ACHIEVING ROBUST INTERCHANGEABILITY OF TEST ASSETS IN ATE SYSTEMS

Roger P. Oblad
Santa Rosa Systems Division
Hewlett-Packard Co.
1400 Fountaingrove Parkway
Santa Rosa, CA 95403 USA
(707) 577-3466
oblad@sr.hp.com

Abstract - This paper identifies the key issues that have made it so difficult to achieve asset interchangeability. Several of the historical attempts to solve the problem of asset interchangeability are described, along with an analysis of the reasons that they did not achieve the expected results. Specific topics that are covered are SCPI, VXIplug&play, IVI, ATLAS, and Measurement Subsystems. Principles associated with the ownership of interfaces will be outlined. Finally, a set of rules and principles will be discussed that must be applied to achieve "robust" asset interchangeability. "Robust" is defined as interchangeability that can be "guaranteed" without testing all TPSs against the modified test system.

INTRODUCTION

For the purposes of this paper, "test asset interchangeability" is defined as the ability to replace a given hardware asset with an alternative asset of sufficient capability but of different design or manufacturer in an Automatic Test System (ATS). Software changes are limited to replacing components directly associated with the asset. Application software--or more specifically Test Program Set (TPS) software--must not require changes. After the interchange is performed it must be possible to guarantee that the same results can be obtained with the modified ATS.

The need for test asset interchangeability comes from the fact that target test devices—that is, Line Replaceable Units (LRUs) or Devices Under Test (DUTs)--and their associated test software TPSs can outlive the commercial test instrumentation employed by 10 to 20 years. This is especially true for aircraft, automobiles, trucks, radar systems, and weapons systems.

In many cases the following approaches have been used to deal with asset obsolescence issues: 1) modify TPS software, or 2) assure continued availability of test assets. There is a large investment in TPS software. Having to modify it whenever a test asset is to be replaced is very expensive and logistically almost impossible. Even if it were possible to maintain the availability of test assets, this would lock test systems into "old" assets that could not answer the demands for size reductions or other new performance requirements. It doesn't take much to cause an asset to become obsolete. This can happen due to a business decision of a vendor or it can be caused by an unresolvable technical or supply problem with a lower-level part or process.

Different approaches have been taken to achieve asset interchangeability. Each of them tends to focus on only one aspect of the problem. The problem is complex and a full solution requires that several of these approaches be used, to one extent or another, together. The solution cannot just address the technical issues. It must also address the business issues associated with the behavior of the free enterprise system and also factor in the requirements of logistics and support. As these various approaches are reviewed, it is helpful to notice at what point in the overall test system architecture the solution is being applied.

There have been several recent developments that have been focused on asset interchangeability. These include work by Hewlett-Packard Company, Lucent Technologies, National Instruments, and work at the ATS R&D IPT (ARI) for the Department of Defense (DoD).

Three years ago work began at HP to create the "Measurement Subsystems Architecture," which was first reported in AUTOTESTCON-1997[1]. At AUTOTESTCON-1998, HP Vice-President Ned Barnholt[2] offered this work to the industry as a

starting point for a new industry-standard software approach for achieving test asset interchangeability. At about the same time, Lucent Technologies independently developed an internal solution to test asset interchangeability for their internal needs, which they referred to as "Call by Name Drivers." In August 1998, National Instruments initiated the formation of the IVI Foundation[3], whose mission is to develop semantic standards for common classes of instruments. From the DoD, the ARI[4] has developed an overall ATS architecture over the last several years that has been focused on both test asset interchangeability and TPS transportability. Another concept investigated by some DoD customers is that of synthetic instruments.

Although these various efforts are largely complementary, their shared focus has resulted in confusion. In order to avoid the confusion between the HP work on Measurement Subsystems and the IVI Foundation work on driver specifications, the two activities have been combined under the sponsorship of the IVI Foundation. There are strong similarities between the ARI architecture, the HP Measurement Subsystems work, and the work done at Lucent. An effort is underway to adopt common terminology.

INTERCHANGEABILITY

The Value of Semantic Standards

Both interchangeability and interoperability are dependent upon semantic standards. Semantic interfaces are without question the most important element of any interchangeability solution. All of the interfaces in an automatic test system present their capabilities through ASCII (usually SCPI) commands, function calls, or object methods. The spelling of these commands and their associated meanings establish all of the functionality in a system.

Ownership of Interfaces

It is very useful to clearly identify the interfaces in a system and determine who the owners of these interfaces are. The client of an interface is at risk if it is owned or controlled by a person or organization other than the client. When the provider of a component or asset does not know how it is used, then the provider cannot guarantee the use. These two principles are key to understanding why test asset interchangeability has been so difficult to achieve. They are also key principles in designing a solution.

Architecting for Interchangeability

Early in the history of ATE, application programs directly interacted with instruments. When this is done, where is the measurement? Part of the measurement is in the application and part of it is in the instrument firmware. Both interoperability and interchangeability are sacrificed when this is done because a physical barrier, the I/O interface, lies in the middle of the measurement. The test asset vendor controls the asset's interface but doesn't know how it is used. The application developer is depending on the test asset interface but has no control over it.

By adding two new abstraction layers, clean roles and responsibilities are established. The details follow.

Requirements of Interchangeability

In some cases, interchangeability is not possible. This happens when an alternate candidate test asset cannot perform the required measurement or generate the required signal of the original test asset. In this case there is no solution other than finding a different test asset or set of test assets that has the necessary capabilities.

The following conditions are necessary to achieve interchangeability:

1) Alternative test assets must be capable of performing the same physical measurements, control, or stimuli required by the end application.

2) Any software that uses a test asset's API must either find the same API or be interchanged along with the test asset.

3) Any test asset-specific peculiarities of an interchanged test asset must be managed by software.

One may wonder if achieving partial interchangeability would suffice. Is 80 percent interchangeability enough? The interchangeability problem is a difficult one to solve, and going part way will certainly reduce the effort required to do an interchange. The problem is that the remaining 20 percent can take a lot longer than first anticipated. Most people really are not satisfied if the ATE can

run a TPS without a software failure. What they really need is the “same answer.” However, if the real requirement for asset interchangeability is to be able to “guarantee" the same answer without thoroughly testing the end application or all associated TPSs, then a more comprehensive solution is required.

Second Order Effects and Asset Peculiarities

There are several good examples of test assets of the same general type, but of a different model or manufacturer, that should have been able to perform the same operation of another. However, in actual use the assets do not produce the same answer to the same measurement. This can occur for a variety of reasons. The internal algorithms or methods of instruments are different. These problems affect even the simplest of instruments, such as a digital voltmeter (DVM). Following are two actual examples of difficulties encountered attempting to interchange DVMs.

In one system, the differences between the internal autorange features of two DVMs resulted in the instrument presenting different input impedances to the test circuit. This difference resulted in measurement errors that were unacceptable.

In another system, when two different DVMs were used to measure the RMS voltage of the same signal, they returned different answers. This was caused by differences in the way the internal algorithms of the DVMs interacted with noise on the measured signal.

Test assets that are considerably more complex than a DVM, such as microwave sources and spectrum analyzers, create interchangeability problems that are much more difficult.

Other differences can involve setup requirements. An example of this is using a four-channel oscilloscope in place of a two-channel oscilloscope. New decisions need to be made that could not have been made during the initial development of the ATS.

IVI-MSS, described in this paper, provides a place to put the code, which addresses these problems.

APPROACHES TO TEST ASSET INTERCHANGEABILITY

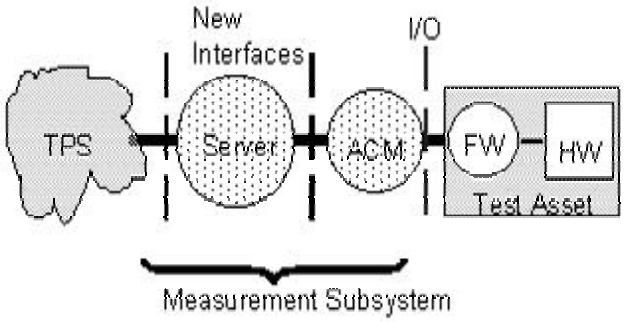

Figure 1. AutomaticTest System

There are three general approaches to test asset interchangeability (Figure 1). The left side of the figure represents the application TPS environment. A solution that applies directly to the TPS would specify the semantics of an abstract interface that TPS software must use. This is often referred to as a signal interface, and the most common implementation is the ATLAS language. The right side of the figure represents the physical test assets in a system. The solutions focused on here are those that would specify a standardized semantic interface for test assets. Examples of this are CIIL, SCPI, and IVI-Drivers. The section in the middle represents a third approach, which is to insert two new abstraction layers between the TPS environment and the test asset environment. This is the subsystem approach, which is the focus of this paper. Each of these approaches has value and the best ATS designs could well incorporate all three of them.

ATLAS

ATLAS has developed over the last 30 years. It was started by the commercial aircraft industry in the mid-1960s. Initially, it was a language of English nouns, verbs, and adjectives, which was intended to describe the specifications of avionics equipment in a standard way. The documents created using this language were originally meant for human reading only. In the mid-seventies, ATLAS made the transformation to a programming language. Software compilers were created to read ATLAS statements and use them to help develop TPSs. A basic principle of ATLAS was that programs written in it would not make any direct references to test instrumentation. This was to assure interoperability of TPS software and to create asset interchangeability. As a result, ATLAS supports no

direct way to address instrument functionality. This seems like a good thing for interoperability. However, the translation between ATLAS programs and instrument functionality has to take place somewhere in a system. Unfortunately, there has been no consistent standard on how this is done. Also, since ATLAS has not always been able to keep up with providing all the needed standard nouns, verbs, etc. in a timely fashion, TPS developers have sometimes resorted to direct calls to test assets. Using non-ATLAS modules (called NAMs) does this.

It has become common for these NAMs to talk directly to a hardware asset's first-level interface. These interfaces are owned and controlled by the test asset manufacturers, while the NAM is owned by the TPS developer. When this is done, both interoperability and interchangeability are sacrificed. The problem here is not ATLAS or the concept of a signal descriptive language. A quality ATLAS implementation fully abstracts test asset functionality. The problem is in the lack of semantic standards that NAMs could use to access ATS functionality. This has forced TPS implementers to deal directly with instruments. If these semantic standards existed, they would simplify the task of maintaining ATLAS implementations as well.

Standardized Interfaces for Test Assets

Another approach to test asset interchangeability is to try to standardize the programming interface to instruments. This has taken two paths, one for ASCII-controlled test assets and another for driver-controlled test assets. Both paths have similar histories.

The ASCII path started with the advent of the IEEE-488 or GPIB bus. This was used for 10 years before the development of SCPI (Standard Codes for Programmable Instruments). Messages were in ASCII, but there were no standardized semantics. Different companies used different commands for the same functionality. Even different divisions of the same company used different commands for similar functions. The situation improved when the IEEE 488.2 standard was released. This produced a few common commands and protocols, but there still was no semantic standard for instrument functionality.

The SCPI language was created to solve this problem. SCPI was very helpful to application programmers because it provided an extensive set of standardized commands, but it did not deliver much asset interchangeability. This is because the asset providers own these interfaces and need to use them to expose all of the features of their product. The requirements of the free enterprise system make it impossible to standardize unique features or even new features.

The driver path started ten years later than the ASCII path but it is following a similar history. Early driver-controlled test assets were created with no interface standards. Later, the VXI Consortium developed the VXIplug&play driver standard, which provided improvements similar to those that were provided by IEEE 488.2 for ASCII programming. The VXIplug&play standard established a few common commands, but there still was no semantic standard for instrument functionality. For this reason, VXIplug&play did not provide any asset interchangeability. Semantic standardization for drivers is finally being developed as a result of the work of the IVI Foundation, which is now creating standards for the software interfaces of common instrument types.

There are more changes in store for drivers. In the past, drivers have had the reputation of being of poor quality, incomplete, unsupported, and free. On the other hand, SCPI interfaces have been of good quality, complete, supported, and considered part of a test asset product. As the test and measurement world moves from ASCII to driver programming, it seems appropriate that the “primary automation interface” of test assets will move from SCPI to Application Programming Interfaces (APIs). As a result of this, the time will come when test assets will directly present APIs and not SCPI interfaces. As this happens, drivers will become part of the product and will also be supported, complete, and of high quality. Unfortunately, they will no longer be available in source code form. This means that another place will need to be identified to apply code that is needed for customer adaptation. IVI-MSS, described later, provides an answer for this.

In addition to this, there are technology changes taking place that affect drivers. There have been many problems with ANSI-C drivers that can be solved by the use of ActiveX technology, which is now available on most major platforms. All of this is emerging[9].

Measurement and Stimulus Subsystems

Hewlett-Packard’s Measurement Subsystem work[5] is another approach to test asset interchangeability. This work was done specifically to make it possible

to deliver a complex measurement solution so that the software investment could outlive the associated test assets. To bring the story up to date, the HP E5500 Phase Noise Measurement Subsystem[6], now supports some 30 different test assets. This provides the end customer with tremendous flexibility in use of the system. It was the discontinuance of a test asset that resulted in the initial development of this system architecture. It is interesting to note that just this year, one of the supported assets for the new system, a spectrum analyzer, unexpectedly was discontinued. It was replaced using the concepts described in this paper, and delivers the same results as the prior asset. The effort to do this, including the testing burden, was small.

This work is now being carried forth under the sponsorship of the IVI Foundation. A subgroup of the IVI Foundation is focused on creating a formal specification that will enable other solution providers to use the Measurement Subsystem Architecture. In this paper, the work is referred to as IVI-MSS for Measurement and Stimulus Subsystems. Other names that have been used at different times are the Measurement Subsystem Architecture (MSA), and Measurement Subsystems.

WHAT IS IVI?

Interchangeable Virtual Instruments (IVI) is the name of an industry group originally formed in August 1998. It is focused on the challenge of test asset interchangeability. The IVI Foundation, now consisting of 29 companies [3], is an excellent group of concerned and interested end users and test asset providers. The announcement of the formation of the IVI Foundation, and the offering of the HP-developed Measurement Subsystem Architecture as the basis of an industry standard, occurred within a few days of each other. Since these two efforts are focused on different aspects of the problem of test asset interchangeability, it was proposed that they be developed together under the same organization. It is expected that this will reduce confusion over conflicting claims. Subsequently, the IVI-MSS subgroup of the IVI foundation was formed with the mission to create a specification for building subsystem solutions.

What are IVI-Drivers?

IVI Class Drivers provide standardized API functions for test assets, and by themselves provide a certain measure of interchangeability. When the utilization of a test asset is limited to the common class driver functions and there are no problems with "secondary effects" of instruments (see “Second Order Effects”), the additional requirements of IVI-MSS are not needed.

It is important to note that neither the asset producer nor the driver provider can be expected to “guarantee” that an interchanged asset will deliver the same answer.

What is IVI-MSS?

Measurement and Stimulus Subsystems (IVI-MSS) are solutions that deliver test and measurement functionality. These solutions often involve two or more physical test assets that are used together to perform the required functions. Associated with IVI-MSS is a set of design rules and principles, which, if followed, delivers the following values that go beyond the capabilities of IVI-Drivers:

1. The ability to use an instrument of a different type to fulfill the functions served by another. (Of course, the instrument has to be capable of doing the required operation.)

2. A place to put software code that is necessary to take care of "second order" instrument effects and other peculiarities. These are the things that, besides semantics, keep one instrument from performing like another.

3. A way to create solutions that involve more than one physical instrument. The resultant code is reusable, making it possible for the code to be separate from application programs.

4. A guarantee of the “same answer” after an asset is interchanged. Because of clearly established rules on the ownership of interfaces, this becomes possible for a solution provider.

5. A means of achieving software reuse for complex measurements and stimulus scenarios.

HOW DO YOU VERIFY INTERCHANGEABILITY?

The simple answer is by testing. The difficulty depends on how well understood the interfaces are behind the interchange. In the case where a test asset is interchanged, the first requirement is that interface features of one asset must be available in the other. Unfortunately, this is seldom the case.

The problem then comes down to finding out which interface features were actually used in the application. This is almost impossible in today's large ATE systems that operate many TPSs. This is particularly true in systems where TPS software is allowed to directly interact with test assets. In systems like these, the only way to verify that the "same answers" are obtained after an interchange is to collect all of the LRUs or DUTs along with their associated TPSs and check every one. Because this is so undesirable, a demand has been created for better approaches to the problem.

The problem of verification is one of clearly defining interfaces and controlling their semantics and usage. IVI-MSS specifically addresses this problem with the following two elements: 1) the solution provider knows exactly what capabilities are used, and 2) the responsibility for performing the verification is placed directly on the solution provider. The term "solution provider" refers to organizations that "own" the interfaces in an Automated Test System and are able to guarantee the functionality behind them. Clearly, the problem of test becomes simpler as the number of things to test is smaller and better defined. A key design principle in IVI-MSS implementations is to only expose the functionality that is needed. This simplifies the testing burden.

Verification includes two tasks: 1) Making sure that all of the required APIs of an interchanged component are found and match up, and 2) Verifying that the client of the API receives the "same answers" or results from use of the new components. This verification is necessary to make sure there are no "secondary effects" or other asset peculiarities.

The "guarantee" is possible because the guarantor knows what to test. Facilitating interchangeability comes from reducing the burden of test and verification by breaking out modular components.

IVI-MSS ARCHITECTURE

Overview

A key factor in the development of the Measurement Subsystem Architecture is clearly establishing who owns the various interfaces in a test system, and how they're developed, supported, and managed over the life cycle of the system. An IVI-MSS solution inserts a new "owner" between the ATS specifier and the asset provider. The ATS specifier relies on the solution provider to provide a stable API for measurement or stimulus servers. This is the interface that the solution provider "guarantees" will be supported over 10 to 20 years.

IVI-MSS provides a standard way to create and sell test and measurement solutions where part of the value is in software instead of just hardware assets.

In order for products like this to be accepted by customers, they need an interface with a familiar, comfortable look and feel. It is also necessary to overcome the perception that the products rely on proprietary technology. The work currently being done by the IVI Foundation should help this occur.

Architecture

The distinguishing characteristic with IVI-MSS is the introduction of two new software interfaces between the TPS programmer and physical test assets.

One of these is the interface to a new software component called an Asset Control Module (ACM) (figure 3).

The other new interface is for a Measurement Server or Stimulus Server. This interface provides solutions that are both client- and asset-independent.

The solution provider owns the interface between a Measurement Server and all associated ACMs. To achieve interchangeability, the solution provider creates new ACMs for interchanged assets and thoroughly tests their interfaces after an interchange. The new ACM is the place to put any unique code that is necessary to deal with any peculiarities of the new asset.

In the example of a Measurement Subsystem (figure 2), a Measurement Server is associated with three ACMs and their associated physical test assets. In addition, the example shows three possible client users of the subsystem. One is a GUI for manual use, the second is a SCPI implementation for use by legacy ATE systems, and the third is a COM-based API for direct use by client software.

Additional GUIs are available in a subsystem because experience has shown that it is helpful to be able to observe and interact with each level of a system hierarchy. There are also two common components, the Asset Server and the Event Server, that are shared among multiple subsystems.

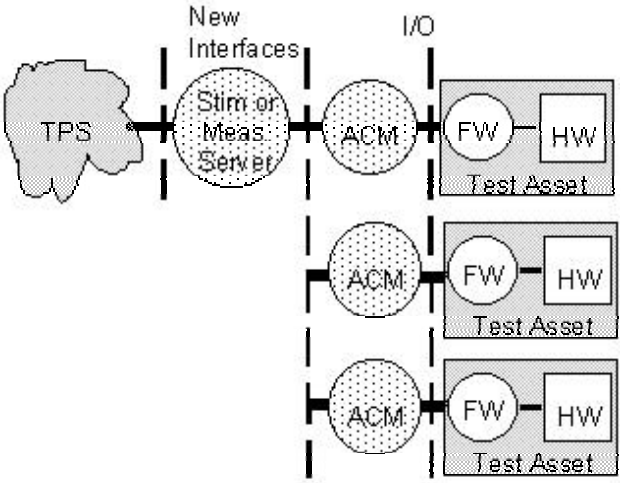

Figure 2. Measurement Subsystem

Measurement Servers and Stimulus Servers

Each unique measurement domain would have its own Measurement Server or Stimulus Server. Typical ATSs can have multiple servers. Examples might include radar measurements, distortion measurements, phase noise measurements, and complex signal stimulus.

Each Measurement or Stimulus Server is associated with a set of two or more Asset Control Modules (ACMs). Each ACM implements a "role" for its associated server.

Making good decisions on which code should be put in a Measurement Server and which code should be put in an ACM is critical to the design of a new measurement subsystem. The IVI-MSS subgroup intends to provide guidelines to assist solution developers in making these decisions.

Asset Control Modules (ACM)

An ACM is required for each asset used by a Measurement or Stimulus Server in a subsystem. ACM are not required for other assets in an ATS. The left side of the ACM (figure 3), shows the interface to a Measurement Server. This interface presents only the features and capabilities required for a single "role" in a subsystem.

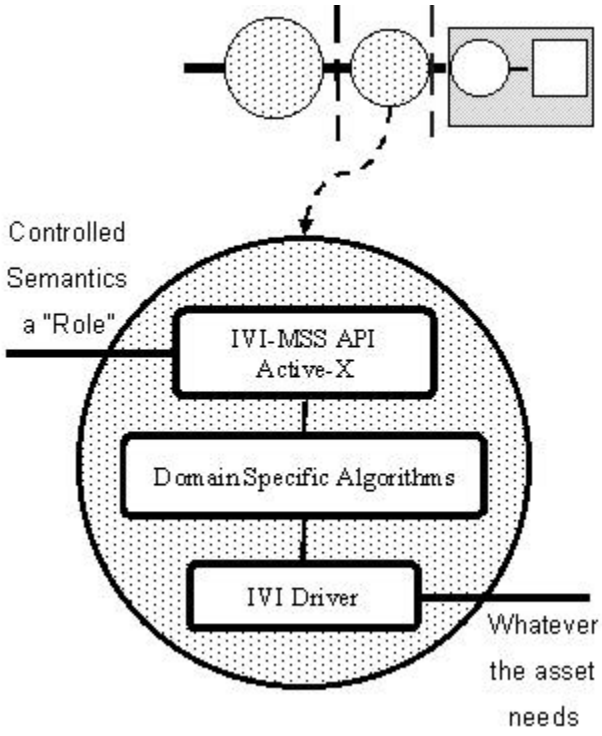

Figure 3. Asset Control Module (ACM)

The right side of the ACM shows the interface to a physical asset. The ACM may do whatever is necessary to communicate with its associated asset. An IVI-Driver is being used for asset communication (figure 3). This is not required; however, using an IVI-Driver provides the benefit that in some cases, an interchange may be possible without creating a new ACM.

When necessary, an ACM contains the software that will make a lesser asset "measure up" to that of a more capable one. For example, if a digitizer card is to replace an FFT analyzer in a measurement subsystem, the ACM must first acquire the time record, then perform an FFT, and finally return the measurement result according to the formatting requirements of the associated "role." This code goes in the "Domain Specific Algorithms" block. Any measurement software that would have to be rewritten for a new asset must be put in the ACM and not the Measurement Server. ACMs are part of the value-added for a Measurement Subsystem. They can be individually sold and license-key protected if desired. ACMs are the key element in providing interchangeability of assets.

ACMs are the responsibility of their client. This means that whoever defined the ACM roles and developed the associated Measurement or Stimulus

Server is responsible for the development and verification of ACMs for that role.

Asset producers are responsible for providing the primary automation interface for their assets such as SCPI interfaces or IVI-Drivers. They cannot be responsible for developing or proving the ACMs that are needed by a subsystem unless they also provide the subsystem.

What is a Role?

A “role” is a specific interface specification between a Server and an ACM. It is defined by the solution provider and must deliver the specific functionality that is needed by the associated Measurement or Stimulus Server. ACMs are subsystem-specific. Different subsystems would use different ACMs for the same asset. To keep the burden of test low, it is vital that a role does not expose unneeded features of an asset.

An ACM frequently acts as a “bandpass filter” on the feature set of physical assets. It provides what a client server needs and NOTHING else. This is what keeps the burden of test manageable after an interchange. A role has a rough correspondence to various types of instrumentation. A real Measurement or Stimulus Subsystem generally requires several roles.

TECHNOLOGY IMPACTS

The emergence of the Windows’ platform as well as component software design technologies has had a tremendous impact on the test and measurement world. Another significant technology change is the use of client-server architectures.

The PC platform is pervasive with ever increasing performance. It is a natural place to implement functionality beyond what is “in the box” or on a card.

Software reuse has been frequently touted in the literature over the last decade, but until the CORBA[7] and COM Object Request Broker (ORB) implementations came along, very little real progress had been made. For a few years, there was a lot of confusion over these two competing implementations. At the time of this writing, COM[8] and its related technologies, DCOM and ActiveX, are becoming pervasive and implementations are available on all major platforms[9] including Linux, Sun-Solaris, HP-UX, and Windows. ActiveX is a very promising technology and should provide an unprecedented amount of interoperability between different software application environments.

Work is needed to standardize how technologies are used between IVI-Drivers, IVI-MSS, and other system components.

SUMMARY OF IVI-MSS PRINCIPLES

1. Measurements have value independent of instruments, applications, and TPSs.

2. A measurement’s utilization of the features of instruments must be rigidly controlled by the use of “roles.”

3. Asset Control Modules allow subsystem providers to guarantee a support life beyond the life of the utilized instruments.

4. It is crucial to know who owns various interfaces.

5. The primary automation interface of a test asset can never be the interface of interchange because the client doesn’t “own” it. This is regardless of whether the interface is ASCII or a driver interface. These interfaces must never be used directly by application programmers or TPS developers in systems that must deliver interchangeability.

6. General-purpose interfaces that expose all of a test asset’s features are difficult (if not impossible) to test.

7. The ability to test and verify increases as API complexity decreases.

8. The ability to interchange increases as the ability to test increases.

9. For interchangeability to be achieved at an interface, the client of the interface must "own" it and be responsible for the test and verification after the interchange.

10. The abstraction of a measurement serves to provide a natural isolation from asset peculiarities.

11. When there is a small, well-defined interface where the differences between two assets are exposed, testing can be limited to this interface.

12. Extra layers of abstraction in a system naturally simplify the burden of test and make it easier to deliver interchangeability.

13. The number of potential interchangeable test assets is inversely proportional to the number of the exposed features in an API.

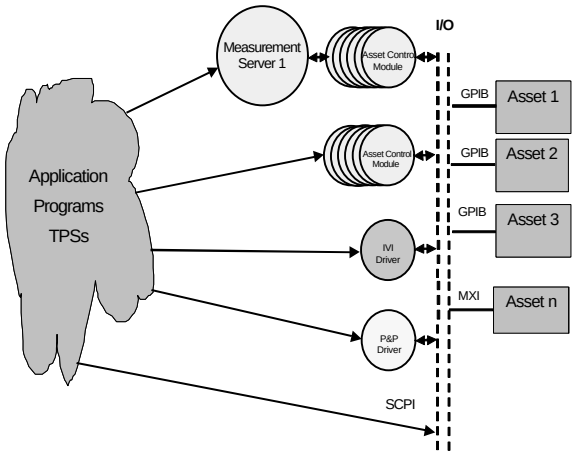

Figure 4. ATS with Various Interfaces

FULL ATS IMPLEMENTATIONS

Figure 4 shows an ATS implementation where an application is allowed to talk directly to different types of interfaces. This demonstrates that IVI-Drivers, IVI-MSS, SCPI, VXIplug&play, and other types of interfaces can all coexist within a given system. An application program can interact with assets in any of the ways shown in the figure. If an application program talks directly to an instrument's SCPI interface or if it directly calls an instrument's plug and play driver functions, there can be no interchangeability without changing the application program.

Calling the functions of an IVI-Driver provides semantic interchangeability.

Calling the functions of an ATS-specific set of ACMs can provide "same answer" interchangeability for dissimilar assets. In the case where the application program is shown interacting directly with ACMs, the assumption should be made that the application program is serving the function of an IVI-MSS Stimulus or Measurement Server.

Calling the functions of a Measurement Server delivers "same answer" interchangeability as well as reuse of multi-instrument measurement or stimulus scenarios.

Figure 5 shows an ATS implementation that adds an additional abstraction layer for TPS software. This is very similar to the DoD ATS R&D IPT architecture[5]. This type of system might use a signal-descriptive language such as ATLAS. The ATS system would do the mapping between the signal-descriptive semantics and the supported stimulus or measurement interfaces in the system. When Asset Control Modules do all of the asset control, truly robust interchangeability can be achieved.

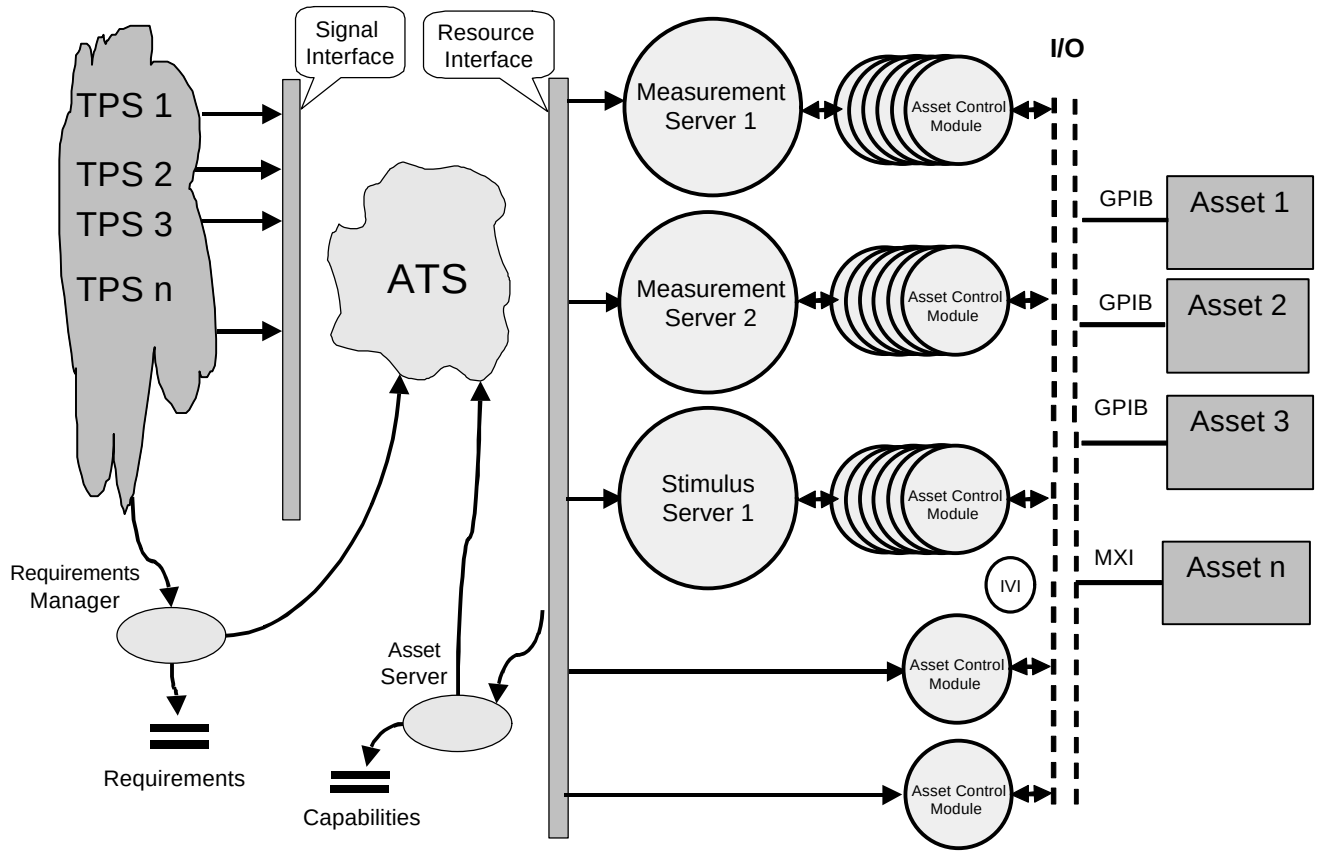

Figure 5. ATS with Signal Interface

STANDARDIZED COMPONENTS

Several components are shared among multiple Measurement or Stimulus Subsystems. It is expected that the IVI-MSS specification that is under development will document the interfaces of these components.

Asset Server

The Asset Server uses the NT registry to store information about each instrument that is used in the system. Information like the I/O bus, I/O address, I/O library, and the selection of a specific asset for use by a Measurement Subsystem is managed by this component.

Event Server

The Event Server accepts call-outs from the various components to collect asynchronous events. It time-stamps and records this data and makes it available for a variety of uses.

Common Base Classes

Common Base Classes for both Measurement Servers and Asset Control Modules can make it easier to build Measurement Subsystems and will contribute to their overall quality and common “look and feel.”

Path and Calibration Management

Managing switching systems that establish signal paths, and properly utilizing calibration data and measurement uncertainties should also be done in a standard way in ATS implementations.

CONCLUSIONS

To successfully build Automated Test Systems that deliver robust interchangeability, the learnings and experiences of all the different approaches need to be leveraged and built upon. This includes ATLAS, SCPI, VXIplug&play, IVI-Drivers, and IVI-MSS. This can be accomplished by taking the following actions:

1. Use signal-descriptive semantics as close to the TPS as possible.

2. Standardize the semantics used to control test instrumentation.

3. Use IVI-MSS Measurement Servers for multi-asset measurements.

4. Utilize IVI-Drivers where available to reduce the effort required to interchange an asset.

There are still some challenges to work out. One of these is to harmonize the underlying software technologies that are being used. IVI-MSS is COM-based, and at present IVI-Drivers are ANSI-C-API-based. Work is in progress to harmonize both efforts under ActiveX technology. When this is accomplished, it will be simpler than ever before to build robust Automatic Test Systems in a great variety of application development environments.

When TPS developers, primes, or other providers use IVI-MSS to build products that implement test and measurement solutions, it is possible for the providers to offer support for an extended period of time such as 20 years. The underlying hardware most likely will have to change, but when there is new hardware that has the necessary capabilities, it can be utilized and the “same answer” can be obtained. Because of the use of Asset Control Modules, the burden of test after the interchange is manageable. IVI-MSS-based solutions can be utilized from a wide variety of application development environments and can be integrated with multi-vendor solutions

GLOSSARY

- **API** – Application Programming Interface. The software interface to a test asset or other software component.

- **Asset Control Module (ACM)** – A key software component of a Measurement or Stimulus Subsystem that translates role interface functions into asset-specific actions. There is one ACM for each asset in each subsystem.

- **Asset Server** – A common IVI-MSS component that keeps track of all assets on an ATS.

- **Event Server** - A common IVI-MSS component that manages asynchronous event communications on an ATS.

- **IVI-MSS** – A written specification, under development by the IVI Foundation, that specifies how to create Measurement or Stimulus Subsystems.

- **IVI-Drivers** – A set of written specifications, under development by the IVI Foundation, that specifies the semantics of common test instruments.

- **Measurement Server** - The core software component of an IVI-MSS subsystem. It presents an API to application programmers, and when desirable, aggregates the behavior of multiple test assets.

- **Measurement Subsystem** – A solution built using the IVI-MSS design rules and common components.

- **Robust** – Pertains to test asset interchangeability where it is possible to guarantee the same answer, where it is possible to interchange an instrument of a different basic type, and where it is possible to compensate for instrument peculiarities.

- **Role** – The interface between an Asset Control Module and a Measurement Server.

- **Same Answer** - A requirement for test asset interchangeability.

- **Secondary Effects** - Those test asset differences--beyond semantics--that interfere with

interchangeability.

- **Semantics** – The meanings associated with specific spellings of ASCII commands, function names, or object method names.

- **SCPI** – Standard Commands for Programmable Instruments. A semantic standard for the ASCII programming of test assets.

- **TPS** – Test Program Set. A software program and associated fixture hardware needed to test a given DUT or LRU.

- **VXIplug&play Drivers** - A software interface to a test asset that provides C API's but little semantic standardization.

REFERENCES

[1]Roger Oblad, "Applying New Software Technologies To Solve Key System Integration Issues." AUTOTESTCON-1997 Proceedings, Piscataway, New Jersey. IEEE, pp.181-189.

[2] Ned Barnholt, "Connecting You to the Future." Keynote Address, AUTOTESTCON-1998. http://www.tmo.hp.com/tmo

[3] IVI Foundation: http://www.ivifoundation.org/

[4] ATS Architecture: Developed by the Test Resource Working Group within the DoD ATS R&D IPT organization (ARI).

[5] Measurement Subsystems, HP-TMO: http://www.hp.com/go/ivi

[6] HP E5500 Phase Noise Measurement Subsystem: http://www.tmo.hp.com/tmo/datasheets/English/HPE5500_Family.html

[7] Corba: The Object Management Group (OMG): http://www.omg.org/

[8] COM Technology: http://www.microsoft.com/com/

[9] EntireX: Multi-Platform Support of ActiveX: http://www.softwareag.com/corporat/solutions/entirex/entirex.htm