# Software Tools: A Key Component in the Successful Implementation of the ATML Standards

Ron Taylor
Summit Test Solutions
4266 Linda Vista Dr.
Fallbrook, CA 92028 USA
ron.taylor@summittests.com

*Abstract* – **This paper examines the IEEE Automatic Test Markup Language (ATML) family of standards and some of the impediments which must be overcome to successfully implement these standards. The paper specifically focuses on how software tools can help alleviate these issues and increase the benefits of using these new standards in Automatic Test System (ATS) related applications. The ATML standards provide a common exchange format for test data adhering to the Extensible Markup Language (XML) standard. ATML promises to provide interoperability between tools and multiple test platforms through the standardization of common test related data. The ATML standards have now been published through the IEEE SCC20 committee and are beginning to exhibit considerable interest in the ATS community and are now a requirement on some new DoD ATS programs. Three aspects of ATML related tools shall be discussed: 1) ATML Development tools which assist in the generation of ATML compliant instance files, 2) New ATS related tools which use ATML data in their applications, 3) The upgrade of existing ATS tools to utilize ATML Data. This paper also examines the work in progress of a Small Business Innovative Research (SBIR) NAVAIR sponsored program to develop ATML and test diagram tools. Utilizing ATML standards without the benefit of tools can be a labor-intensive, error-prone process, and requires an intimate knowledge of the ATML and XML standards. Employing the ATML standards on ATS programs promises to significantly reduce costs and schedule; the use of software tools are a key component in the success of these implementations and will help promote the use of ATML throughout the test industry.**

## I. INTRODUCTION

With the ATML standards now completing their IEEE standard trial use period they are beginning to gain more interest and acceptance in the ATS industry. However there are certain factors that are inhibiting their widespread use and this paper will explore some of these factors. This paper also delves into the use of ATML software tools that show promise to simplify the implementation of ATML and help alleviate some of these impediments thereby increasing the use and benefits of these standards.

This paper will also touch on an ATML related NAVAIR SBIR which is at the beginning of Phase II and is targeted at one specific implementation of ATML. The paper will indentify some of the lessons learned from the Phase I study as related to ATML tools. It has often been noted within the ATML working group that cost effective ATML implementations require ATML development tools. This paper hopes to encourage further development and use of these ATML related tools.

## II. ATML STANDARDS

The ATML standardization effort began in 2002 with a meeting of DoD and commercial ATS company representatives with an interest in the standardization of test data within the ATS industry. The group came together with the goal of defining a common format for exchanging test related data. What began as an independent focus group later joined the IEEE SCC20 committee where the ATML standards have now been published. The ATML schemas, which define the structure and format for test data, are based on the XML standard. The ATML working group set out to define the data formats for each of eight ATS related areas Test Description, Instrument Description, Test Station Description, Test Adapter Description, UUT Description, Test Configuration, Test Results and Diagnostics. Two of the these original ATML components were incorporated into existing IEEE standards: Diagnostics has been published in the AI-Estate IEEE Std 1232, and Test Results has been incorporated into the SIMICA IEEE Std 1636.1. The rest have been published as ATML IEEE 1671 component standards. The development of the ATML standards has been a cooperative effort of many different ATS related companies and DoD representatives. The ATML standards have been proven through reviews by numerous companies, three ATML demonstration efforts presented at Autotestcons and the incorporation of the standards on some real world programs. The ATML standards are now completing their IEEE trial use period and are in the update process and will soon be in ballot for IEEE full use standards. ATML promises to provide significant saving when employed on ATS programs however at this point the standards have had limited use in the ATS industry as a whole. There are certain impediments to successful ATML implementations and this will be discussed below along with

suggestions for ATML related tools that can help overcome these challenges. Test Results, as mentioned above, was originally an ATML component and is now an IEEE SIMICA standard however Test Results is included in this paper as it is considered an ATML related standard and has been included within the ATML Autotestcon demonstrations.

## III. IMPEDIMENTS TO IMPLEMENTING ATML

**Limited availability of ATML instance files** – ATML instance files are the ATML files that are populated with data according to the appropriate ATML schema. Implementing ATML is more of a task when the instance files are not available and would need to be created. As an example if there was an ATML application that would compare capabilities of different instruments yet the instruments did not have instance files to support them it would be a significant task to generate the instance files for all instruments to be compared. ATML is a relatively new standard and the number of ATML instance files to support ATS components is limited. The requirement of ATML for some new DoD programs should increase the availability of instance files. As the standards become more popular and the ATML tools become more prevalent, the number of ATML instance files available will increase.

**Lack of ATML tools commercially available** – The lack of ATML development tools commercially available is an issue that is limiting the use of ATML. However the availability of tools is increasing and certain efforts like the ATML Autotestcon demonstrations and ATML tool funding through DoD SBIR initiatives should increase the tools available in the marketplace. In addition as new programs are beginning to require ATML, the demand for tools will increase and drive their development. Without ATML tools the instance files must be generated manually which is time consuming and error-prone.

**Lack of knowledge and familiarity in industry of ATML standards**

One of the obstacles to overcome in the implementation of ATML is the lack of knowledge of the ATML standards and their associated XML schemas. The standards are extensive and the schemas each can have hundreds of different data items defined. Manually creating ATML instance files requires a knowledge of the ATML schemas and in addition a working knowledge of XML. ATML tools are beginning to meet this challenge by minimizing the need for this knowledge of ATML and XML standards when creating ATML instance files. The ATML tool can provide the user with guidance, auto-formatting and XML and ATML validation without requiring this knowledge of the user.

## IV. ATML RELATED TOOLS

The first types of ATML tools to be discussed are ATML development tools. These are tools which will guide a user in the generation of the ATML instance files (see figure 1 for a portion of an ATML Test Station instance file). The tools would typically provide entry boxes for the input of test related data. The tool would then format the data into XML instance files, according to the ATML schema of interest. There is a critical need for these types of tools as the process of generating the instance files manually is very time consuming, cumbersome and error prone. These instance file generation tools would essentially contain, and make transparent to the user, all the intricacies of the ATML standards providing the user with straight forward data input screens. The tools to generate Test Results instance files would differ and would typically be part of a test executive and would output test results in ATML format at runtime, alternatively a tool could be designed to input test results from another format and generate a test results instance file from this data.

```
< hc:Interface>
  <c:Ports>
  <c:Port name="Station_Chassis_GND1">
    <c:ConnectorPins>
      <c:ConnectorPin connectorID="SIR:J1" pinID="1A"/>
    </c:ConnectorPins>
  </c:Port>
  <c:Port name="DCPSLVA_+">
      <c:ConnectorPins>
        <c:ConnectorPin connectorID="SIR:J1" pinID="3A"/>
      </c:ConnectorPins>
  </c:Port>
  <c:Port name="DCPSLVA_-">
      <c:ConnectorPins>
        <c:ConnectorPin connectorID="SIR:J1" pinID="3B"/>
      </c:ConnectorPins>
  </c:Port>
 </hc:Interface>
 <Instruments>
  <Instrument ID="DCPS1">
    <PhysicalLocation>Crate8Slot1</PhysicalLocation>
    <Address>GPIB::26::INSTR</Address>
  </Instrument>
  <Instrument ID="DMM1">
    <PhysicalLocation>Crate160Slot4</PhysicalLocation>
    <Address>VXI0::19::INSTR</Address>
  </Instrument>
</Instruments>
```

**Figure 1, Portion of an ATML Test Station instance file**

The next type of tools to discuss are existing ATS application tools that are modified to use ATML. One of these examples is the modification of existing test executives which have been modified to output SIMICA Test Results files. The implementers have had some good success in this endeavor and have provided good lessons learned back to the SCC20 committee. ATS application tools, which previously used other data formats, should be considered for modifications to utilize ATML data. One huge benefit is that the ATML modification will allow the tool data to be in an open source format where previously it may have been in proprietary data formats. This open source would then allow other tools to use the data achieving the goal of interoperability.

The next type of tool would be new ATS application tools which are designed to use ATML data. There are many differing types of applications that could benefit from the use of ATML data. Understanding the type of data in ATML files allows one to envision new concepts of ATS applications tools which can save time and reduce errors in ATS processes. Some of these new concepts are discussed below.

Following is a brief description of each of the ATML standards and suggestions for tools which would be applicable to these standards. In addition a brief description of possible tool functionality is provided. Note that some applications will use multiple ATML file types, for example a resource manager may require ATML Test Station, ATML Test Adapter, ATML WireLists and UUT Description instance files (See Figure2).

**1) ATML Test Description IEEE-1671.1**

ATML Test Description is a tester independent description of tests to be performed on a UUT. It is essentially an electronic Test Requirements Document (TRD). All stimulus and response signals required for each test would be described in an ATML Test Description instance file.

**Tools:**

**Instance file generator** --This tool would provide a GUI for the operator to input test related data which is then stored into ATML instance files according to the applicable schema. The tool would provide XML and ATML validation. The tool should not require the knowledge of XML or the ATML standards.

**TPS Generator** -- Auto-generate a TPS from a Test Description instance file. This tool would extract stimulus and measurement signals from the Test Description instance file and generate a TPS in the target test language.

**Test Requirements Verifier** -- Compare test requirements from an ATML Test Description instance file to the capabilities from an ATML Test Station instance file to determine if the target test station is capable of testing the UUT.

**2) ATML Instrument Description IEEE-1671.2**

ATML Instrument Description holds information about an instrument. The instrument could be one physical instrument or a synthetic or composite instrument consisting of multiple instruments. The instance file will contain information about the I/O of the instrument, the instrument capabilities and accuracies of the instrument. This would be like an instrument manual in a standard electronic format. It would be highly desirable if instrument manufacturers would provide ATML Instrument Description instance files to support each of their instruments.

**Tools:**

**Instance file generator** -- (see description above)

**Instrument comparator** -- Compares the capabilities from an ATML Instrument Description instance file describing one instrument to an instance file describing another instrument. This could be useful for choosing replacement instruments for obsolete instruments

**3) ATML UUT Description IEEE-1671.3**

ATML UUT Description describes certain aspects of the UUT. The instance file would describe the UUT Interface, UUT components, wiring and any special operating requirements.

**Tools:**

**Instance file generator** -- (see description above)

**Requirements comparator** -- Ensure all pins are tested by comparing to the ATML Test Description instance file.

**Test Adapter designer** -- Use as an input for test adapter design. Generates wiring from test station to UUT pins.

**Test Diagram generator --** Generate test diagrams that show the routing of signals for each test from the test station instruments to UUT pins. This application would use data from ATML UUT Description, ATML Test Station, ATML Test Adapter and ATML WireLists instance files.

**4) ATML Test Configuration IEEE-1671.4**

ATML Test Configuration describes all components needed to test a particular UUT. The Test Configuration instance file will identify the test stations which support the UUT along with all required accessories such as Test Adapter, cables, ancillary equipment and test program software. This standard is closely aligned with the Navy Master Test Program Set Index (MTPSI) format.

**Tools;**

**Instance file generation** -- (see description above)

**Configuration Comparator** – Compares the required assets from a Test Configuration instance file to the subject test station assets from the ATML Test Station instance file.

**MTPSI file generator** -- Generate an MTPSI formatted file from Test Configuration instance file to support Navy TPSs

**Runtime asset verifier** -- Use Test Configuration instance file to compare with assets in test station to verify all assets are in the station and functional. Also inquire of operator to ensure proper interface hardware is available.

### 5) AMTL Test Adapter IEEE-1671.5

ATML Test Adapter provides information to describe TPS interface hardware: adapters, cables, load boxes etc., which are used in a TPS to test a UUT. The ATML test adapter elements describe the interface, wiring, capabilities and components.

**Tools:**

**Instance file generator** -- (see description above)

**Resource manager** -- A resource manager would determine which resources are capable of applying or measuring the required signals. This can take place at runtime or prior to runtime. The capabilities and the networkLists of the ATML Test station and ATML Test Adapter instance files would be used in this process. See Figure 2 for a possible Resource Manager design using ATML Data.

**Test Diagram generator** -- (see description above)

**Test Adapter designer** -- (see description above)

**Path Allocator** -- Use ATML Test Adapter, ATML Test Station and ATML WireLists instance files to calculate paths for test signals in TPS.

### 6) ATML Test Station IEEE-1671.6

ATML Test Station provides information to describe a test station. The interface, instruments, switching, and connectivity within a test station are found in this instance file. The ATML Test Station instance file can contain detailed signal characteristics defined using IEEE 1641 Std. signal models

**Tools:**

**Instance file generator** -- (see description above)

**Resource manager** -- (see description above)

**Test Diagram generator** -- (see description above)

**Test Adapter designer** -- (see description above)

**Path Allocator** -- (see description above)

**Requirements Verifier** -- (see description above)

**Test Station comparator** -- Compare capabilities of two different test stations by comparing their ATML Test Station instance files.

**1641 Signal Modeler** -- Tool to generate signal models compliant with IEEE 1641 Std. to use for ATML Test Station capabilities element.

### 7) Test Results, SIMICA IEEE-1636.1

Test Results, as mentioned above, is a SIMICA standard that is an ATML related standard. Test Results provides for a common formatting of all test related data from an ATE such as test limits, test names, test numbers, measured values and Probable Cause of Failure (PCOF).

**Tools:**

**Test Executive test results** -- Test executive designed to output a test results XML instance file compliant with SIMICA test results standard.

**Test results trend analysis** -- Compare different test runs using Test Results instance files to analyze trends in data. This can help identify intermittents and tests with measured values that are close to limits.

**Test Results archiving** -- Archive test results instance files for historical analysis.

### 8) WireLists, ATML IEEE-1671

The WireLists schema is part of the main IEEE-1671 standard. This schema is a system level schema to provide for the definition of connections between interfaces in different ATML instance file. For example the interface ports from an ATML Test Station instance file can be assigned to the corresponding ATML Test Adapter instance file interface ports. This schema allows for defining the complete paths from test station instruments all the way to UUT pins. All ATML instance files are stand alone so this schema was added to allow for system level connectivity.

**Tools;**

**Instance file generator** -- (see description above)

**Resource manager** -- (see description above)

**Test Diagram generator** -- (see description above)

**Test Adapter designer** -- (see description above)

**Path Allocator** – (see description above)

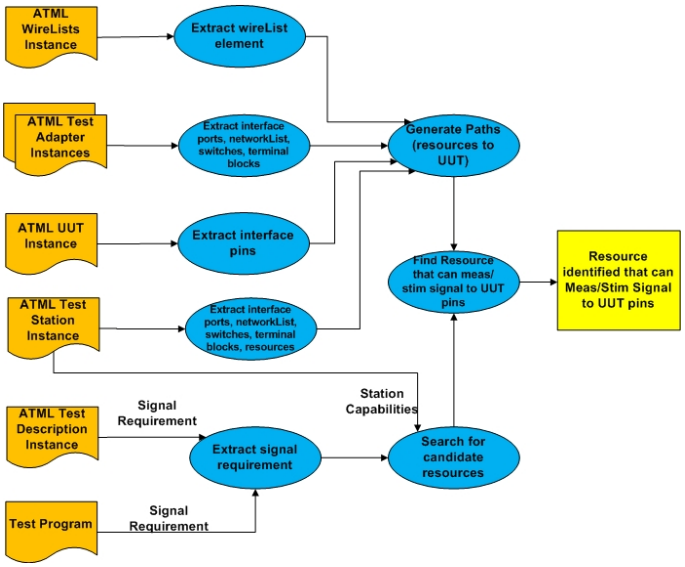


**Figure 2, Resource Manager using ATML data**

## V. IMPLEMENTING ATML

There are a number of factors to consider prior to undertaking an ATML implementation on a program. Initially an analysis should be undertaken to determine which ATML standards are possibly applicable for the target program. Figure 3 shows how all the ATML components would support a typical ATS system. From the diagram one can provide an initial assessment of which ATML files would be needed for a particular application. Additionally the diagram provides an indication as to the number of separate ATML instance files that may be required to support a particular system. This is a key bit of information since the quantity of ATML files of a certain type that would need to be generated can drive the need for ATML tools. As an example to support the test system TS1 in the diagram would only require one ATML Test Station instance file however there could be multiple ATML Test Adapter instance files, one to support each interface item: cable, test adapter and ancillary equipment. This initial assessment would make a strong case for using an ATML Test Adapter instance file tool, and the need for an ATML Test Station instance file tool is not as critical since only one file would need to be generated manually.

Inquiries have been made as to where to find ATML instance files or who should be responsible for the generation of ATML instance files. There is no firm answer in all cases, however here are some suggestions. The instrument manufacturers are the logical answer as far as who should generate ATML Instrument Description files. These would be like an electronic instrument manual and would allow users of the instrument to use the files in various applications. For new test stations it is suggested that ATML Test Station instance files would be the responsibility of the manufacturer of the test station. For the ATML Test Adapter instance files the manufacturer of the interface hardware should have the requirement to provide ATML Test Adapter instance files for each interface component: test adapter, cables, load boxes etc. The UUT manufacture should be responsible to deliver a UUT Description instance file to support all hardware produced. The company responsible for generating test requirement documents, which often is the TPS developer, should be required to provide ATML Test Description instance files to support each UUT. The Test Configuration instance file would most likely be generated by the TPS developer as they would make the determination of which assets would be required to test a particular UUT. The producer of the test executive software should be required to supply a SIMICA Test Results instance file output from the test executive. The ATML WireLists instance file should be generated by the TPS developer as they are the ones that determine how all of the interface hardware is connected to the test station and UUT. The problem arises in the implementation of ATML on existing systems. In this case the ATML instance file may be the responsibility of the in-house engineering or an outside consultant experienced in ATML however with the proper tools the knowledge of ATML is not as critical.

Another issue to consider when implementing ATML is to ensure all ATML data that will be needed for the intended use cases is populated in the instance files. ATML was developed to handle a wide variety of use cases and as a result most of the data items are optional. A procuring activity must ensure that the supplied ATML files are robust enough in the population of data to support the intended applications for the life-cycle of the procured item.

Issues to consider prior to embarking on an ATML implementation journey:

1) Which ATML standards are relevant to our program?
2) What are the ATML applications foreseen for our program?
3) Which tools would help the process?
4) Are there existing ATML tools available to fulfill our intended needs?
5) Are there existing tools that can be modified to utilize ATML data which are needed?
6) Are there tools that our company should create for the intended task? If so do we have the expertise in-house or should outside consultants be considered to develop the tools?

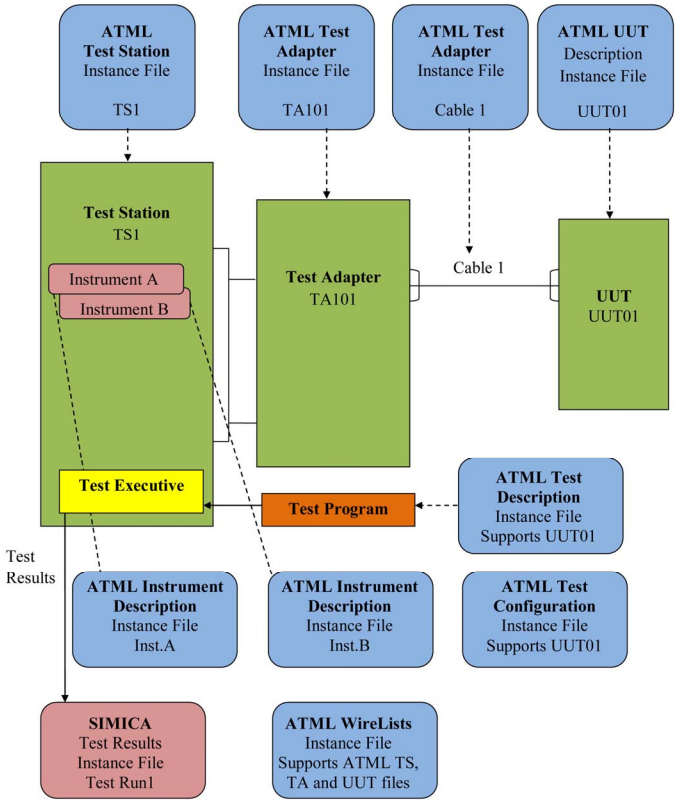


**Figure 3, ATML Instance File Diagram**

## VI. NAVY SBIR PROGRAM

Summit Test Solutions was awarded a Phase I Small Business Innovative Research (SBIR) contract in 2010 by NAVAIR to investigate the feasibility of developing automated Test Diagram tools which use ATML data**.** The target platform for this SBIR is the Navy Consolidated Automated Support System (CASS) test station however applicability to other DoD test stations is being considered. Two of the tools to be developed in this SBIR provide for the generation of ATML instance files, one for the ATML Test Station and the other for the ATML Test Adapter standards. In addition the SBIR effort will provide a tool to automatically generate test diagrams to support the CASS station TPSs. These test diagrams will show the routing of signals from test station instruments to UUT pins for each test. The concepts to be employed will use the ATML data from the ATML Test Adapter, ATML Test Station and ATML WireLists instance files and generate the active paths from this data. The Phase I findings lead to a Phase II development program which we are in the initial phase of performing. While much of the SBIR work is of a proprietary nature, we will discuss some issues related to ATML tools which came out of the SBIR work. This SBIR required the generation of a number of ATML instance files, one to describe the CASS test station using the ATML Test Station standard in addition ATML Test Adapter instance files are required to describe the interface hardware for the target CASS TPS. Our initial instance file work was in the generation of the ATML Test Station instance file to describe the CASS Hybrid test station. Below are some lessons learned from the generation of this instance file.

1) Consider tool functionality that would validate names when the same name is used to define different elements. As an example the Interface Port names may also be used in the NetworkList and a tool should verify the names match or identify inconsistencies.
2) A typical ATML schema could have hundreds of different data items defined, methods to sort through the data and possibly categorize should be considered.
3) ATML tools should not require the knowledge of ATML or XML standards to input data for instance file creation.
4) Tools should identify to the user which data items are required according to the ATML schemas and which are optional.

## VII. ATML STANDARDS REFERENCE MATERIAL

The ATML standards are available for purchase from the IEEE at IEEE.org. The ATML schemas, which is the crux of the standards, are available for download free of charge at http://grouper.ieee.org/groups/scc20/ATML/. An informative ATML website is available through the SCC20 at http://grouper.ieee.org/groups/scc20/tii/welcome.htm which was created to help ATML users and software developers learn more about ATML For information on participating in the ATML within the SCC20 see http://grouper.ieee.org/groups/scc20.

## VIII. SUMMARY

In this paper, we have examined the new ATML family of standards which promise to reap cost saving benefits in the ATS industry. We have identified some impediments to the implementation of the standards. The paper has identified ATML tools as a key element to successful cost-effective ATML implementations. We have discussed the two main types of ATML tools, ATML instance file generation tools and ATS application tools which utilize ATML data. The NAVAIR SBIR to generate ATML tools was also discussed.

ATML promises significant cost and schedule savings on existing and future programs through the standardization of test data. The use of these standards should increase in the ATS industry as ATML tools become more available thereby yielding significant savings throughout the test industry.