TestStand/LabVIEW as a Reusable Framework for Automated Medical Device Testing Systems

by

Marcela Maldonado
Consultant, Measurement and Automation
Data Science Automation
USA

And

Marisette Edwards
Staff Software Engineer
Medrad, Inc.
USA

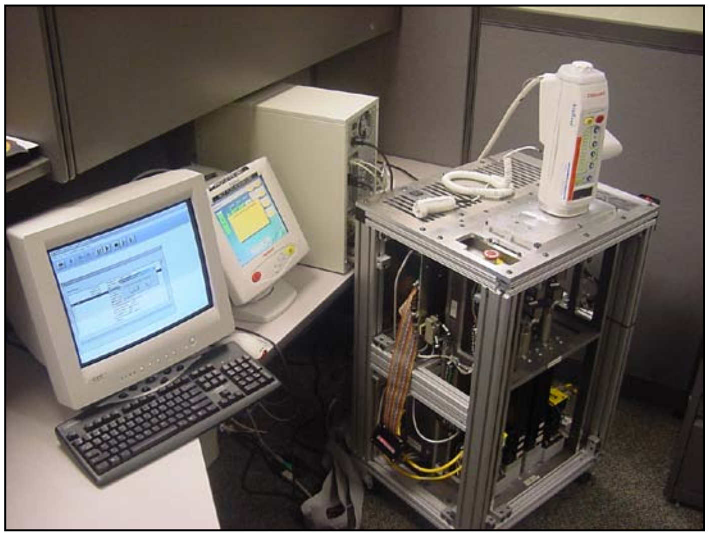
Category
Manufacturing Functional Test

Products Used
NI TestStand 2.0.1
NI LabVIEW 6.1
NI PCI-MIO-16XE-50

The Challenge
Replace an aging C-based test executive that is a maintenance nightmare and does not integrate with other applications or devices easily.

The Solution
Build a customized, reusable TestStand/LabVIEW Framework that meets medical device functional testing and FDA requirements.

Introduction
Medrad, Inc. is a manufacturer of medical devices. The Manufacturing Test Development group is responsible for functional testing of various vascular injection systems. The injectors are complex electromechanical devices that are used to inject contrast (a solution that will improve the contrast of the image) intravenously into patients undergoing procedures such as CT and MR scans and angiographs. The test group must develop test systems to verify that a newly manufactured injector is functional and safe. Records must be kept in compliance with FDA regulations. New products are constantly being developed and the manufacturing team must be prepared to test each new product as it is launched.



Data Science Automation, NI Select Integration Partner, was selected as the lead architect for this developmental effort based on significant TestStand experience and NI tools versatility. The project team included DSA and Medrad personnel working in concert and under tight deadlines.

Medrad's experienced production operators expect to start up an automated test by double clicking on an icon on their Windows desktop. This should start up the automated test by displaying a user interface that guides them through the proper steps to complete the test. Startup of the program involves identifying the operator and whether he has production-only or troubleshooting privilege, initializing the fixture, performing fixture self-tests if needed, and if the operator has troubleshooting privilege allowing the operator to select which mode to run in.

From that point on, a Unit Under Test (UUT) is identified and tests are run in sequence on that UUT. For each UUT tested, a traveler (document that travels with the tested unit) is prepared and rework information is requested and stored if applicable. A log is created for debugging purposes. Test results are written to a Microsoft Access database. When a UUT fails a test, it must be reworked and retested, and as part of its rework it may be tested in troubleshooting mode to repeat only the test(s) that failed.

Test systems performing as above, previous to the Framework under discussion, were written in C and based on a test executive product no longer supported by its manufacturer. Its source code was modified over and over throughout a decade of building test fixtures. Recent injectors were being developed with single board computers that enabled communication between the test PC and the injector using the Component Object Model (COM) protocol. The old test system design was not compliant with COM, and its maintenance became too cumbersome and time-consuming.

A new testing system was needed that would:

- support modularity, such that a basic framework could be used as a starting point and customizations added in a modular fashion instead of actually modifying the framework itself;
- support modern communication between different types of software, including databases, using COM;
- integrate easily with the UUT's software internal function testing package, including handling events generated by the UUT;
- support future complete automation of product tests;
- provide a consistent, reliable, easy to use operator interface;
- have technical support readily available;
- be maintainable and easily customizable.

After the options were considered, National Instruments rose to the top as the company that understood the testing environment and had available several software packages that supported our needs. National Instruments is a large well-established company with excellent technical support options and a large library of virtual instrument modules available. Many interfaces that the former system required us to write from scratch are now available for instant integration and use.

The Challenges Met

Although the sequential model roughly met the needs of the test system, the order of steps required was different from any existing entry point within the process model. We created the "Medrad single pass" entry point to support the following features:

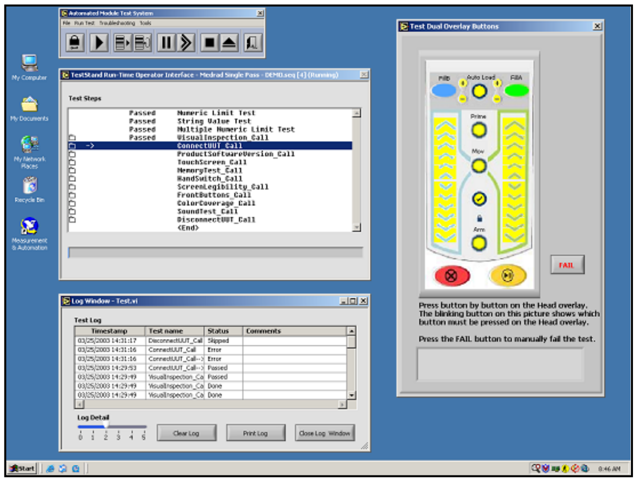
- enter and validate multiple serial numbers for one test (where two or more parts are tested as one module);
- prompt for a fixture serial number;
- create a customized traveler indicating the results of the test;
- customize the database schema (see further discussion below);
- handle events generated by the UUT;
- provide additional callbacks for tests that we expect every system to run, but which may be different for each type of module to be tested;

- run in production or troubleshooting mode.

In addition, we developed a completely customized operator interface with LabVIEW. The operator interface provides the operator with a simple and standard tool that supplies all of the functionality needed for him to do his job without confusing additional choices. The number of choices available depends upon the mode selected (production or troubleshooting), and the privileges assigned to the user. The production mode simply guides the user through the test steps required for a UUT. The troubleshooting mode allows the user to select steps to run, to loop selected tests, and to break at steps. The sequence editor was considered for this mode, but there were too many choices available that would be confusing to an operator who just wants to find out what is wrong with the unit he is troubleshooting.

One of the biggest challenges was the results database. Saving multiple serial numbers sounds easy, but no one in the NI community had an answer. Instead of just configuring a schema, we create the schema dynamically, allowing any number of serial numbers to be used. This allows the test developer to configure the number of serial numbers consistent with the parts combined into a unit to be tested. Complying with the FDA CFR 21 Part 11 rule was multi-faceted. Result data had to be moved from the local PC to the network, not copied, because only one copy of electronic records may exist. This required transaction processing and providing for rollbacks in case a failure occurred during the transaction. In addition, since the traveler must be signed by the operator, the FDA considers our record-keeping to be a hybrid system. A link must exist between the electronic record and the signed traveler. We accomplished this by ensuring the integrity of the database using a proprietary solution. No traveler is printed without first verifying that the data is unchanged from when it was initially saved.

To respond to events generated by the UUT, the process model calls a sequence that executes in its own thread, which monitors the UUT's events and provides feedback to the test thread asynchronously. Some events cause the test to be failed, and others are expected to occur. Synchronization is necessary between the two threads. Logging of all the events was added, and the logs can be accessed from the operator interface menu.


Automated Module Test System
File Run Test Troubleshooting Tools
TestStand Run-Time Operator Interface - Medrad Single Pass - DEMO.seq [4] (Running)
Test Steps
Log Window - Test.vi
Test Log
Log Detail
0 1 2 3 4 5
Clear Log
Print Log
Close Log Window
Test Dual Overlay Buttons
FillB
Auto Load
FillA
Prime
Mov
Arm
FAIL
Start
8:46 AM

```
        Passed     Numeric Limit Test
        Passed     String Value Test
        Passed     Multiple Numeric Limit Test
        Passed     VisualInspection_Call
   ->              ConnectUUT_Call
                   ProductSoftwareVersion_Call
                   TouchScreen_Call
                   MemoryTest_Call
                   HandSwitch_Call
                   ScreenLegibility_Call
                   FrontButtons_Call
                   ColorCoverage_Call
                   SoundTest_Call
                   DisconnectUUT_Call
                   <End>
```

Timestamp	Test name	Status	Comments
03/25/2003 14:31:17	DisconnectUUT_Call	Skipped	
03/25/2003 14:31:16	ConnectUUT_Call	Error	
03/25/2003 14:31:16	ConnectUUT_Call-->	Error	
03/25/2003 14:29:53	ConnectUUT_Call-->	Passed	
03/25/2003 14:29:49	VisualInspection_Ca	Passed	
03/25/2003 14:29:49	VisualInspection_Ca	Done	
03/25/2003 14:29:49	VisualInspection_Ca	Done	

Press button by button on the Head overlay. The blinking button on this picture shows which button must be pressed on the Head overlay.

Press the FAIL button to manually fail the test.

The Framework

The Framework resulting from the solutions discussed above can be run minutes after it has been retrieved from source code control, and the test engineer can then begin to add features specific to the test under development. Tests currently under development include hardware fixtures using NI DAQ boards to acquire voltage measurements, measure sound, and control UUT power.

All of the required features such as FDA regulatory compliance and traveler generation are built in to the Framework so that test engineers can focus on developing just the necessary modular tests. With the new product development teams building new products faster and faster due to their common platform, the Framework serves as the test development team's own common platform. Test systems built using the Framework will be developed more quickly than the previous systems. Once a test system is in service, the operators can put the UUT into the automated test system and after a few manual procedures the test system will complete the test. This will allow the operator more time to build new modules.