THE ROLE OF A SIGNAL INTERFACE IN SUPPORTING INSTRUMENT INTERCHANGEABILITY

Narayanan Ramachandran
TYX Corporation
1910 Association Drive
Suite 200
Reston, VA 20190
(703) 264-1080
nr@tyx.com

Roger P. Oblad
Electronic Products and Solutions Group
Agilent Technologies
1400 Fountaingrove Parkway
Santa Rosa, CA 95403 USA
(707) 577-3466
Roger_Oblad@agilent.com

Ion A. Neag
TYX Corporation
1910 Association Drive
Suite 200
Reston, VA 20190
(703) 264-1080
ion@tyx.com

David F. Tyler
TYX Corporation
1910 Association Drive
Suite 200
Reston, VA 20190
(315) 336-6579
dtyler@tyx.com

Abstract – This paper investigates the principles of instrument interchangeability in Automatic Test Systems by identifying the effects of instrument replacement and by analyzing the existing solutions for interchangeability problems. The "robust" instrument interchangeability provided by the IVI-MSS approach is presented in detail. The solution proposed in the paper consists of the standardization of a Signal Interface as semantic contents for IVI-MSS interfaces. Besides the inherent advantages of the IVI-MSS architecture, this approach provides portability of components among different testing environments. The functional requirements, integration issues and associated business model for the proposed solution are analyzed.

Introduction

In the present paper, "instrument interchangeability" is understood as the ability to replace a given instrument with an alternative instrument of sufficient capability, but of a different model, class, design or manufacturer.

Instrument interchangeability is a critical supportability issue in application fields such as avionics, nuclear power plants, transportation and weapon systems, where the Units Under Test (UUTs) and their associated Test Program Sets (TPSs) have operational lifetimes covering several generations of test instrumentation. As the inventory of existing Automatic Test Systems (ATSs) ages, without the prospect of many newer systems becoming operational soon, maintenance organizations face the issue of ATSs approaching obsolescence. Typically, three solutions are considered:

1. Total replacement with new Commercial Off The Shelf (COTS) ATSs, requiring the redevelopment of all test programs and their associated documentation.

2. Updating the existing ATS's computer and instruments and re-hosting the legacy TPSs to run on the new computer's operating system and with the new instruments.

3. Migrating existing TPSs to newer ATE systems with the same or greater capabilities,

thus reducing the multiplicity of ATSs in the organization.

The development of new TPSs represents a large investment, the cost of a complete TPS (with interfacing devices and documentation) ranging typically from $80,000 to $250,000.

The re-hosting of legacy TPSs in order to support new instruments is also very expensive and sometimes logistically impossible, due to the lack of UUT data, software development support or programming expertise for the original development language. Moreover, TPS changes require a new Independent Validation and Verification (IV&V) process, which is expensive and time consuming [10].

Consequently, the replacement of instruments must be possible without changing the TPSs, while the test results are guaranteed to remain identical. In the following, this feature will be called “robust interchangeability” [9].

Principles of Instrument Interchangeability

ATS Architecture

The following analysis of interchangeability principles is based on a generic Automatic Test System (ATS) architecture, derived from the Automatic Test Systems Subdomain Annex of the DoD Joint Technical Architecture (JTA) [3] (Figure 1). The Resource Adapter Interface component defined in [3] was omitted from the picture because it is not supported by standards currently in use. The functional requirements assigned to this component by the JTA will be analyzed in the next sections of the paper.

Instrument-based TPSs perform the direct control of instruments through the following methods:

1. Sending string commands through the Bus Driver or the Instrument Communication Manager (I/O Library).

2. Calling functions of Instrument Drivers.

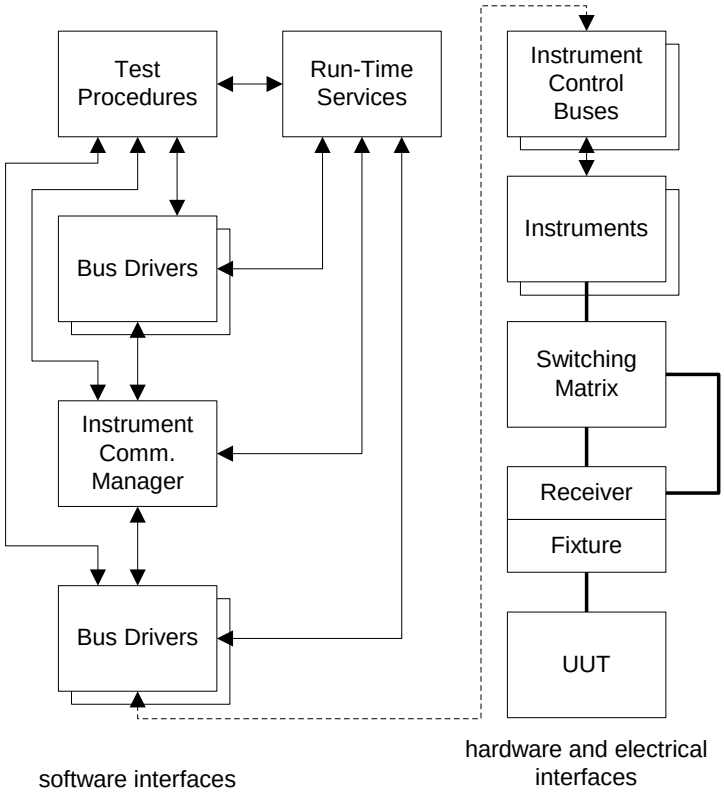

Figure 1. Generic ATS Architecture

For signal-based TPSs, the control of instruments is performed by the Run-Time Services of the ATS, also using one of the above methods.

Effects of Instrument Replacement

As visible in the generic ATS architecture, the replacement of an instrument (along with its driver, if applicable) may involve changes at the following **interfaces** of the ATS architecture:

1. Software interfaces between Test Procedures and the Bus Drivers:

    1.1. The new instrument uses a different control method (string commands vs. driver calls).

    1.2. Both instruments are controlled through drivers, but the new driver provides different functions, because the new instrument is of a different model or belongs to a different class.

    1.3. Both instruments are controlled through string commands, but the new instrument has a different command set, because it is of a different model or belongs to a different class.

2. Hardware interfaces between the Bus Driver and the Instrument; the new instrument uses a different Instrument Control Bus.

3. Electrical interfaces between the Instrument and the UUT; the new instrument has differently organized ports (e.g. a different number of outputs).

The above problems become more complex when **multiple instruments** are involved in the replacement (for instance, when several instruments are replaced by a unique new instrument).

An additional interchangeability problem, often overlooked, is the **behavior** of the instrument (or the instrument-driver subsystem, if applicable). The following situations are identified:

1. The new instrument provides a **different answer**, which may consist of different physical signals generated in response to the same commands or driver calls, or different measurement results for identical physical signals. This may be due to range, resolution or precision differences that are not properly taken into account when planning the replacement, or to more subtle causes such as different methods and algorithms implemented in drivers or the instrument firmware [9].

2. The new instrument has a **different state behavior**. The “state” of an instrument is characterized by a set of attributes (settings) such as ranges, operational modes, etc. For different instruments, the state may have different evolutions in response to identical sequences of commands or driver calls, starting from the Reset or Power-on state. For example, the attributes may have different values after Reset or Power-on. Calling a function or changing an attribute may affect the other attributes differently, for different instruments.

In addition, the replacement of an instrument may require a new **calibration** of the signal paths that include the instrument and its connection cables.

Existing Solutions for the Interchangeability Problems

The interchangeability problems described before are addressed by several families of standards developed over the years. The benefits and limitations of these standards will be summarized in the following.

IEEE-488

IEEE-488 is an Instrument Control Bus standard, primarily addressing the hardware interface between the Bus Driver and the Instrument. The IEEE-488.2 standard addresses the software interface between Test Procedures and Bus Drivers for command-based instruments, by specifying a small set of common commands and protocols. This level of standardization insures consistency in usage, but does not provide a common semantic for instrument control (i.e., identical commands for the same functions). In consequence, although TPS changes are always required when instruments are replaced, this operation is simplified to some extent.

SCPI

The Standard Commands for Programmable Instrumentation (SCPI) standard addresses the software interface between Test Procedures and Bus Drivers for command-based instruments, by specifying common sets of commands for diverse classes of instruments. The level of interchangeability provided by SCPI is limited by the need of instrument vendors to extend the command set in order to expose instrument-specific functionality [9]. This means that any use of instrument-specific functionality may require TPS changes when the instrument is replaced, thus compromising the interchangeability. Additionally, SCPI does not support interchangeability solutions involving multiple instruments.

VXIplug&play

The VXI*plug&play* standard addresses the software interface between Test Procedures and Bus Drivers for driver-based instrument control by specifying a small set of common functions in the driver's Application Programming Interface (API). Similar to IEEE-488.2, this offers consistency in usage but does not provide common semantics.

IVI Drivers

The IVI Foundation [7] is currently developing a set of Class Driver standards. These standards address the software interface between Test Procedures and Bus Drivers for driver-based instrument control by specifying common driver APIs for diverse classes of instruments. The interoperability of IVI Drivers from different vendors is supported by a standardized architecture and set of standardized IVI Common Components. IVI Drivers will be available with COM interfaces.

The interchangeability provided by IVI Class Drivers is limited by the following factors:

1. The class standards deal with the inherent differences in instrument capabilities by defining "extension groups", which support the capabilities provided by a limited, but significant, number of existing instrument models. This approach is expected to cover about 80% of the existing instrument models. Consequently, the access to instrument-specific functionality still requires vendor-specific extensions to driver APIs, in the form of "specific driver interfaces". The direct use of **instrument-specific functions** in the TPS code compromises interchangeability.

2. The attribute model defined by the class standards does not cover the interaction of attributes and the effects of function calls on these attributes. Consequently, the interchangeability may be compromised by differences in **instrument state behavior**.

3. The class standards do not address the problem of **different answer**, as previously defined in the paper.

4. Because the standardized interfaces are instrument-class specific, the class standards do not support replacement with instruments from a **different class** and **multi-instrument** replacement solutions.

In conclusion, although an important step forward in providing interchangeability for driver-controlled instruments, the IVI Class Driver standards do not provide "robust" interchangeability, as defined in the Introduction of the present paper. When instruments are replaced, the use of IVI Class Driver guarantees (with some limitations) the execution of TPSs, but does not guarantee identical test results.

Role of IVI-MSS in Providing Instrument Interchangeability

The IVI-MSS (Measurement and Stimulus Subsystem) standard is based on the "Measurement Subsystem Architecture" developed by the Hewlett-Packard Company [8]. This architecture design was offered to the industry [1] and later included under the sponsorship of the IVI Foundation [9]. This has opened the way for unification of concepts and architecture, extending the range of interchangeability solutions offered by the IVI Foundation.

Objectives

The IVI-MSS standard addresses most of the interchangeability problems outlined before, according to the following objectives [9]:

1. Allow replacement with instruments from a **different class**.

2. Support interchangeable **multi-instrument** measurement and stimulus solutions.

3. Provide a place for instrument-specific code that compensates for instrument peculiarities that produce **different answers**.

4. Promote a business model that requires the vendor to **guarantee the same answer** when an instrument is replaced.

5. Support **software reuse** for complex measurement and stimulus solutions.

Architecture

While interchangeability problems involving interface incompatibilities may be solved by standardizing the syntax and semantics of these interfaces, the problems caused by **differences in instrument behavior** require an architectural solution.

The IVI-MSS standard defines an architecture with additional layers of COM components between the TPS and the instruments (Figure 2).

The Role Control Modules (RCMs) are instrument-specific components that provide a place for code that compensates for differences in instrument behavior. The interfaces of these components implement an IVI-MSS "role", which means they have rigid semantics and are "owned" by the client (e.g., the developer of the Aggregation Component).

The Measurement and Stimulus Subsystems (MSSs) are components that interact with more than one RCM to implement multi-instrument measurement and stimulus solutions. They may also expose "role" interfaces.

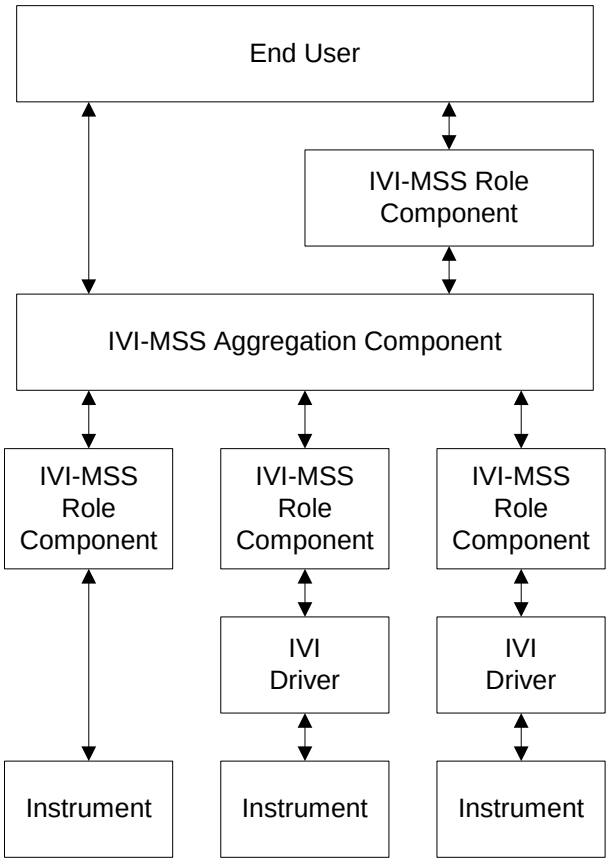

Figure 2. IVI-MSS Architecture

All IVI-MSS Components, as well as the IVI Drivers, use the services of IVI Common Components:

1. The IVI Factory, which instantiates the components.

2. The IVI Config Store, which provides access to configuration information defining the hierarchy of components.

3. The IVI Event server, which allows components to communicate through events.

The above architecture supports complex ATS implementations, potentially combining IVI-MSS components, IVI Class Drivers, VXI*plug&play* drivers and direct instrument control [9]. It also supports signal-based ATS architectures, as described in the following.

Interface Ownership

As specified above, "role" interfaces are "owned" by the client. This means that the organization developing Test Procedures or Measurement and Stimulus Subsystems using RCMs specifies the contents and the semantics of RCM interfaces. RCM developers are responsible for implementing and guaranteeing the specified behavior.

Because "role" interfaces are not required to expose all the functionality of the instrument, they are less complex than the interfaces of IVI Drivers. This simplifies specification and verification of functionality, reducing the cost of initial development and the costs incurred by subsequent instrument replacement operations [9].

In conclusion, IVI-MSS provides robust instrument interchangeability by defining an architecture and specifying ownership rules for the semantic interfaces. The specification of semantic contents for these interfaces is outside the scope of IVI-MSS. The **reuse** of IVI-MSS Components from different solution providers among different test environments requires the standardization of interface semantics. Such standards may be application domain-specific, for instance dedicated to RF measurement. A domain-independent solution based on the signal-based testing paradigm is proposed in the following.

Role of Signal-Based Testing in Providing Instrument Interchangeability

The Signal-Based Testing Paradigm

Within the current test software technology, test procedures may be developed using one of the following approaches ("testing paradigms"):

1. **Instrument-based testing**. Test procedures specify the behavior of instruments through instrument-specific control methods such as string commands or driver calls. The connection of instrument ports to the UUT pins is controlled by specifying in the test procedures the behavior of the Switching Matrix, through device-specific control methods.

2. **Signal-based testing**. Test procedures specify the desired behavior to be obtained at the pins of the UUT, in terms of signals to be applied and measured. The selection of appropriate instruments may be performed automatically by the development environment and/or the run-time environment, through automatic resource allocation. The behavior of the Switching Matrix is determined automatically by the development environment and/or the run-time environment, through automatic switching.

Two code samples included below illustrate the implementation of the same signal generation operation using both testing paradigms. The generated signal is sinusoidal, with 1V amplitude and 100kHz frequency, and must be applied at the UUT pins labeled `J1-1` and `J1-2`. The ATS hardware (Figure 3) consists of a function generator, the Switching Matrix and the Receiver (R), while the TPS includes Test Procedures and the Fixture (F).

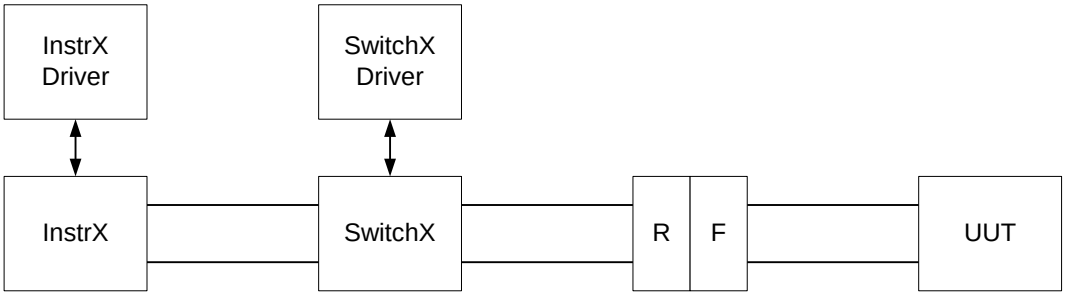

Figure 3. Example of ATS Hardware

The following code fragment is extracted from an instrument-based test procedure that controls the function generator and the switch through IVI Drivers:

```
IviFgen_init("GPIB:22:INSTR",
  VI_TRUE, VI_TRUE, &viFgen1);
IviFgen_ConfigureStandardWaveform(
  viFgen1, "CH1",
  IVIFGEN_VAL_WFM_SINE,
  1, 0, 100E3, 0)

IviSwtch_init("GPIB:17:INSTR",
  VI_TRUE, VI_TRUE, &viSwtch1);
IviSwtch_Connect(viSwtch1,
  "IX-HI", "P10")
IviSwtch_Connect(viSwtch1,
  "IX-LO", "P11")

IviFgen_InitiateGeneration(viFgen1)
IviFgen_EnableOutput(viFgen1)
```

It may be observed that the above code contains references to the instrument's I/O address, to an instrument channel and to instrument ports. The driver calls are instrument-class specific and the function `IviFgen_ConfigureStandardWaveform()` belongs to an extension group of the "IVI function generator" class. All the above elements limit interchangeability.

The following ATLAS code implements equivalent functionality in a signal-based test procedure:

```
REQUIRE, 'AC_SIG_GEN',
  SOURCE, AC SIGNAL,
  CONTROL,
    VOLTAGE RANGE 0V TO 2V BY 1MV,
    FREQ RANGE 10HZ TO 1MHZ
      ERRLMT +-0.1PC,
  CNX HI J1-1 LO J1-2 $

APPLY, AC SIGNAL
  USING 'AC_SIG_GEN',
  VOLTAGE 1.0V,
  FREQ 100.0KHZ,
  CNX HI J1-1 LO J1-2 $
```

The first statement describes the requirements for a signal, indicating the following:

1. signal role (`SOURCE`)
2. signal type (`AC SIGNAL`)
3. minimum range, resolution (`BY` keyword) and precision (`ERRLMT` keyword) for signal parameters Voltage and Frequency

The second statement specifies a signal generation operation, indicating the values of signal parameters and the UUT pins (`J1-1`, `J1-1`) where the signal ports (`HI`, `LO`) must be applied.

To support signal allocation and automatic switching, signal-based ATSs must contain information about the capabilities and the connectivity of Instruments, Switching Matrix and Fixture (Figure 1). This information is typically provided in text files, using a description language. The standardization of this language is addressed by the IEEE Standard for Test Equipment Description Language (TEDL) [5].

For example, an ATS including an instrument that may be allocated to the signal from the above ATLAS code may contain the description presented in the following, expressed in a TEDL-like language.

The **instrument description** shown below specifies the following: signal role and signal type; signal capabilities, in terms of range, resolution and precision for signal parameters; signal connectivity, in terms of instrument ports (`IX_HI`, `IX_LO`) where the signal ports (`HI`, `LO`) are applied.

```
DEVICE_MODEL InstrX
  FUNCTION SOURCE
  NOUN AC_SIGNAL
    SIG_CHAR
      VOLTAGE RANGE 0V TO 5V
        BY 0.1MV
      FREQ RANGE 1HZ TO 10MHZ
        ERRLMT +-0.05PC
    AT HI IX_HI LO IX_LO
END DEVICE_MODEL InstrX $
```

The simplified **Switching Matrix description** presented below specifies the signal paths that may be closed by the Switching Matrix between the instrument ports and the pins of the Receiver-Fixture interface (`P120`, `P11`).

```
SWITCH_MODEL SwitchX
  PATH p1 CONNECTS IX_HI TO P10 $
  PATH p2 CONNECTS IX_LO TO P11 $
END SWITCH_MODEL SwitchX
```

The simplified **Fixture description** presented below specifies a set of hardwired connections between the pins of the Receiver-Fixture interface and the pins of the UUT (`J1-1`, `J1-2`).

```
ADAPTATION_MODEL Ita1
  UUT Uut1
  PATH p1 CONNECTS P10 TO J1-1 $
  PATH p2 CONNECTS P11 TO J1-2 $
END ADAPTATION_MODEL Ita1
```

The capabilities of the instrument `InstrX` satisfy the requirements specified in the ATLAS code from the previous example. Moreover, the Switching Matrix is able to provide all the required signal paths between signal ports and UUT pins. Consequently, this instrument may be allocated to the ATLAS signal.

Because they do not specify instrument control operations and do not contain references to instrument ports, signal-based test procedures are inherently instrument-independent. The following specific interchangeability benefits may be identified:

1. Because test procedures do not specify instrument control operations, changes in **software interfaces** between the Test Procedures and the Bus Drivers and in the **hardware interface** between the Bus Driver and the Instrument do not impact TPS operation (Figure 1). When instruments are replaced, the signal-based test development and execution environment is able to compensate for differences in the above interfaces.

2. Because test procedures do not contain references to instrument ports, changes in the **electrical interfaces** between Instruments and the UUT (Figure 1) do not impact TPS operation. Connectivity differences in the above interfaces are compensated by automatic switching. Moreover, automated switching enables the automatic **calibration** of signal paths, if adequate capability information is provided for the Switching Matrix and the Fixture.

3. The differences in **instrument behavior** that may be expressed in terms of range, resolution and precision capabilities are automatically taken into account by automatic resource allocation. Only instruments that satisfy the requirements expressed in the test procedure code are used at run-time.

Instrument Control in Signal-Based ATSs

Currently the signal-based testing paradigm is available through the ATLAS language [6]. Originally developed as a language for specifying test requirements for human readers, ATLAS has evolved into a programming language. Because the ATLAS standard does not specify how the actual control of instruments is implemented, vendor-specific solutions were developed. Due to the limitations of some of these solutions, existing ATLAS TPSs often contain instrument control operations, implemented by Non-Atlas Modules (NAMs). This approach compromises the instrument independence provided by the language.

To avoid the above problem, signal-based ATSs must include code modules (called in the following “signal drivers”) that implement the control of instruments through string command or driver calls. The “signal drivers” are instrument-specific, being replaced along with the instrument. To

support the flexible allocation of instruments to signals, the “signal drivers” must have software interfaces that are both instrument model-independent and instrument-class independent. This is achieved by implementing the basic signal operations corresponding to ATLAS single-action verbs, as exemplified below:

1. For the “source” and “load” signal roles: `Connect`, `Setup`, `Change`, `Close`, `EnableEvent`, `DisableEvent`, `Open`, `Reset`, `Disconnect`.

2. For the “sensor” signal role: `Setup`, `Connect`, `Change`, `Arm`, `EnableEvent`, `DisableEvent`, `Fetch`, `Disconnect`, `Reset`.

Some of the above considerations also apply for the control of the Switching Matrix, which should be implemented by device-specific “switching drivers”. The standardization of an interface for “switching drivers” is outside the scope of the present paper.

Architecture of Signal-Based ATSs

Signal-based ATS architectures (Figure 4) including the code modules introduced above are currently implemented in ATLAS products such as TYX PAWS. Similar architectures based on general-purpose programming languages were prototyped by the DoD joint service Automatic Test Systems Research & Development Integrated Product Team (ARI) [2] and TYX [10].

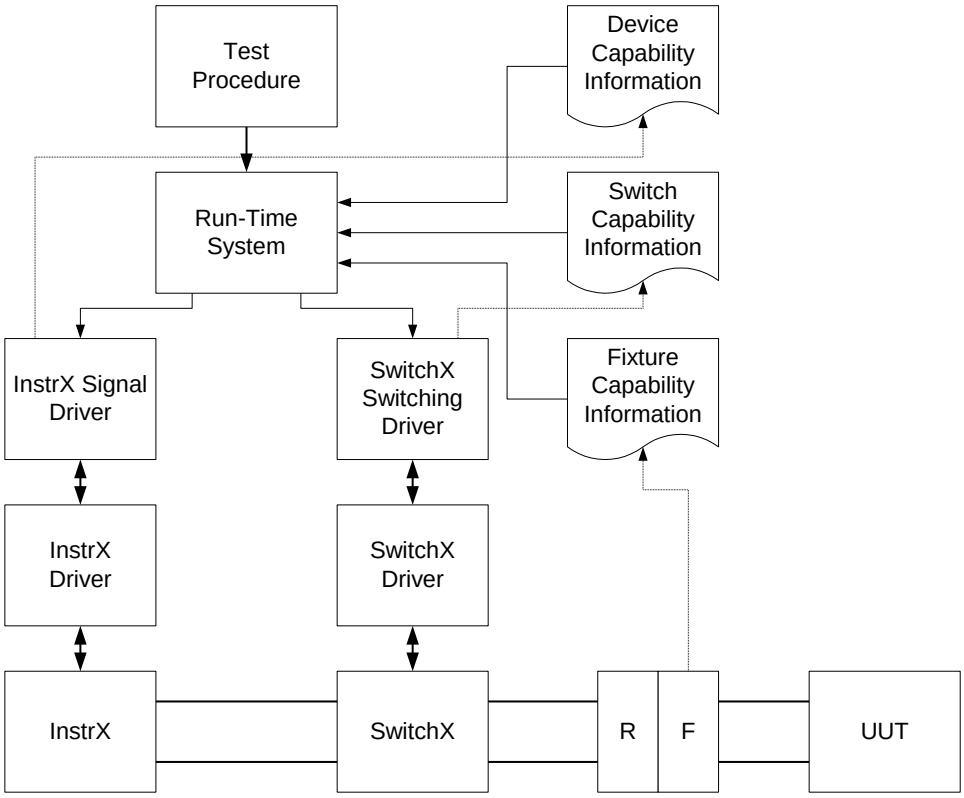

Figure 4. Signal-based ATS Architecture

The IVI-MSS Signal Interface

The lack of standardization for the interfaces of “signal drivers” supported by vendor-specific solutions limits their interoperability and their portability among different testing environments.

The signal-based ATS architecture shown in Figure 4 is perfectly compatible with the IVI-MSS architecture represented in Figure 2, if the “signal drivers” are implemented as IVI-MSS Role Components. Moreover, the use of IVI-MSS Aggregation Components with a “role” interface

may support multi-instrument signal measurement and generation solutions.

Consequently, the standardization of a **Signal Interface** for IVI-MSS Components allows the development of ATSs that combine the benefits of the IVI-MSS architecture and those provided by the signal-based testing paradigm. This combination is able to address all the effects of instrument replacement previously identified in the paper.

The IVI Foundation sponsors this approach through a Signal Interface Working Group, with the mission of "defining a standard specification for IVI-MSS component interface semantics in support of the signal-oriented description of measurement and stimulus operations".

As indicated before for IVI-MSS "role" interfaces, the **simplicity** of the Signal Interface (i.e., the low number of methods) simplifies performance verification, reducing development and maintenance costs. On the other hand, the **generality** of the interface (i.e., instrument-independence and application domain-independence) favors the reuse and portability of components among test environments from different vendors. The simplicity and the generality do not compromise the **capability** of the interface, understood as its ability to satisfy functional requirements for different types of testing (e.g., analog, digital, bus testing) and different types of UUTs (e.g., RF, opto-electronic, electro-mechanical, etc.), in test systems with different levels of complexity. The above capability was demonstrated over the years by the use of "signal drivers" in vendor-specific ATS architectures integrated in a large number of applications.

The unique combination of benefits enumerated before derives from the use of the signal abstraction, which is a very powerful and generic way of specifying behavior in test systems.

Functional Requirements for the IVI-MSS Signal Interface

This section presents the functional requirements currently identified for the Signal Interface design.

General Requirements

Signal Interfaces may be defined for IVI-MSS Role Control Modules or, in the case of multi-instrument solutions, for IVI-MSS Measurement and Stimulus Subsystems. In the following, the IVI-MSS Components with a Signal Interface will be called "Signal Components".

The Signal Interface standard must provide **"robust" instrument interchangeability** by:

1. Defining an instrument model-independent and instrument-class independent interface.

2. Defining an architectural component containing user-developed code that is able to compensate for differences in instrument behavior.

3. Supporting manual and automatic resource allocation.

4. Supporting automatic switching.

The Signal Interface standard provides **portability** of Signal Components among different test environments by:

1. Supporting the use of Signal Components in both signal-based and instrument-based test systems. Consequently, the Signal Components must be able to operate independently of any resource allocation or automatic switching services.

2. Supporting the use of Signal Components from multiple programming languages and development environments, including the ATLAS language, as well as all languages and environments supporting COM. The standard should also provide support for the emerging ATLAS 2000 standard, which is based on COM and general-purpose programming languages [4].

3. Not enforcing the use of a specific instrument control method. Signal Components may use IVI Drivers, VXI*plug&play* drivers, string commands or any other approach.

4. Supporting a business model that stimulates the development of value-added components.

Requirements for Signal Operations

As indicated before, the Signal Interface contains methods for implementing basic **signal operations**, as defined in the signal-based testing paradigm. Different sets of methods are generally required for different signal roles.

To cover commonly required testing functionality, the standard must support the signal roles of source, sensor and load. Additional signal roles may be needed for timing & synchronization, as well as for digital and bus testing. Because each role implies slightly different operations, distinct **signal role-specific** interfaces will be defined.

The Signal Interfaces must be **signal-type independent**. For example, the same set of interface methods should be able to handle the sensing of an AC Signal, a DC Signal or a Temperature signal. This greatly simplifies the standardization process; instead of specifying different interfaces for tenths of signal types, only a few specifications are needed for the different signal roles. Moreover, the above approach allows the delegation of signal type definition to other standards. This idea will be expanded in a subsequent section.

Requirements for Device Description

As shown before, the resource allocation functionality requires the ATS to support the description of **signal capabilities** for the available devices (i.e., signal roles and types; range, resolution and precision for signal parameters). In the following, the term “device” will be used for any hardware, software or combined asset able to provide signal generation or measurement functionality.

Since Signal Components may implement signal processing functionality, the signal capabilities are in general provided by the instrument-Signal Component ensemble. Moreover, these capabilities are guaranteed by the Signal Component vendor. Consequently, from both functional and business model standpoints the specification of capabilities belongs together with the Signal Component. To support this approach, the Signal Interface standard must specify a mechanism for the **formal specification** of device capabilities.

The standardization of device capability description includes the following aspects:

1. **Storage and access to device capability information**. The capability information must be stored persistently in a way that allows its distribution along with the Signal Component and preserves a permanent association with the Signal Component when this component is installed. The capability information is used by the ATS designer when selecting instruments and by the ATS software when performing resource allocation. Consequently, both human readability and programmatic access are required.

2. **Capability modeling**. The modeling of signal capabilities is a non-trivial issue, due to the complex ways in which signal functionality may be provided by instrument subsystems. This issue is detailed in the following subsection of the paper. Capability modeling is addressed by the TEDL standard. Because this standard references the ATLAS language, its use would contradict the requirement of a language-independent Signal Interface standard. Consequently, the development of a Signal Interface standard appears to require the design of a capability model, possibly based on TEDL.

Other Functional Requirements

The requirements presented below allow the Signal Interface to support testing functionality that is commonly required in applications:

1. The standard must support flexible signal implementations, as follows:

    1.1. Devices implemented by hardware (i.e., instruments), software or a software-hardware combination.

    1.2. Multi-channel instruments, where channels are able to concurrently measure and generate signals.

    1.3. Measurement or generation of a single signal using multiple instruments and/or instrument subsystems.

    1.4. Multiple concurrent functionalities provided by a single device subsystem (e.g., a power supply channel that is able to

generate voltage and current and to measure current).

2. The standard must support signal timing and synchronization, including the following capabilities: measurement of time intervals; synchronization of signal operations with events in other signals, with time and as simultaneity of signal operations. The standard must allow both hardware and software implementations for timing and synchronization. The implementation approach should be transparent to test procedures.

3. The standard must support digital testing and bus testing.

4. The Signal Components must be able to operate in simulation mode.

Standardization Issues for the IVI-MSS Signal Interface

To provide portability and reuse for Signal Components, both component developers and component users are required to use consistent signal type information. This information refers to signal type names and signal parameters, in turn characterized by name, data type and physical significance.

The Signal Interface standard will not attempt to specify signal types. This endeavor is considered outside the scope of the IVI Foundation, which is primarily oriented towards instrument control.

Definitions for signal types are currently available in ATLAS standards. A novel approach that provides increased extensibility through the use of the Signal and Method Modeling Language (SMML) is currently pursued by the IEEE SCC20 Test Description Subcommittee, as a part of the standardization process for the ATLAS 2000 language [4].

The Signal Interface standard may specify a formal mechanism for deriving signal type information from the above standards.

Integration of Signal Components in Automatic Test Systems

Signal Components deliver their full potential in **signal-based ATSs**, in conjunction with automated resource allocation and automatic switching. Although currently available through ATLAS, signal-based testing is not restricted to ATLAS. As shown by recent prototyping work [2] [10], signal-based testing may be implemented using a general-purpose object-oriented programming language together with a signal library, containing classes or components corresponding to specific signal types. The component library approach is also embraced by the ATLAS 2000 standard [4].

Instrument-based ATSs are also expected to benefit from the use of Signal Components. Besides the inherent advantages of IVI-MSS, this approach offers a very simple, capable and generic software interface. Moreover, the availability of a formal description for signal capabilities simplifies the selection of instruments.

The Signal Interface offers a natural programmatic gateway to **synthetic instruments**, since a unique Signal Component is able to provide access to the entire functionality of the instrument, by supporting all the signal types that may be measured and generated by the hardware.

The Signal Interface represents a possible implementation for the Resource Adapter Interface defined in the DoD JTA [3].

Business Model

The significant amount of development work required for Signal Components will probably rule out their free distribution by instrument vendors, as is the case with instrument drivers. Solution providers will then develop and sell Signal Components, as value-added software.

The business model promoted by IVI-MSS assigns clear responsibilities for development, testing and verification. The developers of IVI-MSS Components are required to guarantee their behavior and performance. The Signal Interface standard supports this approach by specifying the interface semantics and by requiring a formal description of device capabilities.

The users of Signal Components (end users or system integrators) will be able to build test

systems by selecting instruments for which Signal Components are available and whose capabilities match TPS requirements. In signal-based ATSs, the above matching is verified automatically.

When instruments are replaced, the expertise of the original Signal Component developer is no longer a critical issue, because the Signal Interface semantics are standardized and the capabilities of the original Signal Components are formally described. These specifications allow a different solution provider to develop Signal Components for the new instruments and to certify their performance. This certification along with automatic resource allocation guarantees identical test results.

Because Signal Components are instrument-specific, the Signal Interface standard is able to generate a market with a significant potential. With new generations of instruments emerging, new Signal Components need to be developed permanently, for both for new applications and for replacement purposes.

The business model described above is currently operational for signal-based ATSs used in military, aviation, nuclear plant and transportation applications. The standardization of the Signal Interface is expected to support solution providers by allowing them to deliver Signal Components to multiple end-users. This, in turn, is expected to stimulate competition and increase quality.

Conclusion

As demonstrated by the comparative analysis of existing solutions, different technologies provide different **degrees of interchangeability**. Because higher interchangeability generally comes with a higher cost, different categories of users are interested in different interchangeability levels. In application domains where the UUT lifetime covers several generations of test equipment, “robust” interchangeability is required. This feature allows instruments to be replaced without changes in the test procedure code or the fixture hardware, while identical test results may be guaranteed.

The standardization of a Signal Interface for IVI-MSS Components is able to provide such “robust” interchangeability, along with portability between different test environments. Signal Components may be used in both signal-based and instrument-based test systems. They are able to support a new generation of signal-based ATSs using general-purpose programming languages.

The design of a capable Signal Interface presents several challenges. The first one concerns the device capability model, which must support a flexible assignment of signal functionality to instrument subsystems. Another design problem is the support for hardware and software signal timing and synchronization that is transparent to the test procedure. A third issue concerns the transmission of signal parameters in a generic, signal type-independent format.

The support provided by the IVI-MSS framework and the IVI Common Components along with the experience accumulated by signal-based ATS designers have the potential to create a solution that supports the functional requirements for a large class of applications, while using state-of-the-art software technologies.

Glossary

ATS - Automatic Test System

COTS - Commercial Off The Shelf

Device - In the present paper, any hardware, software or combined asset able to provide signal generation or measurement functionality.

Instrument Interchangeability - The ability to replace a given instrument with an alternative instrument of sufficient capability, but of a different model, class, design or manufacturer.

IV&V - Independent Validation and Verification

IVI-MSS Aggregation Component - An IVI COM Component that aggregates the functionality of multiple instruments.

IVI-MSS Measurement and Stimulus Subsystem - An IVI COM Component that presents its API to application programmers and may aggregate the functionality of multiple instruments.

IVI-MSS Role Component - An IVI COM Component that follows the ownership rules specified for the IVI-MSS “role” interfaces.

"Robust" Instrument Interchangeability - The ability to replace instruments without changing the TPSs, while the test results are guaranteed to remain identical.

SCPI - Standard Commands for Programmable Instrumentation

Signal Component - IVI-MSS Component with a Signal Interface

"Signal Driver" - In the present paper, an instrument-specific software module with a signal interface; the term is used for vendor-specific ATS implementations; the equivalent term used within the IVI-MSS framework is "Signal Component"

Signal Interface - IVI-MSS "role" interface with standardized semantics, implementing the basic signal operations as defined by the signal-based testing paradigm

Signal Parameter - An attribute of the signal indicating the value corresponding to a characteristic of the physical signal

Signal Role - An attribute of the signal specifying the type of operation it performs on the (e.g. source, sensor, load).

Signal Type - An attribute of the signal specifying the type of physical signal it models (e.g. AC Signal, DC Signal, Temperature, Time, etc.).

TEDL - Test Equipment Description Language

TPS - Test Program Set. Contains the test procedures and the fixture hardware required to test a given UUT.

UUT - Unit Under Test

References

[1] Barnholt, N., "Connecting You to the Future", Keynote address, AUTOTESTCON 1998, http://www.tm.agilent.com/tmo/techinfo/English/ned_autotestcon.html

[10] Tyler, D., "Java-Based Automated Test Systems: Management Considerations for an Open Architecture for Test", *Proc. AUTOTESTCON*, San Antonio, TX, 1999, pp. 699-706

[2] Department of Defense, Automated Test System Research and Development Integrated Product Team (ARI), *FY98 Status Report*, 1998

[3] Department Of Defense, *Joint Technical Architecture*, version 3.0, November 1999, http://www-jta.itsi.disa.mil/

[4] IEEE SCC20 Test Description Subcommittee, http://grouper.ieee.org/groups/scc20/atlas/index.html

[5] IEEE Standard for Test Equipment Description Language (TEDL), IEEE Std 993-1997

[6] Standard Test Language for All Systems - Common/Abbreviated Test Language for All Systems (ATLAS), IEEE Std 716-1995

[7] IVI Foundation, http://www.ivifoundation.org

[8] Oblad, R., "Applying New Software Technologies to Solve Key System Integration Issues", *Proc. AUTOTESTCON*, Piscataway, NJ, 1997, pp. 181-189

[9] Oblad, R., "Achieving Robust Interchangeability of Test Assets in ATE Systems", *Proc. AUTOTESTCON*, San Antonio, TX, 1999, pp. 687-698