Geometric

Scriptless Test Automation

Next generation technique for productivity improvement in software testing

Version 1.0
February, 2011

WHITE PAPER

Copyright Notice

Confidentiality Notice

Geometric

Contents

Scope: GUI based Test Automation

GUI based Test automation is the activity of creating automated test scripts that can be run without human intervention. They run off the same UI that the end User or Customer would see.

These scripts can cover a lot of application areas in a repeatable manner. They can cut down on test execution time significantly as they run continuously and unattended, even overnight.

From here on, in this document, “Test Automation” will mean “GUI based Test Automation”.

Evolution of Test Automation

Historically, Test Automation tools started off by giving a macro recording facility – example is MS Excel. Though the application UI was used to record the automation script, the replay mechanism depended on synchronous API calls, rather than the UI. The test execution engine would fire a command, and then wait for the command to complete.

A macro based tool was good for such short length automation. Combine a logical sequence of such short length macros, and your test flow is automated!

This mechanism can be seen in some desktop based CAD tools, wherein macros can be quickly recorded, and the recorded macros can be immediately played back with little or no modification.

This approach is the best, even today, but applies to only those applications that have such a framework built in. This is a good example of testability being built into an application, right at the design stage.

However, not all software applications that are in the market today have such a test framework built in. It’s this segment of applications that the **third party test tools** address.

Test Automation using third party test tools

Test automation using third party test tools historically started with test tools that provided a dumb record and replay facility.

Though this approach worked well for some desktop based applications, there was a lot to be desired in terms of execution speed control, type of validation check points, modularity of scripting, how well test input data could be fed to the script.

This was the **first generation** of test automation tools.

Moreover, even though the speed of automation creation was fast, the **scripts were very sensitive to UI changes**, and changes to UI would eventually result in throwing away the previous automation scripts and creating new ones.

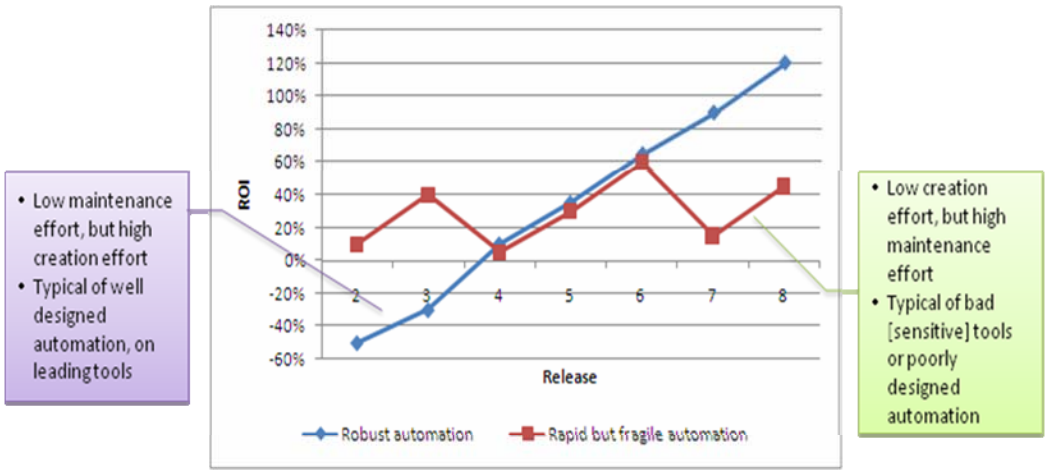

Figure 1. Test Automation Productivity

The **second generation** of test automation tools brought a whiff of fresh air; they brought a lot of features to the test tool that brought up similarities between test tool IDE and developer's IDE. These tools provided full-fledged scripting, or even objected oriented language support to the automation engineer. A lot of tool APIs were present to simplify the common windows based tasks like application launch, file operations, generating elementary test reports. Some of these also came with platform based extensibility SDK, for example, for .Net or Java.

The variety of object recognition as well as check points possible was extensive and coding could achieve virtually anything that the automation engineer desired – be it data driving, keyword driving, use of variables, functions, separation of object definition from object references across the scripts.

However, this **script based approach still needs a lot of in-project investments**, in terms of time to learn and gain expertise in the scripting language, as well as the time to create a suitable framework that not only harnesses reusable code, but also eases the data driving of the scripts – all of these eventually draw a line between the manual tester and the automation engineer.

So, even though the automation 'feasibility' problem got solved with the advent of more robust object recognition technology, the 'productivity' problem still leads to a low ROI from test automation.

With these thoughts in the background, we **started looking for test automation accelerators**.

The quest for ROI from Test Automation

We evaluated the following approaches for the productivity problem:

Table 1. Evaluation of various approaches

#	Approach	How it helps	Remarks
1	**Open source tool**	• Direct reduction in investment needed	• Might help overcome budgetary constraints • Investment in effort might be much larger than that in the tool • Test Tool support virtually non-existent • Usually take time to catch up on the latest platform or technology
2	**Big bang automation**	• Benefit of automation made available much sooner by deploying a huge automation team	• Could be constrained by tool costs • High GUI stability important
3	**Pre-built framework**	• Part of the development already done • Available at low cost as it is sold to multiple buyers	• ERP example: SAP automation frameworks
4	**Model based**	• Automation based on a model of the app, thus the scale of the operation is smaller • Being based on a model, it provides a higher level of abstraction, thus test design itself can benefit from this approach	• Many different approaches evolving • Wait and watch
5	**1st generation record and playback, with little or no scripting support**	• Extremely rapid creation of automation	• Poor robustness • Poor maintainability • Lack of tool APIs hamper creation of any suitable frameworks • Usually unsuitable for enterprise apps • Perhaps for very stable, desktop apps

6	**2nd generation Record and playback, with scripting language support**	• Robust object recognition	• Maintainability has to be built in through a suitable framework • Enables the automation engineer with a full-fledged scripting language support • Coding skills required

None of the above approaches looked promising enough in terms of productivity of automation script generation.

But the second generation test tool IDEs have become full-fledged code editors that support the automation engineer in not only creation of scripts but also debugging and execution. And yet, frameworks still need to be built before any long term test automation project can go live with testing.

At this point, we started thinking on other lines.

Years ago, Microsoft's Visual Studio and Foundation Classes brought **a quantum jump in the productivity** with which Windows GUI could be developed – instead of hand coding with the Win32 API, developers could now use wizards to 'visually design' the GUI in a 'studio'.

We soon discovered that **Scriptless tools bring the same concept to test automation.**

Instead of a code editor, automation engineers now work in a graphical environment, where they can compose automated tests by selecting objects and actions from dropdown menus. They can also visually create conditions, iterations, etc.

We have worked with several Scriptless tools and are getting 50% or higher productivity as compared to script based automation. We are recommending it to our customers, and of course, for our own internal projects.

We believe it is justified to view Scriptless automation tools as constituting the next generation of tools that will gain prominence in test automation.

Constituents of Scriptless Test Automation tool or framework

Scriptless test automation provides a very easy to use interface – instead of learning a scripting language, or even a programming language, one gets to work on a drop-down and Excel kind of user interface.

All the requirements of a test automation script, like the sequence of keywords, steps within each keyword, data per step, variables that have a scope of the full length of the keyword, or even across the sequence of keywords for the test, and UI object definitions – are available in a very easy to use interface – and no programming is involved!

All the good concepts of test automation like data driving, keyword driving, and modularity are all packaged in very easy to use software, **without taking away any of the power that a scripting based tool provides.**

Some of the notable features of a Scriptless framework that aid in automation creation are:

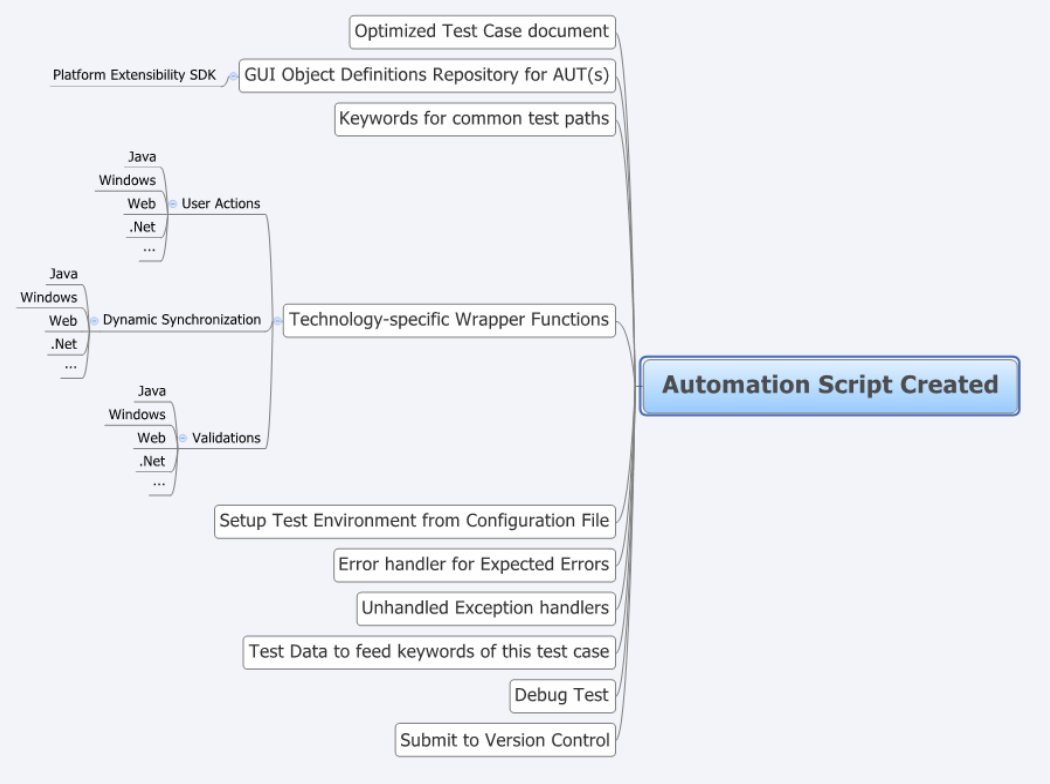

Figure 1. Inputs and Activities required to Create Automated Test

Moreover, they have a client-server based approach, wherein, automation once created is readily available to the whole team, at the push of a button.

These tools not only provide script creation support, but also go beyond – they let you **debug from virtually any step within the script.** They also provide test management required for test automation, like composing test suites, running the test suites, reporting the test run result. If one needs **history of test runs,** even that is available!

Some of the notable features of a Scriptless framework that aid in Test execution or re-execution are:

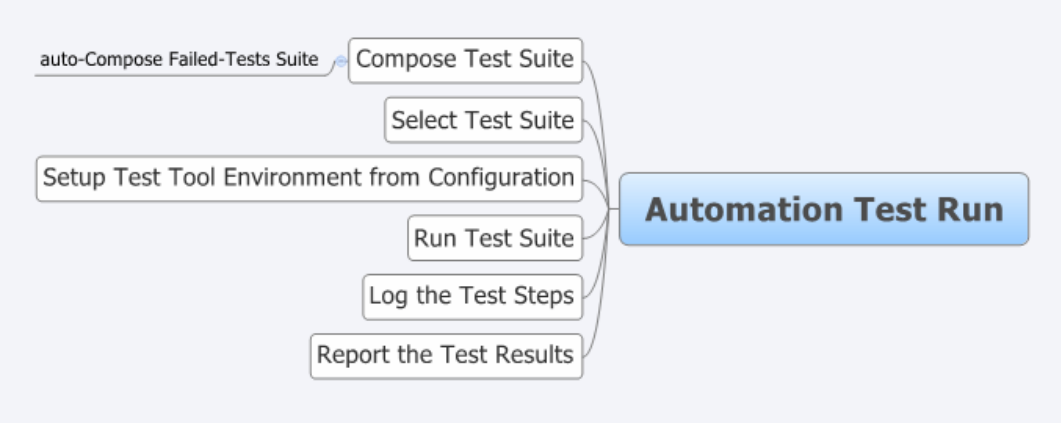

Figure 2. Activities involved in Automated Test Execution

Some of the notable features of a Scriptless framework that aid in Automation Result Analysis are:

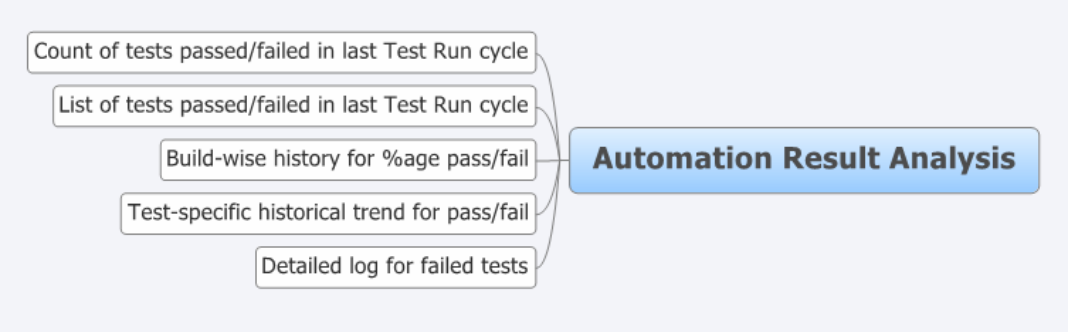

Figure 3. Current or Historical Test Run Analyses

We would like to note two such tools that have a lot of these features – Certify from Worksoft, which is a test automation tool, as well as Qualitia from Zensoft Services, which is a test automation framework that rides on top of other leading test automation tools such as QTP, RFT. We believe other vendors are also working on providing Scriptless capabilities.

ROI from these tools is impressive as they are productive virtually from day one, as you do not have to spend time initially on writing a framework. Add to this, the fact that they are time tested – across multiple applications and over considerable amount of time.

The fact that they are easy to use also means that there is a very good rate of adoption of the tool by the Manual Tester community, and this benefits the organization!

Inference

Scriptless automation has the potential to be the next generation automation technology.

It not only addresses test automation's 'productivity' problem, but also does it in a way that does not force you to draw a line between the manual tester and the automation engineer.

Moreover, it lends a comprehensive UI to all the test automation Best Practices, thus it enables the Organization to Standardize – after one has done the first test automation project on a Java based application, one does not need to start from scratch on a .Net based project – the same test tool and the same UI will apply, as all the framework stuff is already available.

Caution

A note of caution, not all tools that initially appear to be Scriptless are truly script less, or are as feature packed as others.

Automation of custom UI components could in some cases need coding expertise.

Not all test automation tools support all technologies – so check with the tool vendor for any specific technology requirements.

Distributed test execution across machines, and a unified result reporting is a work in progress in some cases.

About the Author

Sudipto Paul is a Test Consultant at Geometric, and is responsible for building test teams, initial hand holding and guidance on test processes for iterative and agile projects. He has over 13 years of experience in the information technology field, and has had an all - round exposure to the software development life cycle. His experience includes roles of test consultant, test automation architect & engineer, manual test lead, requirements analyst, designer and developer.

Sudipto's primary areas of interest include test processes and technical testing. He has authored several utilities and test automation frameworks to aid both developer side and tester side testing.

Sudipto hold a BE (Mechanical Engineering) from Walchand College of Engineering, Sangli, and PGCBM from XLRI, Jamshedpur.

About Geometric

Geometric (www.geometricglobal.com) is a specialist in the domain of engineering solutions, services and technologies. Its portfolio of Global Engineering services and Digital Technology solutions for Product Lifecycle Management (PLM) enables companies to formulate, implement, and execute global engineering and manufacturing strategies aimed at achieving greater efficiencies in the product realization lifecycle.

Headquartered in Mumbai, India, Geometric was incorporated in 1994 and is listed on the Bombay and National Stock Exchanges. The company recorded consolidated revenues of Rupees 5.11 billion (US Dollars 108.1 million) for the year ended March 2010. It employs over 3600 people across 10 global delivery locations in the US, Romania, India, and China. Geometric was assessed as CMMI 1.1 Level 5 for its software services and is ISO 9001:2008 certified for engineering operations.