, each capability is required in the initial release of the AST.

2.2.1 External Interfaces

2.2.1.1 User Interface

[AST-PD-RQ-00001] A graphical user interface (GUI) shall be used for all interaction with AST users. The GUI shall use multiple active windows. The layout of the user interface is left to the tester software supplier.

[AST-PD-RQ-00002] The user interface shall not rely exclusively on color to convey results. (Example issue: yellow disappears under typical yellow litho area illumination.)

[AST-PD-RQ-00003] All user interactions with the AST shall be governed by an explicit security model that verifies that the user has the appropriate privilege to access the requested service (see Section 2.3.6, Safety and Security).

[AST-PD-RQ-00004] The user interface shall provide copy, cut, and paste functions between fields and records during the editing process for various AST entities.

[AST-PD-RQ-00005] The tester should include “wizards” for the most common use cases to guide the user efficiently through these processes. (optional)

[AST-PD-RQ-00006] These wizards should provide default settings for configuration parameters where appropriate. (optional)
2.2.1.2 Hardware Interfaces

[AST-PD-RQ-00007] The system shall interface with an E84 emulator (such as the GetControl, Inc. product) using a USB port or a parallel port.

[AST-PD-RQ-00008] The system shall have at least one network connection to be used for the HSMS and EDA connections.

[AST-PD-RQ-00009] The system should provide a way to expand the number of physical network connections. (optional)

[AST-PD-RQ-00010] The system hardware should be suitable for use in a Class 1 cleanroom environment. (optional)

2.2.1.3 Communications Interfaces

The tester shall have the capability to use the following protocols:

[AST-PD-RQ-00011] SECS/GEM over HSMS. (multiple sessions)


[AST-PD-RQ-00013] XML with SOAP.

[AST-PD-RQ-00014] The system shall be extendable to communicate using other interfaces and protocols. Such extension shall be done in a modular fashion such that the current capabilities are not affected and no redesign is required.

2.2.2 Specific Product Features

2.2.2.1 Enter Test Requirements

[AST-PD-RQ-00015] The system shall allow users to enter information about test requirements. This includes (at a minimum) a unique requirement ID, name, type, description, document source/version, communication protocol, message structure, and expected result.

[AST-PD-RQ-00016] The system shall allow entry of multiple user-assigned fields, such as category, comments, version information, etc.

[AST-PD-RQ-00017] The system should allow the user to view various sets of requirements using ranges of values for some of the attributes as selection criteria. For example, this would be useful in searching for the latest sets of requirements that were in place at a given time. (optional)

[AST-PD-RQ-00018] An import/export feature for test requirements information should be provided. (optional)

[AST-PD-RQ-00019] The system shall support flexible partitioning of requirements.

[AST-PD-RQ-00020] The system shall support locking of sets of requirements by partition.

[AST-PD-RQ-00021] The system shall support marking sets of requirements as private by partition. This will prevent them from being viewed, copied, or exported.

The relationship of various types of source documents and the test requirements stored in the AST is shown in Figure 3.
2.2.2.2 Develop Test Cases

[AST-PD-RQ-00022] The system shall enable test developers to write test cases that communicate with the equipment to validate specific test requirements.

[AST-PD-RQ-00023] Test cases shall support a full range of computational and communications capability, including arithmetic calculations, conditional logic, looping, branching, waiting for external events, multi-threading, operator interaction, equipment communication, database access, exception handling, etc.

[AST-PD-RQ-00024] The system shall support multiple active test cases that run asynchronously and concurrently. These may in fact be separate instances of the same test case.

[AST-PD-RQ-00025] Test cases shall support a parameter passing mechanism (input and output).

[AST-PD-RQ-00026] Test cases shall be able to communicate with equipment operators through the tester user interface to prompt them to perform the manual actions that support the test case.

[AST-PD-RQ-00027] The system shall provide an explicit mechanism for describing the pre-conditions that must be satisfied before a test case can be executed and the post-conditions that will exist after a test case has executed.

[AST-PD-RQ-00028] The system shall provide a mechanism for comparing pre-condition (and post-condition) descriptions for equivalence.

[AST-PD-RQ-00029] Users shall be able to create, name, save, open, copy, rename, delete, and manage sets of test cases in ways that are familiar to users of Windows-based applications.
The user shall be given the option to add comments when a test case is changed and saved.

The system shall generate a universally unique identifier (UUID) for each test case.

Any time a test case is changed, a new UUID shall be generated. The case shall record the predecessor UUIDs for the three most recent changes of a test case.

Test cases shall have a user-assigned version name.

Different versions of a test case shall have a different UUID generated for them.

The system shall provide a comment field or other mechanism for capturing the difference between distinct versions of a test case.

The system shall allow a user to lock a test case to prevent further change.

A locked test case may only be unlocked by the user that locked it or by a user with AST system administration privileges.

The system shall allow locked test cases to be copied, renamed, and stored as a new, unlocked test case with a new UUID. This enables test developers to build on others’ work while still preserving the integrity of the original test case.

When a test case is copied, the linkage references to the test requirements, document origin information, and author field for that test case shall be cleared, and the current user listed as the new author.

Test cases shall generate test results that indicate the degree of conformance the equipment has achieved with respect to the covered test requirements (or portion thereof).

Test cases shall generate output to various logs to allow users to monitor the activity both during and after a test run.

The system shall allow a user to mark a test case as “private.” This allows the test case to be included in test collections and executed but prevents its “source code” from being viewed, edited, copied, or exported. This is intended to protect the intellectual property of the test case developer, not to limit an AST user from seeing all the message traffic, logs, results, etc. that are required for the effective use of that test case.

The system shall support a mechanism for invoking “plug-ins” (e.g., custom DLLs) in a test case. This would be useful in situations where a complex test requirement or a specific user extension required a custom piece of software to test it. This might also be the most convenient approach for verifying compliance to structural standards (e.g., an E120 CEM node schema) and otherwise extending the AST’s capability beyond communications-oriented test requirements.

The system shall allow the entry of descriptive information about a test case that will be helpful to other users (test developers, test engineers, test operators, etc.).

This descriptive information shall include at a minimum a text description of the test case’s behavior, authorship, version information, the pre-conditions that must be satisfied before it can execute, the message sequence it will send and expects to receive, the
range of return codes it may produce and the reason for each, and the post-conditions that will exist after its execution.

Figure 4  Data Model of Key AST Test Entities

[AST-PD-RQ-00046]  A field for free-form user comments shall be provided. A typical use of this field would be documenting the changes in capability and coverage from version to version.

[AST-PD-RQ-00047]  The system shall allow the user to display this information on request.

2.2.2.3  Link Test Cases to Requirements

The capabilities described in this section are key to achieving the AST’s requirements traceability objective.

[AST-PD-RQ-00048]  The following relationships between test requirements and test cases shall be supported: one-to-one, one-to-many, and many-to-one. In other words, a test requirement may be validated by a single test case, or it may require multiple test cases. Conversely, a test case may validate multiple test requirements, and single test requirement, or only a portion of one.
There may be more than one correct way to validate a test requirement, the linkage mechanism between test cases and test requirements shall support this situation.

The system shall allow the user to view relationships between test requirements and test cases in useful ways, except in a manner that exposes test requirements marked as private. Examples include showing all test cases that validate a given test requirement, showing all test requirements that are validated by a given test case, showing all test cases in a given category, etc.

For a given category (e.g., Data Collection Management tests) and version (e.g., March 2005) of test requirements, the system should be able to indicate which test cases provide the best test coverage. This may take several steps, in which the user provides various sets of filtering criteria to tune the results of the search.

The system shall provide a “backwards compatible” field for denoting whether or not a test case is sufficient for use with previous versions of test requirements. This situation can arise when a standard has been extended without changing the behavior within its earlier scope and when changes to a standard are primarily editorial.

2.2.2.4 Develop Test Utilities

The system shall allow a test developer to write test utilities that communicate with the equipment to set up the pre-conditions for a test case, to clean up after a test case executes, and similar functions.

All the required capabilities for creating and managing test cases pertain to test utilities, except that test utilities do not generate test results. However, there shall be some way to know whether or not test utilities complete successfully (e.g., to know if the pre-conditions have been set up correctly).

Failure of a test utility to execute properly shall be noted in the test log(s).

There shall be an easy method to make a test utility from a pre-existing test case or collection.

2.2.2.5 Create Test Collections

The system shall enable the user to assemble test cases and test utilities into test collections.

Test collections shall refer to test cases/utilities by name and version name.

The system shall support use of wildcard characters in the version name, allowing the resolution to a specific test case/utility to be postponed until a test run is executed.

The system shall support creation of test collections comprised solely of test utilities. It is expected that AST users will construct such test collections if a large set of interactions with the equipment is required to put it in a specific state before addressing a particular test requirement. Moreover, many of these “utility collections” may be almost identical to their counterpart test collections that report test results for the same set of set-up interactions.

Test collections shall support a parameter passing mechanism (input and output).
[AST-PD-RQ-00062] The system should provide browsing utilities that help select the latest test cases for a given revision level of the underlying test requirements. See Figure 5 for an illustration of how traceability of test requirements should flow all the way to test collections.

![Figure 5 Requirements Traceability](image)

[AST-PD-RQ-00063] The system shall allow test collections to include other test collections (i.e., test collections may be nested).

[AST-PD-RQ-00064] The ordering of test cases, test utilities, and test collections is significant, as this is the order in which they shall be executed when included in a test sequence.

[AST-PD-RQ-00065] The system shall allow a user to mark a test collection as “private.” This allows the test collection to be included in other test collections and executed but prevents its “source code” from being viewed, edited, copied, or exported. This is intended to protect the intellectual property of the test collection developer, not to limit an AST user from seeing all the message traffic, logs, results, etc. that are required for the effective use of that test case.

[AST-PD-RQ-00066] The system shall provide a flexible “tree-view” capability for viewing and editing test collections. This shall enable the user to expand or collapse nested entries in the collection to better understand the overall flow.

[AST-PD-RQ-00067] In addition to the logic that is contained in the test cases and test utilities that form the basis for a test collection, the user shall be able to add conditional logic within a test collection. The implementation approach for this logic is left to the AST supplier, but may take the form of a scripting language.
Logic at the test collection level shall be able to support conditional execution of test cases, branching within the test collection, and looping around sets of test cases, depending on the value of run-specific parameters or results of previous test cases and test collections. This feature is intended to support more realistic emulation of an actual production environment, performance and life testing, and ease of continuation of a test run if a specific test case fails.

The system shall support communications with the test operator within a test collection (in addition to what may be generated by the included test cases and test utilities).

The system shall allow test collections to be created, named, saved, edited, copied, renamed, deleted, and managed in familiar ways.

The user shall be given the option to add comments when a test collection is changed and saved.

The system shall generate a UUID for each test collection. If a test collection is changed, a new UUID shall be generated. The collection shall record the predecessor UUIDs for the three most recent changes of a test collection.

Test collections shall also have a user-assigned version name.

Different versions test collections shall have a different UUID generated for them.

The system shall provide a comment field or other mechanism for capturing the difference between distinct versions of a test collection.

The system shall enable a user to lock a test collection.

The system shall allow locked test collections to be copied, renamed, unlocked, and stored with a new UUID.

The system shall enable the user to check the test collection for validity during the assembly process. This shall include checking to ensure the order of test cases and test utilities satisfies their specified pre-conditions, applying any precedence rules that may be stated in the test requirements, and so on.

Create Test Sequences

The system shall enable the user to assemble one or more test collections into a test sequence for execution in a test run.

The system shall provide a mechanism for calculating the test requirements coverage for a defined test collection or test sequence.

The system shall be able to calculate and express coverage with respect to the various elements in the requirements traceability scheme.

The order of test collections in a test sequence is significant only in that this is the order of execution in a test run if all the collections in the test sequence are selected for execution.
In a test sequence, it shall be possible to run multiple test collections simultaneously, as this better emulates the real production fab environment than a single carrier/process job test.

In a test sequence, it shall be possible to synchronize the execution of multiple test collections.

The system shall allow test sequences to be created, named, saved, edited, copied, renamed, deleted, and managed in familiar ways.

Test sequences shall also have a user-assigned version name.

The system shall allow a user to mark a test sequence as “private.” This allows the test sequence to be included in test collections and executed but prevents its “source code” from being viewed, edited, copied, or exported. This is intended to protect the intellectual property of the test sequence developer, not to limit an AST user from seeing all the message traffic, logs, results, etc. that are required for the effective use of that test case.

Different versions of test sequences shall have a different UUID generated for them.

The system shall provide a comment field or other mechanism for capturing the difference between distinct versions of a test sequence.

The user shall be given the option to add comments when a test sequence is changed and saved.

The system shall generate a UUID for each test sequence. If a test sequence is changed, a new UUID shall be generated. The sequence shall record the predecessor UUIDs for the three most recent changes of a test sequence.

The system shall provide a flexible “tree-view” capability for viewing and editing test sequences. This shall enable to user to expand or collapse nested entries in the collection to better understand the overall flow.

The system shall enable a user to lock a test sequence.

The system shall allow locked test collections to be copied, renamed, unlocked, and stored with a new UUID.

2.2.2.7 Enter Test Run Specifications

The system shall allow a user to enter the information necessary to run a test sequence against a particular piece of equipment.

A test run specification shall consist of (but is not limited to) the following:

Static information about the configuration of the equipment being tested.

Run-specific test sequence parameters that affect the behavior of the test cases that will be executed and the conditional logic in the test collections.

Other configuration parameters necessary to establish a safe, secure connection between the tester and the equipment.
[AST-PD-RQ-00101] The system shall allow the user to copy related subsets of run spec information to speed the configuration process for similar runs.

[AST-PD-RQ-00102] The test run information (inputs and outputs) associated with a particular test run shall be locked and assigned a UUID. This information shall be edited only by users with the proper authorization.

[AST-PD-RQ-00103] The system shall be required to test only one piece of equipment at a time. However, an approach for extending the system to support multiple, simultaneous test runs should be considered in its initial design. This may be useful in testing common equipment combinations, such as linked litho (track + stepper).

2.2.2.8 Run Test Sequences

[AST-PD-RQ-00104] The system shall enable users to select and run test sequences (see Figure 6).

[AST-PD-RQ-00105] If the run spec information required for running a selected test sequence is incomplete or invalid, the system shall identify which areas need additional information and enable the user to provide it.

[AST-PD-RQ-00106] The system should provide a mechanism for “single stepping” through a test sequence or setting “breakpoints” at specific places in a test sequence. This is useful for debugging test cases as well as giving a test operator a way to ensure that all pre-conditions are met before a test case begins to run. (optional)

[AST-PD-RQ-00107] The system shall provide a test run status window that shows the structure of the test sequence, the overall progress within the sequence, and the status of all active tests.

[AST-PD-RQ-00108] The system shall have the option of tracking metrics during a test run. These metrics shall include the average number of trace parameters/second, the average number of messages per second, the average number of parameters per message, and the maximum for each of the metrics that is recorded during the test run.

[AST-PD-RQ-00109] It shall be possible to start and stop the metric tracking.

[AST-PD-RQ-00110] If a test run is interrupted for some easily correctable reason (e.g., a simple pre-condition for one of the test cases was not satisfied), the user shall be able to resume the test run as close to the point of interruption as possible.

[AST-PD-RQ-00111] The system shall prompt the user to enter reasons for test run interruption and actions taken by the test operator.

[AST-PD-RQ-00112] Redoing a test run at a later time shall require less than 15 minutes of AST configuration and set-up. This situation will exist when a test run has failed and the customer wants to try again or when a new revision of the equipment’s control or interface software must be validated for conformance to the same set of test requirements.

[AST-PD-RQ-00113] The user should be able to add comments to the test logs during a test run.
2.2.2.9 Generate and View Test Logs

[AST-PD-RQ-00114] The system shall generate logs during test runs.

[AST-PD-RQ-00115] The system shall support multiple logging levels.

[AST-PD-RQ-00116] The system shall timestamp all log entries to a resolution of 0.01 seconds.

[AST-PD-RQ-00117] One specific type of log shall include the messages sent between the AST and the equipment.

[AST-PD-RQ-00118] Log messages from different EDA sessions and sources shall not be mixed in the same log.

[AST-PD-RQ-00119] Other kinds of logs shall include at a minimum 1) a test run progress log and 2) a debug trace with variable verbosity levels.

[AST-PD-RQ-00120] The system should support a mechanism for generating user-defined logs. (optional)

[AST-PD-RQ-00121] The system shall be able to display each log in a separate window that the user can scroll, with the most recent log entry shown at the bottom of the window.

[AST-PD-RQ-00122] If the user chooses to display an active log and the cursor is on the bottom position, the entries shall scroll up automatically as new entries are received.

[AST-PD-RQ-00123] Users shall be able to select and view different logs in a single window to understand the interaction of various test cases.

[AST-PD-RQ-00124] Entries in a log shall be viewed in time sequence order; the system should also provide display options that enable log entries to be viewed in order of other user-selectable fields (e.g., test case category, UUID, EDA session ID, data collection plan ID, source, fully qualified naming scheme, etc.).

[AST-PD-RQ-00125] The system shall be able to filter the information shown on a log view on certain user-selectable fields (e.g., suppress/display debug trace).
Logs for a test run shall be locked, and may only be annotated with proper authorization.

### 2.2.2.10 View/Analyze Test Results

The system shall be able to quantify the conformance (i.e., pass/fail score) and coverage (i.e., portion of the test requirements actually tested) of test results in terms of the original source documents (at one end of the spectrum) and the test sequence requirements (at the other end).

The system shall provide an easy-to-use mechanism for comparing the test results of otherwise identical test runs.

The system shall be able to relate test results to all elements of the traceability scheme (i.e., from test sequence all the way back through test requirements to the original source documents). For example, separate pass/fail statistics shall be available for test requirements derived from the SEMI standards (as individual source documents, or the complete set of standards under test), the ISMI consensus set of requirements, an individual ISMI member company’s automation requirements document, the combination of all these, and so on.

The system shall provide a fully qualified naming scheme for instances of test cases in a test sequence so the test results can easily be correlated to the specific test case that generated them.

The user shall be able to view portions of test results and test logs for individual test cases and test collections.

In addition to viewing the logs, the user shall be able to access other portions of the test results as they are being created (i.e., during a test run). This will help a test operator/engineer understand how the test is progressing to determine whether or not to interrupt or stop it.

### 2.2.2.11 Generate Test Reports

The system shall generate built-in reports to convey the scope, objectives, configuration, and results of a test run.

The reports shall provide various levels of detail depending on the intended audience.

The content of test reports shall be configurable by the user.

Test results shall be stored in such a way that allows for flexible report generation. For example, if a test case is executed multiple times because the logic in the test collection dictated this behavior, the test results analysis should be able to count this as a single instance or multiple instances and calculate the proper percentage compliance accordingly.

The system shall generate an.xls file as one of the report output options to enable the user to easily view results using Microsoft Excel.

The system shall support XML as one of the report format options to enable them to be easily parsed by other applications.
2.2.2.12 Transfer AST Entities

[AST-PD-RQ-00139] The system shall enable the user to transfer various AST entities (test cases, test sequences, test results, run information, etc.) among instances of the AST for the purpose of sharing them with other users.

[AST-PD-RQ-00140] The state of AST entities (e.g., locked vs. unlocked, unique identification, all other information) shall not be affected by the transfer process.

[AST-PD-RQ-00141] Related sets of entities shall be assembled by the system into a single transfer container.

[AST-PD-RQ-00142] Normal e-mail, ftp, and other file-sharing technologies shall be used for sending transfer containers from one system to another.

2.2.2.13 Export/Import AST Entities

[AST-PD-RQ-00143] The system shall provide export/import capabilities for the subset of AST entities that could be used by popular office software and analysis packages.

[AST-PD-RQ-00144] Specific entities for which export capability shall exist include (as a minimum) test results, test run specifications, test sequence structure (i.e., the titles and ID numbers of test collections and test cases included).

[AST-PD-RQ-00145] The default directories into which various kinds of AST entities will be stored when exported should be user configurable. (optional)

[AST-PD-RQ-00146] Specific entities for which import capability shall exist include test run specifications (especially the static equipment configuration information) and test requirements lists.

2.2.2.14 System Installation and Configuration

[AST-PD-RQ-00147] The tester shall be installed from a single CD-ROM or downloadable installation file.

[AST-PD-RQ-00148] The system installation process shall be automated, prompting the user for essential directory and configuration information only.

[AST-PD-RQ-00149] Other AST entities containing significant intellectual property (test requirements lists, test cases/utilities/collections/sequences) shall also be installed using similar mechanisms.

2.2.2.15 System Administration

[AST-PD-RQ-00150] The system shall provide utilities for managing, archiving and restoring test run information (inputs and outputs). In an equipment supplier context, this will be useful in handling the volume of test information generated for each piece of equipment that is shipped. In a semiconductor fab context, this is necessary because of the variety and volume of tests that are performed.

2.2.2.16 Other Features

[AST-PD-RQ-00151] The user shall be able to send individual messages to the equipment.

[AST-PD-RQ-00152] The user shall be able to run individual test cases.
2.2.3 Priority of Standards Categories to be Tested

[AST-PD-RQ-00153] The first priority for the use of the AST is to support the industry’s testing needs for the EDA suite of SEMI standards. These include E120, E125, E132, and E134 and all the .1 binding standards. However, the creation of this specific test content is not considered part of the AST itself.

[AST-PD-RQ-00154] The second priority for the AST is support for the GEM 300 suite of standards, including the E84 material handoff protocol. This set consists of E5, E30, E37.1, E39, E40, E87, E90, and E94. It does not include any of the specific equipment models (SEMs) or E95 (SEMI Specification for Human-Computer Interface).

2.3 Attributes

2.3.1 Performance

This section summarizes the speed, size, and bandwidth limitations of the tester.

[AST-PD-RQ-00155] The tester shall support simultaneous use of all the communication mechanisms that may be active during a complex test of the equipment’s automation requirements. For example, it shall support concurrent use of GEM, EDA, and multiple E84 connections that would be representative of a complex multi-chamber cluster equipment running five process jobs from three front opening unified pods (FOUPs) and supporting an on-line fault detection application.

[AST-PD-RQ-00156] The tester shall support a 2 GB maximum message size.

[AST-PD-RQ-00157] The tester shall support 10 simultaneous EDA sessions.

[AST-PD-RQ-00158] The tester shall support an average EDA data collection rate of 20K data points per second.

2.3.2 Ease of Use

2.3.2.1 User Interface

[AST-PD-RQ-00159] The user interface shall be intuitive such that a test engineer familiar with popular Windows-based applications, SEMI standards, and conformance testing can effectively use the tester after two hours of training.

2.3.2.2 Training and Documentation

[AST-PD-RQ-00160] Training material shall be provided that enables each of the major AST user types to perform the basics of his/her role after no more than 4 hours of guided instruction.

[AST-PD-RQ-00161] User documentation shall be provided that summarizes the installation and use of the AST. The target audience for this documentation is test engineers and test operators.

[AST-PD-RQ-00162] Documentation for the test developer shall include sample test cases for some portion of each of the SEMI standards (expressed in test requirements terms) to be supported by the AST.

[AST-PD-RQ-00163] Other documentation typically expected for a standard software product shall be provided (quick-start guides, release notes, etc.).
2.3.3 **Backwards Compatibility**

[AST-PD-RQ-00164] A given release of the AST shall be able to execute (or convert and execute) test sequences created on earlier releases. Test cases generated on a given release of the tester are not expected to run on previous versions.

[AST-PD-RQ-00165] A given release of the tester shall be able to view and analyze results generated by earlier releases.

2.3.4 **Flexibility**

[AST-PD-RQ-00166] The tester shall be able to support new versions of the existing suite of SEMI standards without modification.

[AST-PD-RQ-00167] Within the scope of its intended use (i.e., testing equipment automation capabilities), the tester should accommodate the addition of new features and functions without extensive redesign.

[AST-PD-RQ-00168] The tester should make some provision for supporting application-level integration standards (e.g., “Interface B”) in the future.

2.3.5 **Reliability, Availability, and Fault Tolerance**

[AST-PD-RQ-00169] The tester shall crash no more than once every 2 weeks of continuous use.

[AST-PD-RQ-00170] The tester should provide some sort of “warm start” that takes no more than 2 minutes to resume a test run after a crash. (optional)

[AST-PD-RQ-00171] The tester should crash the operating system no more than once every 2 months of continuous use. (optional)

2.3.6 **Safety and Security**

[AST-PD-RQ-00172] The tester shall base its user security model on the underlying Microsoft security capabilities.

[AST-PD-RQ-00173] The system shall implement the “principal, role, and privilege” model for assigning the authorization levels to different user types that are appropriate for their respective AST activities (see Figure 7).

![Diagram](Image)

**Figure 7** User Security Model
The system shall define multiple privilege levels for discriminating among the authorizations to enter and protect test requirements; create manage test plans/collections; define and execute test runs; analyze and lock test results; administer system privileges, and so on. The specific assignment of privileges to AST services will be defined later.

The system administrator’s privilege shall not imply inclusion of all other privileges. For example, a system administrator would not necessarily be allowed to unlock and change test case content that was delivered by an outside supplier.

The tester shall provide a mechanism to ensure that the equipment is in a testable state.

The tester shall be able to protect equipment-specific data from unauthorized access. This may include customer-specific test cases/collections/sequences, test run specifications, and test results.

The tester shall provide a way to “wipe the slate” after a test run is complete and all relevant information is saved. This is important when the equipment under test is in a secure environment, and the equipment owner requires that the tester enter/exit this environment in a clean state.

Maintainability

The tester should provide debug mechanisms that enable its supplier to diagnose and correct problems effectively. (optional)

Software Portability

The tester shall run on the following Microsoft operating systems: Windows 2000 and XP/Professional. It does not need to be portable to a non-Windows environment.

Design Constraints

In addition to the previous sections, this section contains other items that will influence the AST architecture and detailed design.

Cost

The AST hardware and software platform shall cost no more than a high-end laptop computer configured for portable use.

Transportability

The AST shall be capable of running on a widely available laptop computer. All the additional hardware required to connect an AST to the equipment shall fit in an average briefcase.

Use of Additional Third-Party Software

The system shall not require the installation or licensing of additional third-party software other than the Windows operating system and related built-in utilities. For example, tester does not have to directly incorporate a user-specified external database.
2.4.4  Multiple User Support

[AST-PD-RQ-00184] The system should be capable of supporting multiple users (one at a time) on a single platform while preserving the integrity and privacy of each user’s AST entities.

2.4.5  Co-existence With Other Software

[AST-PD-RQ-00185] The system shall be able to co-exist with other software that is normally used on general-purpose laptop computers. In other words, the AST must not require exclusive use of the computer’s resources.

3  REFERENCES AND RELATED MATERIAL


International SEMATECH Manufacturing Initiative
Technology Transfer
2706 Montopolis Drive
Austin, TX 78741

http://ismi.sematech.org
e-mail: info@sematech.org