Integration of Software Technologies into a Test System

Ron Yazma
Geotest- Marvin Test Systems, Inc.
Irvine, California
rony@geotestinc.com

***Abstract*— Test applications often require the integration of many different software technologies. This paper provides an overview of how several Windows-based technologies can be incorporated into a single application by employing a common software framework and architecture.**

Keywords— software framework; test applications; .NET; DLL; COM

I. INTRODUCTION

The incorporation of various software technologies and methods has become common place in today's functional test systems. Increasingly test engineers are employing multiple languages and test methods as part of a test program set (TPS). And when one includes the use of both textual and graphical programming languages, the breadth test methods grows even larger. Clearly, having a robust tool set that can accommodate and assist the developer in combining these various architectures, languages and methods can be a real benefit to the test developer.

Some specific examples listed below detail the benefits and needs for integrating specific software technologies or methods into an application.

- A specific instrument driver is available only in a particular technology. Re-using or integrating the driver into a new application requires using the driver's underlying technology.
- The application requires the use of external software components to provide additional functionality such as database management for storing of test results or test requirements, the creation of user interfaces (i.e., ActiveX controls), spreadsheet support, test log generation (HTML/XML), support for data analysis libraries, or support of communication protocols.
- Use of existing or legacy code which needs to be integrated into a current or new application.

The ability to integrate these various technologies into applications maximizes code reuse and allows faster completion of the test system application. And by creating an intuitive software environment, coding can be minimized making the integration of these technologies easy and effective.

Using a common software framework and architecture, several key software technologies can be incorporated into a single application. The technologies and integration methods addressed in this paper include:

- DLL and header file usage using a common framework
- COM ActiveX components including the use of an Excel spreadsheet, a data base and an ActiveX control (Microsoft Calendar) application
- .NET assemblies and classes including methods and properties
- Function Panel (.fp) drivers and the use of IVI-C or IVI-COM drivers.

II. INTEGRATING DLLS

DLLs (Dynamic Link Libraries) are the building blocks of Windows. In a test application, DLLs are used mostly as instrument drivers for PC and PXI-based instruments. They are also used to extend the capability of the test framework by calling a Windows API or other software library to extend the test framework's functionality. Many issues can be encountered when calling external libraries and the framework must support them. Specifically, the framework needs to support the following attributes and functionality of a DLL:

- Calling conventions (i.e. stdcall or cdecl)
- Parameter type support and support for simple data types (integer, floating point, string, Unicode/BString)
- Support for structures and packing, objects, procedures (callback) by reference or by value and pointers
- Return type support

The process of declaring the functions and data types in the test application can be tedious. The code will generate many errors if the declarations are not accurate due to the manual declaration process. Support for automatic importing of C header files is therefore a major benefit that will save significant programming and debugging efforts. Figure 1 details how the importing tool can be configured to support the importing of the DLL as well as the C header file.

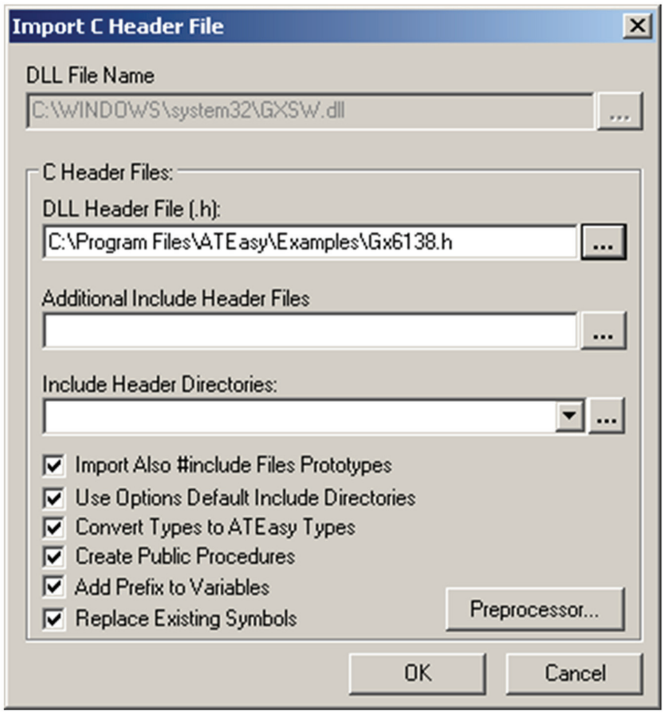

Figure 1: DLL File Import

Additionally, the importing of DLL files can result in the identification of C header files where the data types are ambiguous. Figure 2 details how the importing tool will flag these ambiguous data types – simplifying the overall DLL importing process.

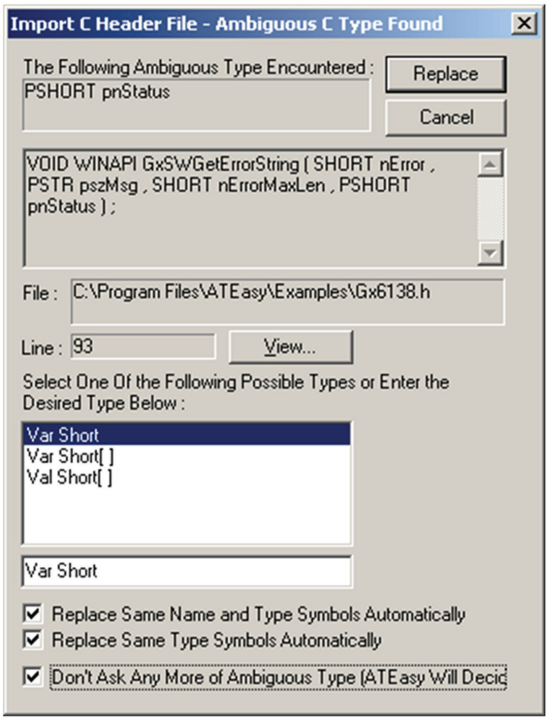

Figure 2: Ambiguous Data Types

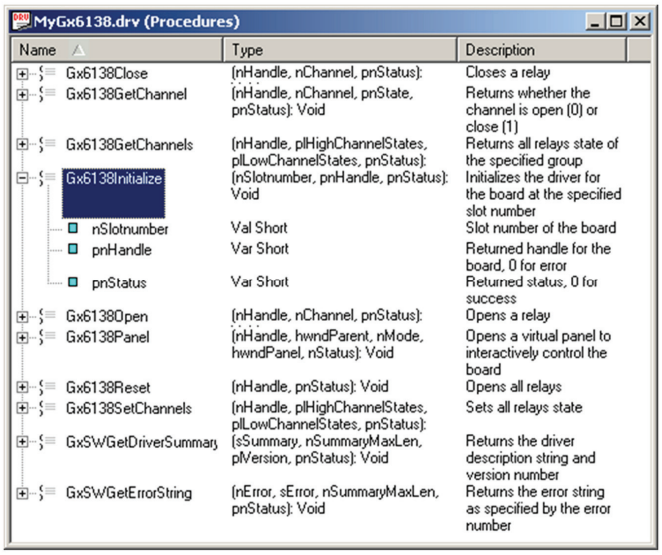

Figure 3: Completed DLL Import

After completion of the importing process, the resulting file or procedure yields a set of function calls with descriptions (Figure 3) which are ready to be used with the application.

III. Integrating COM/ActiveX Components

Microsoft's Component Object Module (COM) and Microsoft's ActiveX controls are popular Windows objects. COM components include a type library that describes the library classes, methods, properties, and events exported by the components. Once the type library imports the files, a framework can display the contents of the library in a browser, allowing the user to browse and see the appropriate classes that need to be created and the methods and properties that need to be called or set. Code completion can also accelerate the coding process by offering a list of methods and properties that can be used when the object is selected for inclusion in the test program.

ActiveX controls are a superset of the COM module, which describe and provide user interface controls that can be placed on a user interface (UI) form – extending a UI's functionality. Once the control library is imported, these controls are added to the framework's control palette. The user now has access to these controls that are similar to the framework's built-in controls. These controls offer enhanced functionality such as Tree View, calendar control, specialized chart, etc.

Figure 4 details how the framework's tools can be used to insert a Type Library (in this case the Microsoft ActiveX Data Object library) which then provides data base read/write capability.

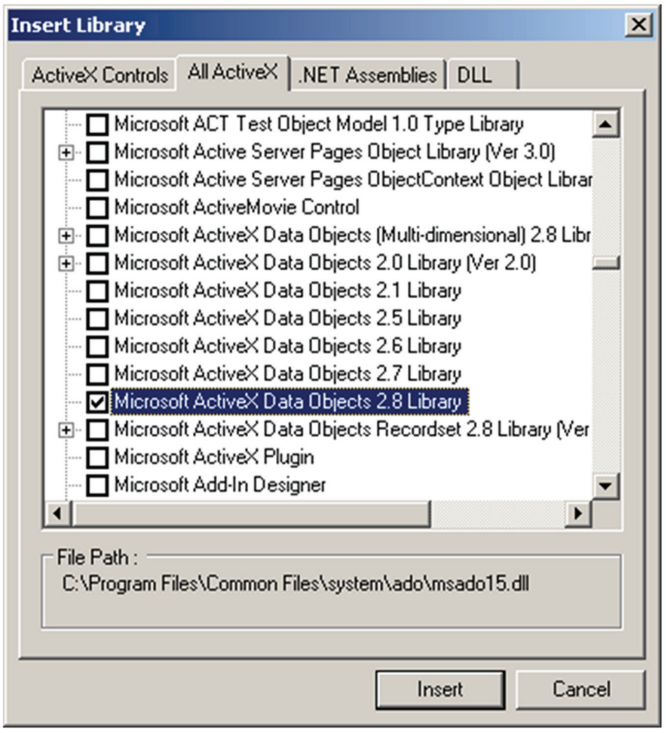

Figure 4: Importing an ActiveX Library

Once the ActiveX library has been imported, the library is available for use as part of a test application with all functions and types exposed for use by the test programmer (see Figure 5).

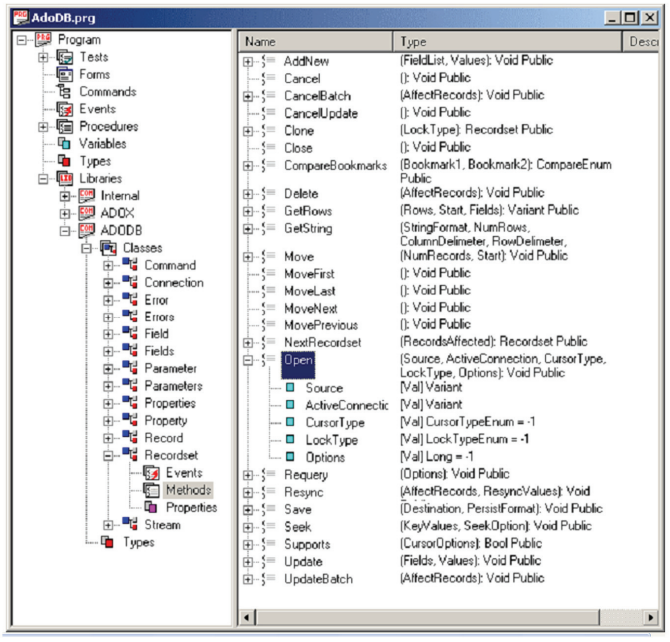

Figure 5: Browser with imported ActiveX Library

IV. Integrating .NET Assemblies

Similar to COM, .NET is another method for accessing software components written with Microsoft Studio .NET. .NET components are more complex and feature-rich than COM with .NET supporting inheritance, parameter constructors, overloading, and static virtual members. The Microsoft .NET framework offers a wider set of classes that are compatible with any Windows OS that supports the .NET framework. The importing method for .NET components is similar to the techniques and methods used for importing COM objects. Figure 6 details the results of importing a .NET library. Note that the framework presents all functions and types to the user via the software framework's browser.

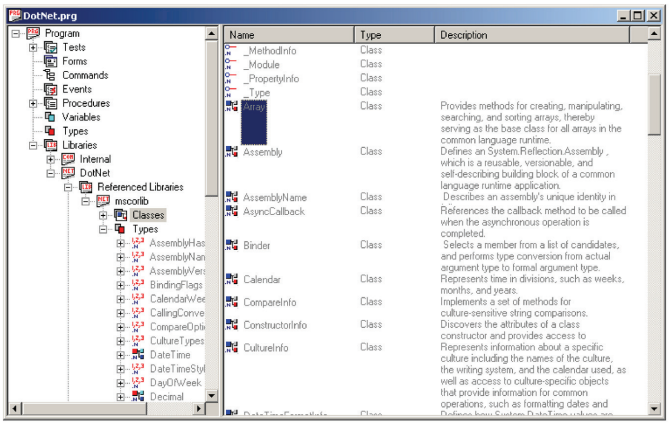

Figure 6: Browser with imported .NET components

V. Importing and Using Function Panel Drivers

Function Panel drivers are based on the VXI Plug & Play driver specification which was developed by the VXI System Alliance (http://www.vxipnp.org) and is now maintained by the IVI Foundation (http://www.ivifoundaton.org). Central to the VXI Plug & Play specification is the VISA (Virtual Instrumentation Software Architecture) I/O library specification which is also maintained by the IVI organization. The Function Panel is comprised of instrument driver functions and uses the VISA interface to communicate with the instrument. The instrument driver is packaged as a DLL and an .fp file that describes the DLL commands, functions, and documentation. The software framework's importing tools can import the .fp files and automatically will create a driver that that is readily useable by the framework's development environment. Imported driver components include commands, data types, and documentation.

Figure 7 shows the results of importing a PnP driver for the HP34401 DMM.

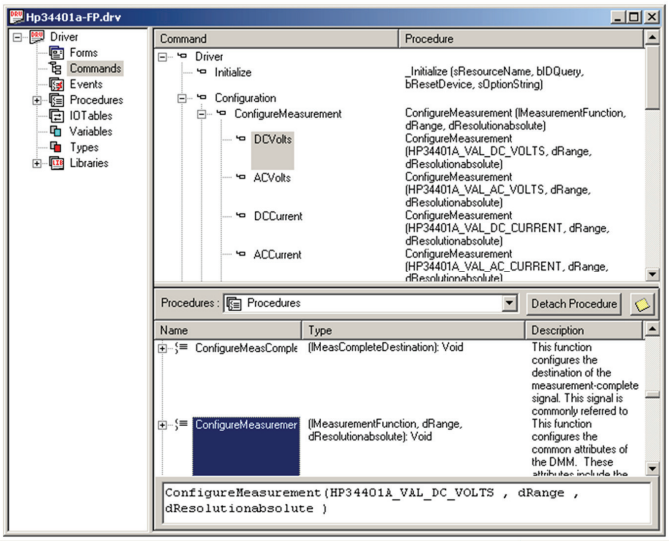

Figure 7: Imported .fp Driver

The imported drivers offer many features not available with the original function panel drivers. Features such as automatic initialization, error and exception handling, strong type declaration using enumerated types instead of constants, and code completion are all available as part of the imported driver.

VI. Importing and Using IVI Drivers

The IVI standard for instrument drivers was created by the IVI Foundation (http://www.ivifoundation.org). The foundation has defined generic, interchangeable programming interfaces for common instrument classes. Currently, the following IVI drivers or classes are defined: DC power supply, Digital multimeter, Function generator, Oscilloscope, Power meter, RF signal generator, Spectrum analyzer and Switch. IVI drivers are based on VXI Plug & Play drivers and require VISA and IVI libraries prior to installing the drivers. By using an IVI class driver, instrument interchangeability can be realized, since the driver offers the same functions and parameters regardless of the specific instrument being controlled. For example, IVI allows you to replace an Agilent 34401 DMM with a Keithley 2000 DMM in your system without changing your code. Similar to the other software technologies discussed earlier, the software framework provides the necessary tools to support the importing and use of IVI drivers as part of a complete test application (Figure 8).

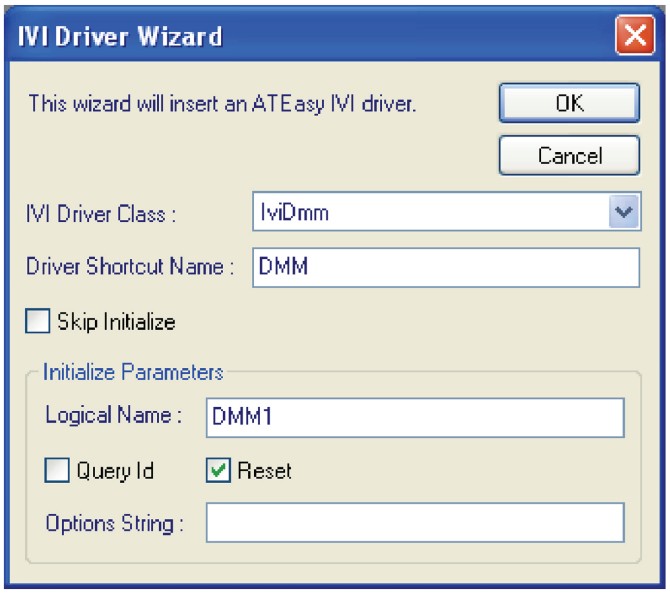

Figure 8: Importing an IVI Driver

The imported IVI driver offers many features not available in the original IVI drivers, including automatic initialization, error and exception handling, strong type declaration using enumerated types instead of constants, and code completion.

Summary

Many of the Windows based technologies discussed in this paper can require extensive programming knowledge and expertise to successfully integrate them into an end application. However, by employing a software framework that supports these technologies, the complexity and time required to employ these technologies and methods can be minimized. With the availability of wizards and automated importing tools, users can readily take advantage of the latest software technologies as well as benefit from a framework that offers features such as code completion, in-place documentation, automatic initialization, error and exception handling, and strong type declaration.

References

http://www.ivifoundation.org

http://vxipnp.org