An Open Source Software Framework for the Implementation of an Open Systems Architecture, Run-Time System

Matt Cornish
EADS TES (UK) Ltd
matt.cornish@eads-ts.com

Anand Jain
National Instruments Corporation
Anand.jain@ni.com

Malcolm Brown
DES JSC SCM-EngTLS-TM-ATS, MoD
DESJSCSCM-EngTLS-TM-ATS@mod.uk

Teresa Lopes
Teradyne Inc.
Teresa.lopes@teradyne.com

Abstract — This paper presents the outcome of a UK MoD sponsored development effort to provide a suite of source code that will be made available to contractors employed in the provision of test system software to the MoD and coalition partners. The primary purpose of this ‘open source software’ is to provide a working test system software framework that meets the requirements of the MoD's DEFSTAN 66-31 [1] (Open Systems Architecture); in particular, the use of IEEE 1641 [2] and ATML [3].

Using the interfaces and data exchange formats defined by both IEEE 1641 and ATML, a software framework has been written to bring together COTS tools and test information, in an application that sees ATML Test Description through to UUT test pin. Specifically, the framework is broken down into the areas of:

- **ATML Test Description Importer** – Converting test requirements into a test program implementation carrying 1641 Test Procedure Language.
- **1641 Signal Translator** – Mapping test signal requirements onto test resource capabilities (making use of ATML Test Station Description).
- **Signal Routing** – Connecting test resources to UUT pins.
- **1641 Test Signal Framework IDL Generator** – Generating a run-time interface from 1641 signal libraries.
- **1641 Run-time** – Implementing a 1641 runtime interface with calls to underlying test resources.

COTS tools have been chosen from three different manufacturers, encompassing test program generation, test signal allocation and switch path routing.

This project is known as the Open Systems Architecture Runtime System (OSA RTS).

Keywords - Open Systems Architecture; IEEE 1641; IEEE ATML; Signal & Test Definition.

I. INTRODUCTION

The standards demonstration programme [4][5][6][7] has shown that solutions supporting the MoD's DEFSTAN 66-31 [1] Open System Architecture (OSA) – particularly the use of IEEE ATML [3] and IEEE 1641 [2] – offer significant through-life cost savings. The programme has produced a wealth of evidence supporting the use of 1641 and ATML. Their implementations have been varied in software architecture for both TPS & ATS; several different vendors’ products have been included and a variety of different hardware platforms have been utilised.

Recent internal MoD studies into ‘demonstrated coverage among ATS of IEEE 1641™ & ATML’ have observed that a number of COTS tools have now become available, supporting these standards. Furthermore, these studies concluded that a system could be built from COTS tools that would bring 1641 & ATML to existing ATS Architectures, whilst making use of the investment already made in run-time tools, configuration data and training.

A. Open System Architecture Compliant System

An Open System Architecture Compliant System is one that conforms to IEEE Std 1641 and additionally provides test information (e.g. UUT, Instrument, Test Adaptor and Test station information, including capability) using the ATML standards. Additionally, an Open System Architecture Compliant System supports TPS import and test results export using ATML standards.

B. IEEE Std. 1641 Compliant System

An IEEE Std 1641 Compliant System is one that supports TPS developed by one or more of the following methods:

- A TPS written in a COTS programming language adhering to the Carrier Language Requirements (Annex G) and using the IDL interface (Annex D) to interface with the Signal and TSF methods defined in Annex B and Annex C.
A TPS written in a COTS programming language adhering to the Carrier Language Requirements (Annex G) and using the Test Procedure Language (Annex H).

A TPS described in IEEE Std. 1671.1 ATML Test Description (according to FTS3/DSA006 Task 16, an allowable variant of a ‘Carrier Language’) and using IEEE Std 1641 XML Signals definitions (Annex I).

In all cases, the TPS shall meet the requirements that:

All UUT behaviour is defined through interaction of IEEE Std 1641 BSC or TSF Signals on the UUT interface.

All TSF Libraries referenced by the TPS are provided.

The TPS’s programming language is agreed and available at acceptable cost.

II. COTS SOFTWARE

COTS tools were chosen from three manufacturers, each manufacturer being a participant in the standards’ development. The products have been developed to adhere to those standards, to promote interoperability.

A. Cassidian newWaveX®

newWaveX is IEEE 1641™ Signal-based test and measurement software, available to fulfil two specific functions:

1) newWaveX SD (Signal Development)
   A complete graphical signal modelling & simulation environment for automatic test, compliant with the following standards’ signal requirements:
   - IEEE 1641 Signal & Test Definition.
   - IEEE 1671.1 ATML Test Description.
   - IEEE 1671.2 ATML Instrument Description.
   - IEEE 1671.6 ATML Test Station.

   newWaveX SD supports the creation and editing of IEEE 1641 Signals & TSF libraries; XML, XSD, IDL and HTML file formats for storage, interface specification and documentation. Additionally, providing ActiveX controls, enabling easy embedding in third-party applications.

2) newWaveX PD (Platform Development)
   A test platform integration toolset, targeted at getting test signals to test pins, newWaveX PD fulfils two main functions:
   - A resource (instrument) description and validation environment for IEEE Std.1671 ATML Instrument and Test Station Description.
   - A compile-time resource manager/translator, reading signal orientated test program descriptions and outputting driver orientated test code, applicable to:
     - IEEE Std. 1641 IDL and TPL
     - IEEE Std. 1671 ATML Test Description.

B. National Instruments ATML Toolkit

NI provides an ATML Toolkit that supplements NI TestStand to import ATML Test Description and to export ATML Test Results. The Toolkit is not intended to handle IEEE 1641 Signals, instead providing an API (Application Programmer Interface) for users to build upon its capability; e.g. using Cassidian newWaveX.

C. National Instruments TestStand

NI TestStand comprises a sequencer and test executive used to assist in the development of automated test sequences, otherwise known as TPS. TestStand is both a development and execution environment in this respect.

In terms of an ability to exchange test definitions to support portability of tests, TestStand is generally considered to be a closed and proprietary test environment, though it has significant universal acceptance and ongoing support. From an OSA perspective, signal and sequence test code contained within the tool is accessible to the user and, therefore, the inclusion of this tool is recommended.

D. National Instruments LabWindows™/CVI

LabWindows/CVI is a proven ANSI C development environment for test and measurement that greatly increases the productivity of engineers and scientists. For more than 20 years, C developers have used LabWindows/CVI to develop high-performance, stable applications in the manufacturing test, military & aerospace, telecommunications, design validation and automotive industries. LabWindows/CVI streamlines development with hardware configuration assistants, built-in measurement libraries, comprehensive debugging tools, interactive execution capabilities that developers can use to run functions at design time) and advanced analysis & scientific user interface tools.

E. Teradyne Xpress Services

Xpress Services has been used as part of the ATML demonstrations, where it was a key component in describing ATE resources, capabilities and their interconnections. It was also used for providing path level analysis and switching to and from resources and UUT pins, as well as helping to identify suitable resource capabilities.

F. Microsoft Visual Studio .NET C# 2008

MS Visual Studio is an IDE and command line toolset for software development.

Pertaining to the OSA Runtime System, it provides C# coding, interface stub generation, compilation and IDL compilation to type library. I.e. not directly supporting standards, but providing the binding.

III. SOLUTION

The Open Systems Architecture Runtime System (OSA RTS) solution consists of open source software, together with COTS software tools provided by key participants in the development of the standards. The key elements of the solution are illustrated in Figure 1.
A. ATML Importer (Test Description).

The ATML Importer imports ATML Test Description instance documents. From this, it generates a TestStand sequence and CVI test action code. The CVI code contains TPL statements to represent 1641 Signals.

Selection of an ATML Test Description file is provided through the TestStand Environment, by the ATML Toolkit.

As an add-in to TestStand, the ATML Toolkit imports ATML Test Description, producing a TestStand Sequence and CVI test action DLL export.

A series of call-backs are produced by the ATML Toolkit when 1641 Signal actions are processed. 1641 Signal actions are not interpreted by the ATML Toolkit, so the open source software performs the interpretation and inserts 1641 TPL statements into the CVI DLL, during each call-back.

In this process, only the CVI DLL source code is altered by the Open Source Software, not the TestStand Sequence.

B. 1641 Signal Translator.

The 1641 Signal Translator processes C-style (e.g. CVI) code, prior to compilation, translating TPL statements into 1641 TSF calls to the 1641 Runtime.

The 1641 Signal Translator processes C-style (e.g. CVI) code, prior to compilation, translating TPL statements into 1641 TSF calls to the 1641 Runtime.
Open source software parses C code to extract TPL statements.

newWaveX-SD validates and converts any signals to 1641-2010 standard.

The 1641 Signal Translator uses newWaveX PD to compare TPL signals with ATML Test Station ‘capability’ signals, to find a match.

An instance file of (newWaveX) Capability Driver Description [8] provides implementation information for each capability in the ATML Test Station Description. newWaveX PD outputs code from Capability Driver Description in place of the input signals.

Thus, the 1641 Signal Translator reads TPL and writes native code for the target test station; in this case, code that calls the 1641 TSFs on the test station.

If duplicate resources are identified, (i.e. more than one that matches the specification), the user is prompted to make the decision.

A log is generated by the translation process, identifying any issues detected.

C. Signal Routing.

Signal Routing is divided into two parts: Conversion of TPL Connect statements to IVI Switch calls in the CVI test program; and an IVI Switch wrapper around Xpress Services, called by the converted TPL statements.

1) TPL to IVI Translator

To Convert TPL Connect/Disconnect statements to IVI Switch calls, the 1641 Signal Translator is extended to parse TPL Connect & Disconnect statements.

Open source software takes the instrument capability, allocated by the 1641 Signal Translator, and parses through ATML Test Station Description to locate the instrument’s ports, using newWaveX.

TPL Connect statements refer to the UUT pin; hence, an IVI Switch statement is formed using this pin name and the identified instrument port.

2) IVI Switch Wrapper

The Signal Routing component in the Runtime Environment implements an IVI Switch wrapper around Xpress Services.

To support the TestStand and CVI environment, utilised by the ATML Importer, the IVI Switch wrapper is written as open source software in CVI.

Note: There remains some initialisation of Xpress Services in TestStand, which is manually added. This is because Xpress Services is supplied as an add-in to TestStand.

D. 1641 TSF IDL Generator.

The 1641 TSF IDL Generator takes ATML Test Station Description and 1641 TSF libraries to create a test station 1641 IDL, from which a runtime callable type library can be built.

The 1641 TSF IDL Generator effectively combines the constituent system Libraries in an interface definition (IDL) file.

Open source software uses newWaveX PD to open, parse and analyse the ATML Test Station Description to collect a list of TSFs supported by the ATE.

These TSFs are then loaded into newWaveX Signal Development, which exports them as a IDL files.

The resultant IDL files are combined to produce a Test Station IDL that can be built into a runtime project.
E. Runtime Environment.

The Runtime Environment is divided into two parts: IDL is compiled to a type library and built into a 1641 Runtime component that is called by the test program; TestStand provides a Test Executive to execute test programs, with the ATML Toolkit generating ATML Test Results.

1) 1641 Runtime

The 1641 Runtime callable object is built in MS Visual Studio using the TSF interface definitions produced by the TSF IDL Generator.

IDL is compiled to a type library.

The Type library is imported into a Visual Studio C# project.

Visual Studio generates function and attribute code stubs.

Trace statements are added to stubs to provide for validation.

2) Test Executive and ATML Test Results

Operation and interaction between the open source software and COTS software is as follows:

TestStand sequence calls test actions in the now compiled CVI DLL.

CVI test actions call the 1641 Runtime for Signals and the Signal Routing for connections.

ATML Toolkit automatically generates ATML Test Results.

F. Alternative Carrier

In order to demonstrate the operation of the OSA RTS with an alternative test program carrier language, the source files developed as part of 1641 Demo 2.5 [7] were used (including a C# test program).

IV. SUITABILITY OF THE SELECTED COTS TOOLS

A. Cassidian newWaveX®

1) Signal Development

newWaveX Signal Development is well established in the development of IEEE 1641 Signals and TSFs. In particular, it is in active use with MoD.

Visual representation is made use of in this OSA RTS, to verify test signals.

Validation is comprehensive for both TSFs and test signals.

A particular feature of newWaveX SD is its ability to read all versions of 1641, up-issuing automatically to 1641-2010.

As well as XML representations (used by ATML Test Description) newWaveX SD provides translation to other formats. In particular, providing a validated route from XML to IDL, for producing a runtime test program interface.

2) Platform Development

newWaveX Platform Development provides validation of all XML documentation. It usefully encapsulates rules for locating XML schema documents (for validation) on a system and project basis – ensuring validation was carried out against the correct versions, on this multi-version OSA RTS.
The Resource Manager component provided the actual translation between the TPL test signal and the ATML Test Station Description resource used for the test. This includes assessing Physical capability ranges & tolerances.

The Resource Manager does not currently expand TSFs during translation.

newWaveX PD includes an ATML Test Station Description development environment, though this was not used.

B. National Instruments ATML Toolkit

This add-in to TestStand provided an additional file type (ATML Test Description) in the ‘open file’ dialog.

During import a call-back is created to a DLL, to allow processing of the 1641 Signal parts of the imported ATML Test Description. A CVI source code template is supplied with the Toolkit for the purpose of developing the DLL to handle the call-backs.

It is possible to use other languages to develop the call-back DLL, though time did not permit further investigation into this solution.

For the purposes of validation, the ATML Toolkit targets specific versions of ATML schema documents.

C. National Instruments TestStand

TestStand presented no issues and was used ‘as is’.

D. National Instruments LabView/CVI

Integration with the ATML Toolkit was everything it could be. Though, it was questioned whether a non object orientated carrier language is well suited to call an object orientated 1641 Runtime.

E. Teradyne Xpress Services

Xpress Services’ TestStand integration was considered very convenient to use, including the switch data configuration tools that this provides. However, the ability to import configuration directly from ATML files would greatly improve control and verification of this data.

Using CVI to call Xpress Services divided switch routing between TestStand and CVI, because the initialisation step is through TestStand and the connections are made through CVI (i.e. where the TPL statements are).

An additional step – through the Xpress Services .NET interface – might be integration into the 1641 Translator, where Xpress Services could be used to eliminate duplicate resource choices that cannot be connected.

V. CONCLUSIONS

This work has provided a suite of source code that will be made available to contractors employed in the provision of test system software to the MoD and coalition partners.

The primary output is ‘open source software’ that provides a working test system software framework to meet the requirements of the MoD's DEFSTAN 66-311 (Open Systems Architecture); in particular, the use of IEEE 1641 and ATML.

Using the interfaces and data exchange formats defined by both IEEE 1641 and ATML, a software framework has been written to bring together COTS tools and test information, in an application that sees ATML Test Description through to UUT test pin. Specifically, the framework is broken down into the areas of:

- ATML Test Description Importer – converting test requirements into a test program implementation carrying 1641 Test Procedure Language.
- 1641 Signal Translator – mapping test signal requirements onto test resource capabilities (making use of ATML Test Station Description).
- Signal Routing – connecting test resources to UUT pins.
- 1641 Test Signal Framework IDL Generator – generating a run-time interface from 1641 signal libraries.
- 1641 Run-time – implementing a 1641 runtime interface with calls to underlying test resources.

COTS tools from three different manufacturers have been used to fulfil the ATML, 1641 and signal routing requirements. Each of these manufacturers credentials are proven by their involvement in the standards’ development and their products have been developed to adhere to those standards, to ensure interoperability.

The open source software that binds these COTS tools together had been written in a mix of C# and C (CVI). Since the software language requirements of the MoD’s Open Systems Architecture have been adhered to, this open source software is readily useable and able to be built upon by a large group of developers and maintainers, now and in the future.

ACKNOWLEDGEMENT

The project wishes to thank NI and Teradyne for their support in the use of their various software tools.

VI. REFERENCES

[4] Implementing IEEE 1641 – A Demonstration of Portability, AutoTestCon 2005
[8] Implementing IEEE 1641 - Compilation Techniques (to IVI Driver Code), AutoTestCon 2009