

# Introduction to Regression Test Automation Software

Test automation software falls into a number categories, such as code analysis, regression testing, and load testing software. We will focus on regression testing software for Windows based applications. This software can be used to do Unit Testing, Regression Testing, Functional Testing and System Testing.

Regression test software requires an understanding of the Windows GUI interface composed of Text and Objects. Modern tools have essential components and associated automated testing methodologies. As development technologies have evolved so have automated test tools. Development environment, test requirements, test methodologies and skill level should be the basis for selecting a particular product.

## Testing GUI Applications

Text applications are most familiar to users of legacy Host based systems, UNIX or IBM Mainframe systems. The original DOS PC supported character based applications. When you execute the command prompt in Windows – that is a character based application.

What distinguishes Character based applications from GUI (Graphical User Interface) is that all verifications of the application are Text.


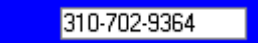


GUI Applications, such as Windows can have Objects as well as Text that may need to be verified to properly test the application. Although objects may contain text they also contain other behaviors that can impact the application such as clicking on buttons. Objects also have States; many of us have seen this as Buttons or Menu items grayed out.

### ***What are Objects?***

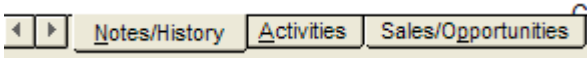
Objects are components that make up a Windows Application; these are typically referred to as Windows and Controls. Controls are found in Windows.


### ***What are Controls?***

Some familiar Controls:

Static or Text		Check Boxes	<input checked="" type="checkbox"/> E-mail
Edit Boxes		Radio Buttons	<input type="radio"/> Daily
List Boxes		Button	

Some Controls may not be so familiar:

Tab Control 

Tree View 

Spin 

And although we have seen many calendar windows these typically fall under the category of Custom Controls as do many grids.



### ***What are Custom Controls?***

They are sometimes called Third Party Controls. They are specially programmed controls that can have very unique features. Most grids, spreadsheets or tables are third party controls. These controls may be composed of a Window with recognizable controls such as list boxes or edit. However, they also may only be recognized as a single window with bit-map graphic images that change based on mouse click or keyboard behaviors. Web applications that use HTML to formulate the page would not have custom controls. Java, VB and C++ applications can have custom controls.

Filter	<input checked="" type="checkbox"/> Notes	<input checked="" type="checkbox"/> Histories	<input checked="" type="checkbox"/> Attachments	<input checked="" type="checkbox"/> E-mail	Insert Note
Date /	Time /	Type	Regarding		
6/28/2004	2:00 PM	Meeting Held	Web Demo - Paul - ownit Mortgage		
6/28/2004	12:00 PM	Meeting Held	Web Demo - i_tech - Owned By First Interstate		
6/25/2004	8:00 AM	Meeting Held	Web Demo - Paul - RAE Systems -Evan		
6/22/2004	1:30 PM	Meeting Held	Conference Call - i_Tech		
6/21/2004	10:11 AM	To-do Done	M Away		
6/21/2004	10:11 AM	To-do Done	M Away		
6/18/2004	1:00 PM	Meeting Held	Web Demo _ Pat Gleason		
6/16/2004	10:30 AM	Meeting Held	Doug's Graduation		

An Example of a Grid

### ***The State of an Object***

There are a number of states that controls can exhibit. Hidden is one – there but not displayed. Check boxes can be selected or not selected. A tool should help you understand and evaluate an object's state. For example, it is not uncommon for the Apply Button to be grayed unless there has been a change made to the contents (objects) of the Window. A change to one of the objects creates a state change and the Apply Button would become active. The application as a whole has states as Windows and Objects change based on previous interactions with the system.

### ***Behaviors***

There are behaviors that impact an application. We most typically see these as mouse or keyboard actions that create other events such as new Windows taking focus or the display of controls or removal of controls from Windows.

## **Components often found in Good Tools**

### **Object Recognition**

At the heart of all good test tools is how they identify and understand objects, behaviors and states. This is commonly called object recognition. This ultimately is translated into a Test Script.

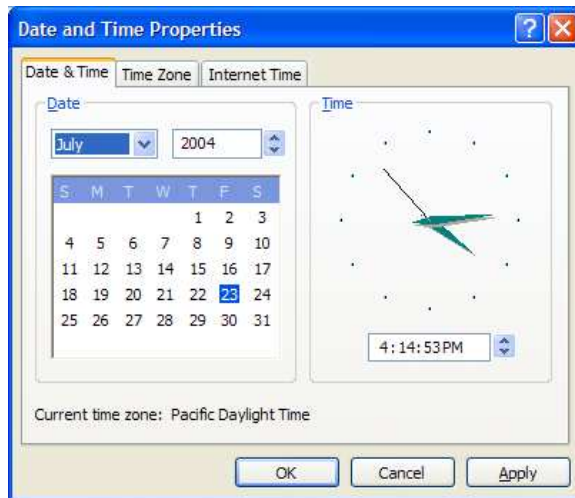
In the process of object recognition, all good tools create variables that can be modified. An example would be modifying the data selected in a list box, or edit area for a password. The value of variables will become clearer when we discuss automated testing methodologies.

Tools have different strengths and can be limited to certain development environments. Web Applications are not the same as C++ or VB applications; they are all different environments with different technologies.

## A Spy feature

*Is the Object I see really what I think? Depending on the development technology supported most good tools should have a Utility to help better understand the Objects.*

For example let's take a look at the Calendar that is part of Windows.



A spy tool helps the tester understand the objects he is testing. Typically the tool will break all the objects into their smallest object components. For example consider the calendar control in the window above. A good spy tool will identify information about that control such as Class –CalWndMain and other properties such as captions or titles, size and other relevant information.

## Robust Test Language

Not only does the development technology impact test automation, the way the application is developed or behaves can have an impact. Sometimes information must be captured from the test application and be used as input to test other functionality of the software. Other situations require logical decisions emulating user interaction at application branching points. A strong test language is invaluable in achieving an automated solution.

## Optional ways to work with Objects

Good automation tools also provide optional methods to recognize or define objects. They will also provide alternative methods to execute objects, such as a mouse click to specified coordinates. Many also will allow you to associate behaviors of common objects to undefined objects. An undefined object may behave as a list box; you then could define it so the automation tool would recognize it's behaviors as a list box.

## **Verification Capability**

The tool must provide a way to verify that the target application is working correctly and providing the anticipated responses. Verifications let you confirm that the application is validating input, performing calculations correctly, saving data reliably and reporting accurately. Verification compares actual responses to expected responses and reports any discrepancy. A good verification feature not only verifies textual information but is able to validate object states and behaviors.

### **A central repository and modular design to manage test assets.**

The main purpose for a central repository is to provide capabilities to share test scripts, verifications and other test assets.

### **An Object Map or method to make a change to an object in all scripts.**

Objects change, and without architecture to support a global change - script maintenance becomes burdensome.

## **Recognize Events for Synchronization**

Windows is event driven – for example clicking a button may display a completely different Window to view. Although most good tools tend to automatically synchronize, there are instances where event-recognition may be needed to properly coordinate further execution. This occurs usually as a result of long processing times or conditional branching behaviors. Good event recognition can be monitored continuously or sequentially. Most events typically are as simple as waiting for text to appear in one window before executing the next control.

## **Logs or Audit Trails**

Logs provide a detailed report or audit trail on each command issued from the script or object and behavior executed. One feature that is nice to have in the log file is time and date information. Logs should allow you to quickly and easily analyze the outcome of the tests. A good log is also an excellent source in debugging the automated tests you create.

Logs should also provide an easy way to understand verifications. If verification fails, that is, the target application did not respond as expected, both the expected and actual responses should be stored in the log for comparison.

# Different ways to approach Automated Testing

## 1. Capture/Playback

This method for regression testing an application simply requires capturing a user's actions. Then you simply playback what was captured. The Capture/Playback method usually leads to a poor test tool experience. This method evolved from automated test tools that were designed for character based applications. Marketing drove this more than reality.

Advantages:

1. Turn on record and no experience is needed.
2. Can use existing staff (Users) no systems development skills needed.

Pitfalls:

1. Script may not run properly as the application may require unique tool settings to properly record the interaction with the application.
2. Transactions entered may not fully test all of the application's key functionality.
3. Not a real strong way to validate Objects. As the state of objects can change so can the state of an application making it impossible for scripts to reenter the same information.
4. Time running repetitive non-essential scripts increases testing time.

## 2. Test Planning and Modular Script Development

Because the state of objects and the application can change, planning is needed to automate testing. It requires consideration of restoring baseline test data, creating test plans that cover the key functionality of the software and understanding how the test tool and application work together. Test scripts then can be recorded and/or coded to properly execute the objects. Although scripts may be recorded with capture/playback methods – they are recorded using a defined test plan with an understanding of how the test software will work with the objects. Part of the planning process will organize the test scripts in modules that can be easily maintained as well as run individually to test a specific functionality.

Advantages:

1. Testing will be more complete and successful
2. Automated test tools will deliver the best results when used in test plans that are repeatable.
3. Easier to maintain test scripts
4. Test specific and key functionality.

Pitfalls:

1. Requires rudimentary development skills for tester. (Understanding of the objects).
2. Scripts may still require a certain amount of duplicate development effort as they may only change slightly to test different functionality.

### **3. Data Driven Test Automation ( This term has been enhanced some and is now being marketed as third generation testing)**

This type of test automation further advances test planning and modular script development. The main idea in data driven testing, is that much of the testing is transaction related and follows a very similar process moving through a common set of objects.

The spreadsheet has been commonly used by test planners to create the test data they incorporate into their test plans and verification. A row of data can represent one test case with different cases represented by additional rows. The columns on the spreadsheet would represent the individual objects. In manual testing, the tester loops through each row of data to perform each test case and verification.

Data driven testing automates this manual process by reading test data from a file and performing the process of looping through each test case. Such testing can require strong system skills as well as basic programming skills. Object Mapping or Capture/Playback typically is used to record basic object manipulation. Test data files are read and then introduced into the script by creating variables. More modern tools may be designed to do this without programming or with minimal programming.

Advantages:

1. Can greatly reduce the number of test scripts required. Fewer scripts to maintain.
2. Have all the benefits of test planning and modular script design.
3. Lends itself well to larger teams of analysts and programmers where test plan development is put into the more capable hands of business analysts and test automation into more skilled development personnel.

Pitfalls:

1. May requires more rudimentary development skills and programming skills.
2. Requires more coordination with test team.
3. Scripts can be more difficult to maintain.

#### **4. Dynamic Test**

While criticism is anticipated, systems by their very design sometimes cannot be set back to a steady state. Also, the cost of creating a duplicate test system may not be feasible, such as when the system requires data obtained from an existing system.

Even in systems where they can be taken back to their original state, there are needs to dynamically test. For example a system that generates unique customer and invoice numbers, may need to capture those numbers and then use them as input to properly test additional functionality of the system.

This requires the greatest level of planning and may require that the tool collect information from another system or system module prior to execution. Data driven testing becomes dynamic as some of the data may be created or captured from the tools interaction with the application. This is where the automation capabilities and data capture capabilities of a tool become important.

Advantages:

1. Dynamic automated testing sometimes can provide great rewards as it can emulate very repetitive and time consuming manual processes.

Pitfalls:

1. Requires highest level of development skills and programming skills.
2. Scripts will be more complex and require more skill to maintain.

#### **5. Smoke Test**

This method is included as it has been another strategy of successful users of test automation tools. But it is more a testing method that can be done in addition to and typically before the other methods. The main purpose of this type of test is to verify the existence of objects. Then you can modify your scripts for new, deleted or changed objects before performing a full regression test. It simply requires creating scripts that execute the application and test for the object's existence.

There were tools that had a dubious feature that created a smoke test automatically. Even though the tool required no human interaction, it had almost all of the pitfalls of the Capture/Playback methodology above.

#### ***Taking a Tour of your application and developing test strategies.***

Testing GUI based applications requires testers to have an understanding of the application and objects. Automated testing manipulates these objects as it will

exercise controls by selecting items from List Boxes, select Check Boxes and click on Buttons. In the process, information that may be contained in a particular control may need to be verified – such as all items in a List Box or text in a particular Window or Control.

Use the tool and see how it actually works with your application, then you can develop your testing strategies before you develop your scripts. Many tools present themselves as just turning on the record button and that's all you have to do to create test scripts. That may happen, but don't expect it as your software may require a little extra help to get test automation to work. Most good tools have features to help in this process and you need to be trained on how to use them.

With proper training and the right approach, test automation can be a very rewarding venture. Test automation saves time and increases quality. If you are only going to test the application once or if the GUI interface will be frequently and extensively modified each release, you probably will need to manual test. Test automation is for repeatable tests.

## **Direction of Automated Test Tools**

Automated test tools have been evolving along with development technologies. At the core of most tools is a proprietary testing language that looks like some other familiar language such as C or VB.

Newer products have tried to spruce this up by hiding the underlying script and laying on top a GUI that displays an object view. This in some way makes life easier for a less sophisticated user who panics at the sight of code.

More significant is that the newer products have been engineered on pure development technologies and have those languages as the underlying base. One is Compuware's TestPartner which uses VBA as its scripting language and OpenDemand's OpenLoad that has Web technologies at its core.

There are some very attractive reasons vendors are choosing to move in this direction as certain underlying architectures are more reliable in testing compatible architectures. TestPartner's VB based technology makes it a solid product for recognizing objects and provides a very extensive test language.

Tool vendors are also targeting specific technologies such as the OpenLoad product, which is Browser based and is for Web Application testing. There can be some great advantages to this as the same tool can be used for regression and load testing.

Other vendors such as AccordSQA are taking a different approach and trying to simplify test script creation by creating a Grid interface which can be considered

a third generation tool for data driven testing. The SmarteScript tool has very solid object recognition capabilities and in a lot of situations offers a better way to test.

Likewise, Openload eliminates scripts entirely, replacing them with an interactive Web interface. Both SmarteScript and OpenLoad interfaces are designed to eliminate as much as possible any user interaction with scripts.

There are other vendors out there that are working on their next generation tools, some older products that have been tried and true are now reaching the end of their useful life. Some of the underlying technology for these older products goes back quite a few years. This may cause discomfort to those who are experienced with a particular technology, but if you are starting to test for a new release that was developed with more recent tools it may not be the best decision to automate testing with older products. These older architectures may fail over time and they may not be as well suited to test newer technologies.

In larger enterprises or ones with legacy systems, there is the desire to have one tool, not many. If the tool works well for all the environments then there is a definite advantage over a more specialized tool that might work better as training and software acquisition costs are reduced.

However, the biggest cost is not the acquisition cost of the tool – it is the time it takes to create and maintain automated tests. So a tool that is easier to use and maintain is well worth the cost of acquisition. And if the tool does not work or is difficult to use, at best tests will not get automated or even worse your automation effort will fail.

**About the Author:**

Bruce Warner is the owner of Operative Software Products - [www.operativesoft.com](http://www.operativesoft.com). He has over 20 years experience in the computer field and 15 years selling, training and consulting on Test Automation Software. Operative Software Products operates in the United States and Canada and provides solutions for test automation, application analysis and network performance.

Copyright 2004