ActiveX Control Simplifies Design of Operator Interface

by Dennis B. Thiers and Chris DeSalvo, Hewlett-Packard

When developing a new functional test system, creating the operator interface to the system can be a challenging task. The interface is the physical link between the operator and the test system. Whether the operator interface is simple or complex, creating it can be a tedious and time-consuming job.

A very simple interface might provide only start and stop buttons to deliberately limit operator interaction with the test system. A pass/fail indicator may be the only status information offered if the operator’s task is to sort devices into pass or fail buckets.

For troubleshooting, you need to develop an interface for more sophisticated operations. For example, a troubleshooter may require the name of the failing test and information about how the test failed. Also, you may need an option to re-run or halt the test on the point where it fails and to take measurements manually.

Today, developing an operator interface generally requires expertise in object-oriented programming, such as C or C++, and Microsoft Foundation Classes (MFC). Since most test engineers are not experts in MFC and object-oriented programming, it is difficult and time-consuming to use these tools to create an operator interface.

With a definition and understanding of operator interfaces in today’s test systems, let’s look at some recent developments in manufacturing test. In some newer test systems, one operator panel may control multiple test-equipment sets.

In mobile phone testing, because the product volumes are very high and test times can be long, many sets of similar equipment are needed to perform the tests. Normally, you would run an operator interface for each of the test executives.

A typical operator interface is compiled as a .dll with tight links to the test-executive application. This multitasking environment would require multiple log-ins, multiple test plans, and bus-sharing drivers.

So besides the in-depth understanding of object-oriented programming and MFC, you must know how to create this multitasking operator interface. Given today’s pressures to develop and deliver new systems to production, you need easier ways to build an operator interface.

An easier and more efficient method would be for one operator to control multiple fixtures and multiple sets of test equipment from one log-in. New tools make creating an operator interface easier than programming in C++.
**New Tools**

Visual Basic is a superb development language for creating graphical user interfaces. The combination of Visual Basic, Component Object Model (COM) objects, and ActiveX controls has greatly simplified the development and customization of graphical operator interfaces for manufacturing test.

COM establishes a common method for one piece of software to access services provided by another piece of software, including libraries, local processes, and remote processes. It offers a consistent, structured, and language-independent means to provide services.

ActiveX controls define standard interfaces and conventions that build on COM technology. Using ActiveX controls, you can create reusable software components that perform common tasks such as providing a user interface and controlling a test executive.

Recent improvements in state-of-the-art test-executive programs using COM are included in a new mobile phone test platform. This test-system platform uses Visual Basic and ActiveX controls to create an operator interface that allows concurrent measurements through a single instance of the operator interface in a multitasking environment.

**Interface Software**

A key design goal of the interface program is to allow a single operator to control up to four test fixtures without significant loss of accuracy or test performance. Another design goal is to use a single bar-code reader to identify device-under-test (DUT) serial numbers and a strip printer to tag failed DUTs.

These operator interface controls are suitable for a 12” or larger touch-screen display. So, when testing four mobile phones, four sets of test instruments are driven by one PC, with one monitor for the operator interface and one set of peripheral equipment. This not only saves money, but also simplifies the tasks of synchronization and peripheral sharing.

In this mobile phone test platform, concurrent testing of multiple DUTs can be managed efficiently from one Visual Basic application running on Microsoft Windows NT 4.0. This test software contains an ActiveX control to communicate with a COM automation object in the test executive. The COM interface enables the test executive to act as a server and the Visual Basic operator interface to act as a client as shown in Figure 1.

A rich set of functions allows the ActiveX control to perform most test operations such as operator log-in, loading and running tests, and querying the status. To efficiently manage a complex test-executive program, the ActiveX control was designed as a structured object containing a hierarchy of subobjects, such as a Test Plan, a Security object, and collections of objects such as Registered Test Plans and Tests, as shown in Figure 2.

**ActiveX Control Examples**

ActiveX controls typically have associated methods, events, and properties. Methods are analogous to functions or procedures.

An event is a method that is called when a specific condition occurs. A property of an object defines a specific attribute, such as color or size. Here are some examples of methods, events, and properties of the ActiveX control in the mobile phone test system.

The control provided in the test software has methods for loading, manipulating, and running groups of tests called test plans. For example, a test operator might load a test plan by clicking a button control on a form in the Visual Basic application. The button-control click-event routine then would invoke the ActiveX control method, which loads a test plan:

```
TestExecutiveControlOne.LoadTestplan("CDMA Mobile Phone Test")
```

TestExecutiveControlOne is the ActiveX control and LoadTestplan is a method. This method results in a call to the test-executive COM object, which loads the CDMA Mobile Phone Test Plan.

Several events could occur as a result of executing the LoadTestplan method. If the test executive successfully locates and loads the test plan, it calls the ActiveX control which results in an AfterTestplanLoad(Path) event call in the operator interface. A string containing the path to the test-plan file, which was loaded by the test executive, is passed to the event call. A status indicator now may be updated to inform the operator that the test plan is ready to run.

In addition, a property of the ActiveX control is set:

```
TestExecutiveControlOne.Testplan.Name
```

The Name property actually is a property of the Testplan subobject of the control. The Name property of the Testplan object is set to CDMA Mobile Phone Test.

The test operator now clicks the RUN button on the operator interface application. This is programmed to invoke another method in the ActiveX control:

```
TestExecutiveControlOne.Testplan.Run()
```

As in the LoadTestplan example, the run method causes events to occur. There is an AfterTestplanStop(HaltReason) event. This event passes back the reason that the test-executive task has stopped the test. Some valid reasons are pass, fail, user stop, and test exception.

An indicator is updated to inform the test operator of the status of the test-plan run based on the HaltReason parameter. After a test plan is run, properties of the control and its subobject, Testplan, are updated by the test-executive program.

For example, the Elapsed-Time property is set to the elapsed time of the latest test-plan run. Many statistical properties also are updated so you can monitor the performance of the system and testing trends.

**Visual Interface**

The operator interface for the mobile phone test system is
a Visual Basic program that controls testing up to four phones
and communicates status to the operator. The Visual Basic
program also runs automation features, such as bar-code read-
ing, test fixtureing, and datalogging.

Using a Visual Basic Multiple Document Interface (MDI)
form in the mobile phone system provides a more convenient
way to structure the operator interface. The screen is orga-
nized into well-defined regions dedicated to each test fixture.

A main MDI form sizes itself to fill the entire screen. It then
creates child forms associated with each test fixture. Each child
fixture form contains an ActiveX control to communicate with
its own local test-executive process. The fixture forms are
identified and associated with test fixtures in three ways:
• The fixture form is labeled with the fixture number.
• The background of the form is given a unique color that
also may be present on the physical fixture.
• Each fixture form is placed in a separate quadrant of the
MDI form.

This operator interface and test application are completely
event-driven. All actions are initiated by commands from
the operator pressing buttons or test-executive events deliv-
ered through the ActiveX controls as shown in Figure 3. This
operator interface application is single-threaded, and all of the events are queued so that no locking is necessary
to control shared peripherals like the printer or bar-code
reader.

Conclusion
This new method for developing operator interfaces on
test systems provides an easier and more efficient way to do
your job. Improvements in applications using ActiveX con-
trols, multitasking operating systems, bus-sharing instrument
drivers, and Microsoft Visual Basic have allowed effective
use of operator panels to communicate with multiple test
executives and multiple sets of test equipment.

About the Authors
Dennis B. Thiers is the R&D engineer at the HP Measure-
ment Systems Division. He received a B.S. in physics from the
University of Illinois and a B.S. in computer science from
Colorado State University. Mr. Thiers has worked for HP for
14 years, nine of those as a software development engineer.

Chris DeSalvo graduated from the University of Pittsburgh
with a B.S.E.E. and completed graduate-level classes in mar-
keting and management. In his 20-plus year career with HP,
he has worked in a variety of marketing assignments and
presently is a marketing program manager for the system
platforms business.

Hewlett-Packard, Measurement Systems Division, 815
14th St. SW, Loveland, CO 80537, (970) 679-3249.

Reprinted from EE-Evaluation Engineering, March 1999
For more information on Hewlett-Packard Test and Measurement products, applications or services, please call your local Hewlett-Packard sales office. A current listing is available via the Web through AccessHP at http://www.hp.com. If you do not have Web access, please contact one of the HP centers listed below and they will direct you to your nearest HP representative.

United States
Hewlett-Packard Company
Test and Measurement Call Center
P.O. Box 4026
Englewood, CO 80155-4026
1 800 452 4844

Canada
Hewlett-Packard Canada Ltd.
5150 Spectrum Way
Mississauga, Ontario
L4W 5G1
1 877 894 4414

Europe
Hewlett-Packard
European Marketing Centre
P.O. Box 999
1180 AZ Amstelveen
The Netherlands
(31 20) 547 9999

Japan
Hewlett-Packard Japan Ltd.
Measurement Assistance Center
9-1, Takakura-Cho, Hachioji-Shi
Tokyo 192-8510, Japan
Tel: (81) 426 56 7832
Fax: (81) 426 56 7840

Latin America
Hewlett-Packard
Latin American Region Headquarters
5200 Blue Lagoon Drive, 9th Floor
Miami, FL 33126 U.S.A.
Tel: (305) 267 4245/ 4220
Fax: (305) 267 4288

Australia / New Zealand
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130, Australia
Tel: 1 800 629 485 (Australia)
0 800 738 378 (New Zealand)
Fax: (61 3) 9210 5489

Asia Pacific
Hewlett-Packard Asia Pacific Ltd.
17- 21/F Shell Tower, Times Square
1 Matheson Street, Causeway Bay
Hong Kong
Tel: (852) 2599 7777
Fax: (852) 2506 9285