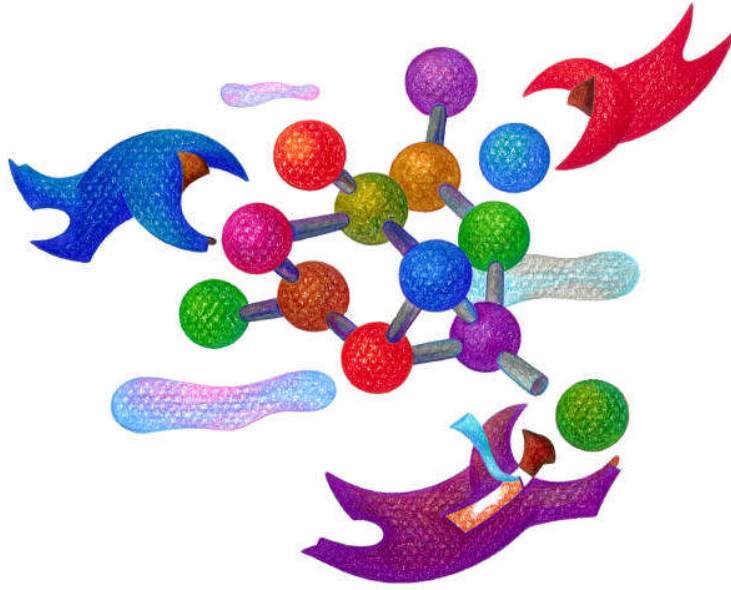# Fifth Generation Scriptless and Advanced Test Automation Technologies

By Jeff Hinz, CTO, TESTars Test Competence Centre.
with Martin Gijsen, Principal Test Automation Architect, DeAnalist.nl

Reviewed for accuracy by Christopher J. Scharer, Director VIVIT Worldwide
and Maurice Siteur, Managing Consultant, in Testing, Capgemini

www.testars.com

Table of Contents

1) Profiles

Jeff Hinz's background;
For the last 20 years Jeff has concentrated on Software Quality Assurance, Testing and Test Automation. Jeff was the beta testing site for Mercury's TestDirector version 1.5 on Windows 3.11 (1994) while at PAYCHEX. Jeff co-wrote the press release for version 4 of TestDirector (1997). Jeff has worked at both of HP-Mercury's largest accounts in the United States, PAYCHEX (East) and QWEST (West). The last 7 years Jeff has been working on the fourth generation test automation framework and structured test method TestFrame. Jeff has developed frameworks from the ground up with WinRunner and TestDirector. Jeff has also used HP-Mercury's latest offering Business Process Testing (BPT) with QuickTest Professional and Quality Center (2004). While at QWEST we were a beta testing site for Mercury's Business Process Testing. BPT is essentially a TestFrame clone, not as flexible as TestFrame, and only works with QuickTest Professional. Jeff has certifications in TQM, CMM, software quality assurance and test automation. Jeff's main interests are structured testing, managed testing, test automation, and the efficiency of Testware.

Martin Gijsen's background;
After a university study in computer science, Martin Gijsen has been working as a test automation architect for 10 plus years. In early 1998, Martin became involved in test automation for unit testing. In late 1998, he was part of a project that started black box testing of telephony switches, which are large embedded systems. Martin has extensive experience working on extending the resulting test automation platform to cover more features of the switches.
Martin also has experience automating the testing of diverse systems, including messaging systems (telecom and banking/payments), web services and GUI applications. Martin is interested in test automation for any kind of system. Martin is the Author of the Essential Test Automation Framework, and has presented at QWE, PSTT, ICSTEST and the Dutch Testing Day ('Nederlandse Testdag'), as well as customer conferences. Martin is a member of the Dutch Testing SIG TESTNET and is part of the working group for Model Based Testing within TESTNET.

2) Introduction

In 2007 in need of explaining fourth generation test automation Jeff wrote a white paper to get clients knowledgeable in basic test automation principles. For those needing an introduction to the basics of test automation please refer to the white paper Jeff wrote addressing the subject, Test Automation Awareness[i]. To understand advanced test automation, it is important to understand the progression/evolution of test automation approaches (see **Figure 1**):

➢ **1st Generation** – Record and Playback

➢ **2nd Generation** – Use/reuse of functions in

                       test scripts

➢ **3rd Generation** – Data Driven scripts/functions

➢ **4th Generation** – Action word (keyword) scripts/func.

➢ **5th Generation** – Scriptless Automation

Figure 1 - Test Automation Progression

Some terms of relevance here are **action word**, and **keyword**. The original **action word** concept was developed by Hans Buwalda at the Computer Management Group (CMG) of the Netherlands to aid in testing an implementation of the Amsterdam Stock Exchange. Action words are part of the structured test method TestFrame. An action word is a business specific action inside a test case. The test case itself is known as an action word test case. A **keyword** is the generic term for an action word but is used very loosely. For example keywords tend to be generic in structure and often utility based. This means that the keyword may not necessarily replicate a business specific action. An important aspect of keywords is that they are not part of a structured test method (STM).

For completeness I am introducing the **Test Action** concept relating to the open source test automation framework JUnit. **Test actions** are JAVA methods to be automated in the open source JAVA test automation framework JUnit (see **Figure 3**). These test actions are essentially steps within a JUnit test case. Test actions may or may not implement business specific actions. Note that JUnit was originally intended for unit testing, however people continue to attempt to extend/use it for functional integration and system testing.
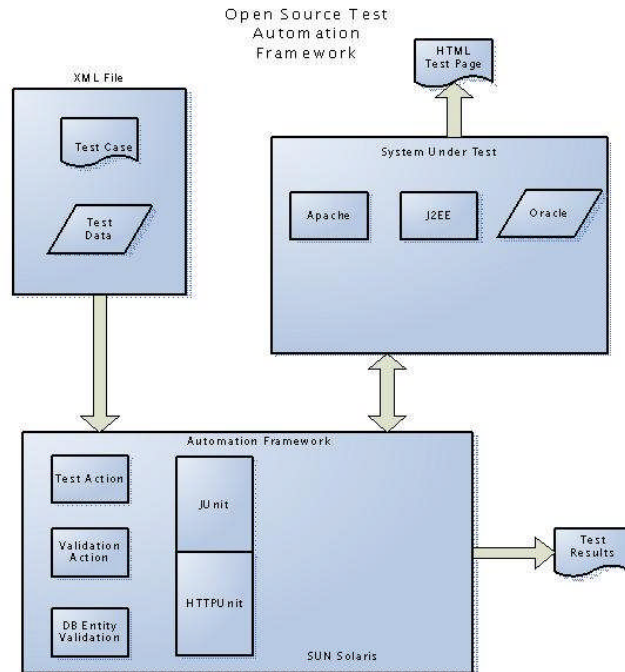
JUnit

Figure 2 - JUnit Architecture

Scriptless test automation takes two forms:
1) a test analyst not having to create an automated test script;
2) a test automation developer not having to create action word functions to implement an action word test case.

Some companies claim scriptless technology in the aspect/sense that a test analyst can create an action word/keyword test case by selecting the action words/keywords and including them in their test case. This eliminates the necessity of a test analyst producing a test automation script since the action words/keywords are implemented by a test automation developer.

Our fifth Generation Test Automation Solution is an, enterprise proven, action based test design and automation solution. One of the main challenges of test automation is maintenance of the test scripts. Today's faster release cycles, frequent changes to functional flow, changes to technical aspects of the product, etc do result in breaking the underlying test automation solutions. How does the automation framework keep pace with the product changes? Maintaining huge set of scripts is not only effort intensive but also error prone. Fifth generation takes a different approach to action based testing and thus drastically reduces the number of scripts to be maintained.

www.testars.com

3) Fourth and Fifth Generation Test Automation

Fourth generation test automation frameworks were developed by Hans Buwalda while at CMG in 1994 along with the action word technique, TestFrame[ii]. In generic terms some may refer to action words as keywords, or test actions in JAVA. HP-Mercury's equivalents are "business components" in their product Business Process Testing (BPT) (See 3rd Party Tools in the Appendix). Action words are somewhat different from keywords in that an action word replicates a business specific action. Many times keywords end up being utility based on the 3rd party test automation tool itself. Around 2001 CMG and Software Development Technologies (SDT) of California implemented TestFrame at a client in North America (Norwest Financial 1999) and started using code templates as a way to clone simple action word functions. The first implementation of this was developed by Christopher J. Schärer (a CUE Data Services, Inc. consultant at the time) on the Sapphire project at Norwest. Eventually an integrated development environment (IDE) was added to the solution to help in building complex action word functions.

Fifth generation is an action based solution, wherein action words are primarily used as building blocks and are reusable across test cases. The action based approach ensures that the maintenance is done at the action word level. Any change to an action word is reflected in all the places where it is being used. To reduce the number of scripts and maintenance overhead, fifth generation supports "**Standard Action Words**", "**Code Templates**" "**Code Generation**" and "**Custom Objects**" concepts that make fifth generation a scriptless technology[iii]. Each of these concepts is defined below see **Figure 3**.
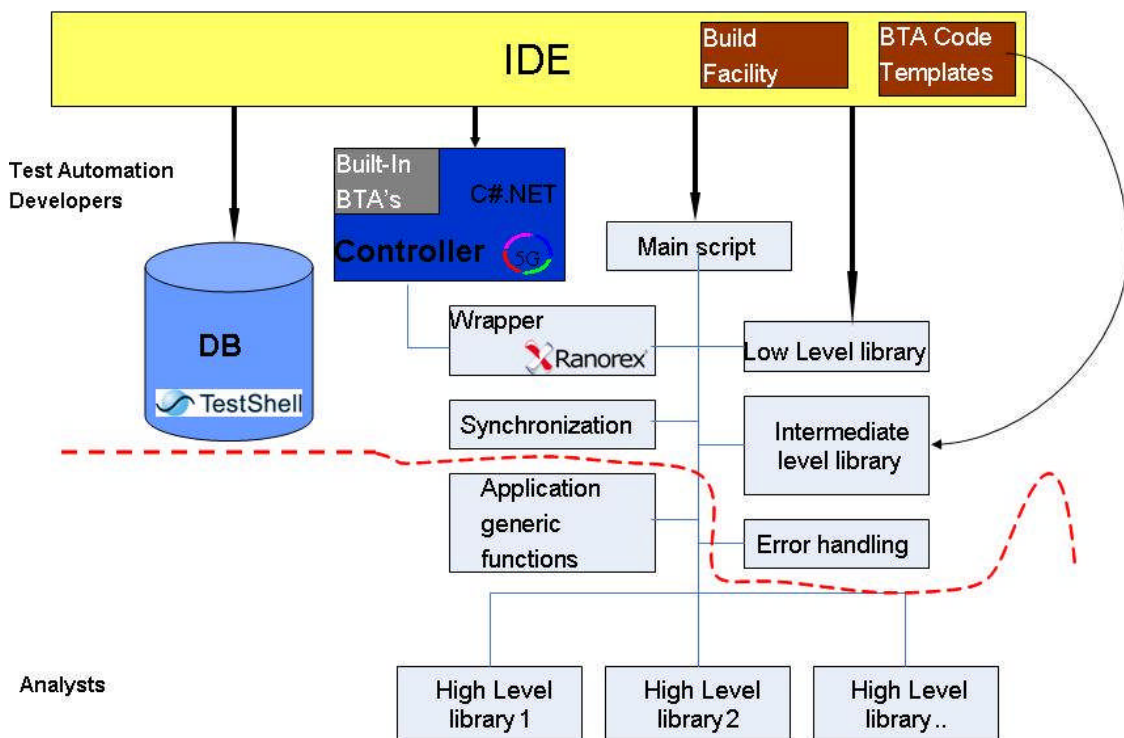

Figure 3 - Fifth Generation Test Automation Components

www.testars.com

3a Standard Action Words

A standard action word is implemented by specifying a few inputs at the action word definition level and saving the information. Fifth generation gets the action word information and follows a standardized process to do the playback for test execution. There will not be any script associated with the action word. Usual observation is 95% of the action words are standard in a Test Automation project thus making the action words scriptless and maintenance free.

3b Code Templates

Action words are associated with code templates that follow a standardized process for doing the playback on an application under test (AUT). The code template is a simple script that drives the process (See **Figure 5 – Appendix**).
Since a single script (code template) is associated with huge number of action words, the maintenance of one script will ensure that all the action words are maintained automatically.

3c Code Generation

Action words can be associated with specific Code Templates. When this event occurs the test automation developer can then automatically generate the action word function by selecting code generation. Once the action word function has been automatically generated it can be viewed and or edited in the integrated development environment.

3d Custom Objects

Fifth generation provides default playback implementation for many standard objects making it easy and faster to implement action words. Today, applications are developed with multiple technologies and also use third party controls to make the development faster and efficient. To do playback on these third party controls or controls that do not have a default implementation in fifth generation, automation engineers will write test automation code. Fifth generation provides a way to organize this code so that the same code is reusable across the test automation framework wherever a similar control is used. Users keep the automation code for these controls in custom objects. And any action word can use this custom object to do playback on a similar object/control in the application. Single point of implementation/code for custom objects makes it easy to maintain.

4) Hybrid Approaches and Tools

Some approaches and tools are called 'hybrid' in that they support multiple generations of automated testing. At first glance, it seems beneficial that several of the approaches are supported because different people like different approaches and such tools could also support a transition period. But some further thought reveals the other side of this coin.
Data-driven tests, for example, represent a complete test case as one line of test input. Such a test can easily be represented by a single high level keyword (that is given the same test data). It is therefore trivial to use keyword-driven tools for data-driven tests. In fact, it can be argued that keyword-driven supports or includes data-driven tests but is more powerful. This brings us to the essence of the question: Should older generation approaches and tools be used at all? The evolution of test automation was largely driven by the need that automated testing would be easier to use and that there would be less maintenance effort. This is what keyword-driven being 'more powerful' really means. Supporting an older generation means allowing people to use an approach with greater risks to the project in terms of time-to-market, cost and quality.
Another issue becomes clear when we consider the analogy with construction. Houses are commonly built using durable materials like concrete, bricks and glass. They used to be built with straw, clay and wood. And if these other materials result in a house that suits your needs, then it is still fine to use them. But using clay will not result in a house that meets the needs of most people. So we do not use it. Neither will we combine materials like straw and clay with concrete and bricks, for another reason: Even assuming that the materials can be combined at all, using them will result in a more complex structure that is harder to fully understand and therefore harder to design, build and maintain. The same goes with automated testing: The newer, keyword-driven approaches and tools are much more effective. A hybrid approach or tool is perhaps more of a project risk than a good thing.

5) Conclusion

In fourth generation the typical reduction in maintenance is up to 70% over second and third generation test automation solutions. In fifth generation you are reducing your maintenance by up to 100% since there is no need to maintain the action word functions (scripts). However there is still the need to maintain the test cases themselves. The action word functions are simply built on the fly and deleted during teardown, when the test is complete. Fifth generation provides a superior automation framework with which test teams can automate the testing of applications with minimal code and less maintenance across the product cycles. It is the combination of the "**Standard Action Words**", "**Code Templates**" and "**Code Generation**" concepts that makes fifth generation scriptless. "Poor software architecture is one of the biggest factors in test automation project failures"[iv].

6) Appendix

6a Test Automation Progression

Hans Buwalda is the primary inventor of TestFrame and the action word technique. Some explanation of inferred principles (test automation progression);

- 1_{st} Generation – Record and Playback
- 2_{nd} Generation – Use/reuse of functions in test scripts
- 3_{rd} Generation – Data Driven scripts/functions
- 4th Generation – Action Based scripts/functions
- 5th Generation – Scriptless automation.

6b Third Party Tools Explained

HP-Mercury has different functional test tools based on the clients needs. In the beginning they created XRunner for Xwindows (UNIX) record and playback (1992). In regards to test automation progression, their main record and playback tool is now known as QuickTest Professional (QTP). Within QTP there are generic keyword capabilities along with data driven features. For example you can create a basic keyword and QTP has a built-in excel spreadsheet to data drive parameters in your keyword. Note all other solutions use Microsoft's Excel outside the tool. For example the current version of Logica's TestFrame has a toolbar add-in for Excel and the 3rd party test tool is agnostic. Embedding Microsoft Office in your application is a poor design decision. This is simply because you are adding another layer of overhead. HP-Mercury has a test management tool originally known as TestDirector (1994) now known as Quality Center (QC). HP-Mercury's fourth generation test automation solution or TestFrame equivalent came out in 2004 in the form of Business Process Testing (BPT). BPT is a Quality Center add on that requires QTP. Within BPT an analyst can develop "business components" the TestFrame equivalent of an action word. Note that in QC 10.0 configuration management is now built-in (see **Figure 4**).

QTP invokes Business Components
(Business specific action, re-usable
test automation functions partitioned
by business process)

QuickTest Professional (QTP)
10.X
(R&P Test tool)

Built-in
Spreadsheet

Business Process Testing (BPT)
10.X
(4G Business process modeling
tool)

BPT
Resources

Custom
Libraries

QC Invokes Test
Case from QTP

Business driven
test case on the
system-under-test

Quality Center
10.x
(Test Management)
Test Results are uploaded back
in Quality Center

Built-in
Configuration
Management
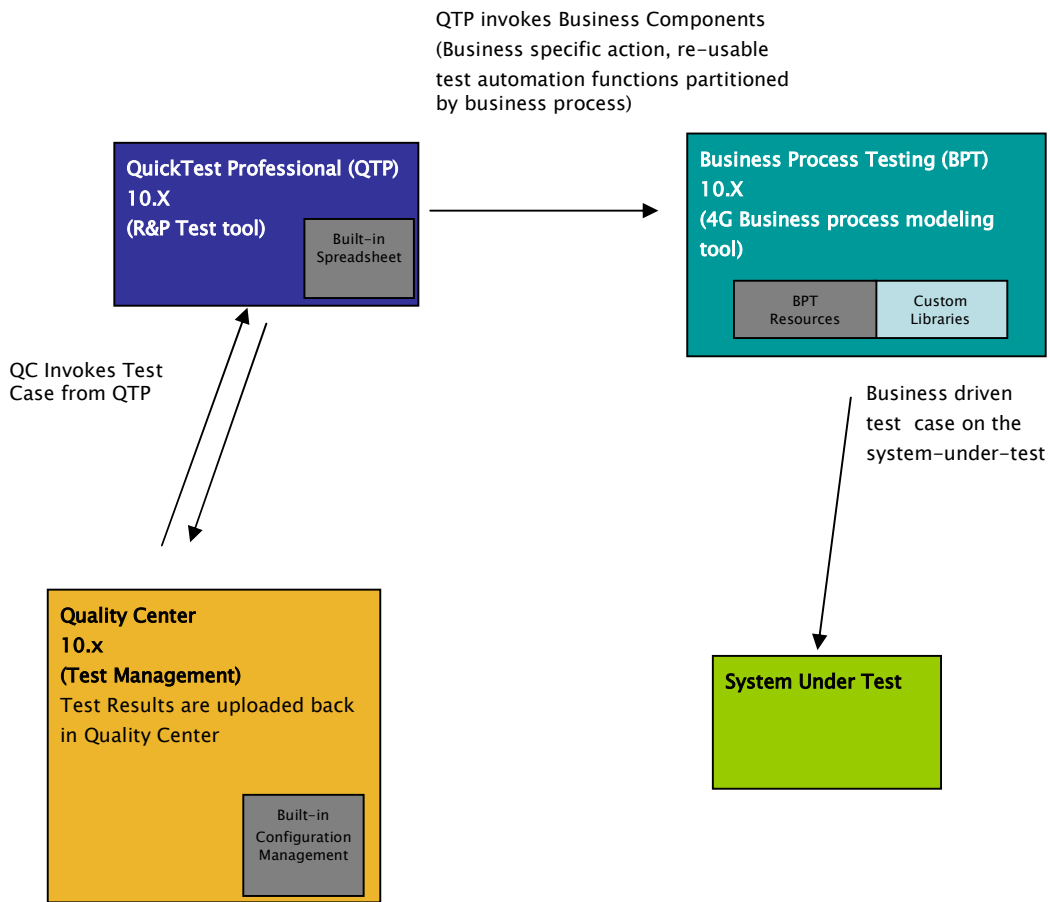
System Under Test

Figure 4 - HP-Mercury QC/BPT/QTP Setup

6c Code Templates

```
'START TEMPLATE
'@ GENERATED HEADINGS
Function GuiEntry() as Long

        'Declare Variables.
'@ GENERATED CONSTANTS - DECLARATION
        Static FunctionName                     As String
        Dim ObjectNames(MAX_PARAMS)             As String
        Dim ObjectTypes(MAX_PARAMS)             As String
        Dim CheckDescriptions(MAX_PARAMS)       As String
        Dim FirstParm                           As Long
        Dim LastParm                            As Long
        Dim x                                   As Long
        Dim StartOfGrid                         As Long
        Dim EndOfGrid                           As Long
        Dim StartTime                           As Long
        Dim apiCall                             As String
        Dim ObjectError                         As Long
        Dim returnValue                         As Long

        'Initialize Variables.
'@ GENERATED CONSTANTS - INITIALIZATION
        FunctionName                            = "GuiEntry"
        FirstParm                               = gFirstParm
        LastParm                                = 0
        StartOfGrid                             = 0
        EndOfGrid                               = 0
        StartTime                               = 0
        apiCall                                 = ""
        ObjectError                             = FALSE


        tf_FunctionEntry(FunctionName, StartTime)

        ' set to appropriate Tab (If any)
        If (TabName <> "") then

                        ObjectError = tf_SetTab(TabName, TabWindow, apiCall)

                        If(ObjectError > 0) then
                                tf_CallStatus(FunctionName, ObjectError, apiCall)
```

```
                                tf_PrintF1Error(FunctionName, TabName, "Could
not set Tab!")
                        End If

        End If

        ' check to ensure we still want to execute Action Word
        If      (tf_CriticalChecks(WindowName) = FALSE) then

                returnValue = FALSE
                goto EndFunction

        End If

        ' set the Object Name and Type (usually the "class")
        x = gCount + FirstParm

        ' Start-of field object, type, description
    '@ GENERATED CODE NON-GRID
    '@ GENERATED CODE GRID
        ' End-of field object, type, description

        ' Set LastParm
        LastParm        = (x - 1) - gCount

        ' Find the StartOfGrid (If any)
        tf_CalculateGrid(FirstParm,    TableName,    LastParm,    ObjectNames,
ObjectTypes, CheckDescriptions, StartOfGrid, EndOfGrid )

        ' Set Values into Objects
        tf_StandardSet(        FirstParm, LastParm, ObjectTypes, ObjectNames,
CheckDescriptions, StartOfGrid, EndOfGrid)

        ' If Acknowledgement message appears,  press MsgButton
        If (MsgWindow <> "") then

                If (tf_ExpGuiWin(MsgWindow ,30) = TRUE) then

                        '# press the button (If one was requested)
                        ObjectError = tf_PressButton(MsgButton, apiCall)

                        If(ObjectError > 0) then

                                tf_CallStatus(FunctionName, ObjectError, apiCall)
                                tf_PrintF1Error(FunctionName, MsgButton, "Could
not press button!")
```

```
                    End If  '(ObjectError > 0 )

             End If '(tf_ExpGuiWin(MsgWindow ,30) = TRUE)

       End If  '(MsgWindow <> "")

       tf_FunctionExit(FunctionName, StartTime)
       returnValue = TRUE

       EndFunction:

  '@ GENERATED CODE - RETURNVALUE

  End Function
  '@ GENERATED FOOTERS
  'END TEMPLATE
```

Figure 5 - Example Code Template GUI Entry

7) References

As a statement of legitimization I suggest reading "Automate your testing, Sleep while you are working" from Maurice Siteur of Capgemini NL[v]. In chapter 9 Maurice suggests there are 9 levels of use regarding capture and playback tools:

- Level 1 – Capture and Playback (using only the functionality of the tool);
- Level 2 – Capture and Playback (modifying for data-driven testing);
- Level 3 - Capture and Playback (modifying for continuity);
- Level 4 – Data-driven testing (pure);
- Level 5 – Data-driven testing (adding information to the data file);
- Level 6 – Test Application (action words);
- Level 7 – Test application with test design;
- Level 8 – Complete test application;
- Level 9 – Test Generation.

One good source of information is Test Automation, From Record/Playback to Frameworks, from John Kent of Simply Testing at EuroSTAR 2007. John is very knowledgeable regarding test automation code generation. John Kent in his series of test automation articles provides evidence that in large systems "we need to automate the building of the automation code[vi]".

7a Fifth Generation Frameworks

LogiGear

Hans Buwalda invented fourth generation test automation frameworks in 1994 while at the CMG test research centre. We consider Hans the godfather of fourth generation. It is Hans' reference that we use to describe the generations of test automation progression. LogiGear extended the TestFrame Engine and now has a fifth generation test automation tool and framework known as TestArchitect. Note that TestArchitect is based on the original TestFrame Engine5 code. TestArchitect has been implemented in hundreds of clients globally.

Logica

CMG created the original TestFrame Engine and framework in 1994. The TestFrame Engine also known as Engine5 is essentially a controller written in C++ and implemented as a Microsoft Windows .dll. The TestFrame engine is very portable and works with any 3rd party tool and code. You can use anything from HP-Mercury's QuickTest

Professional or Microsoft's Visual Basic. TestFrame is not a tool, and a 3rd party record and playback tool typically needs to be provided to recognize the system under test. The TestFrame engine is what we call a DIY kit. It does not generate code and all the action word functions have to be manually coded by a test automation developer. TestFrame and the action word technique are copyright of Logica. Currently all the testing brains behind TestFrame and all the testing books have left the company except for Chris Schotanus. Recently Chris has updated and published the next edition of the TestFrame book. The TestFrame STM has been implemented in over 1000 testing projects globally. Please refer to the whitepaper written by Hans Buwalda and Maartje Kasdorp "Getting Test Automation Under Control[vii]" for examples of action words.

Software Development Technologies (SDT)

SDT originally was the de-facto North American arm of the CMG testing practice. SDT and CMG partnered on testing projects in NA and Europe. SDT received the original TestFrame Engine5 code as a result and rebranded it Unified TestPro in North America. They added Christopher J. Schärer's action word management and automated build facility. Eventually support was added for Rational Visual Test and the object recognition technology is now part of the tool. A maintenance facility in Hyderabad, India was added and the framework was extended to the current 5G level. There are many new components including a test management facility. SDT has even installed a custom version of the tool at Microsoft in Visual Basic. Unified TestPro has been implemented in hundreds of clients globally.

Worksoft

Worksoft is a company created by Linda Hayes originally of the record and playback tool company known as AutoTester. Worksoft has created what they consider a "codeless" [viii] test automation tool and framework known as Certify. Linda Hayes has accumulated code libraries over the years on hundreds of engagements with clients. The result is that many of the necessary generic functions to test an application have already been implemented by the Worksoft team. Worksoft doesn't like the term "keyword" as it has record and playback connotations. There are some advanced technologies in place here regarding test automation. They definitely have scriptless technology from the test analysis point of view and fifth generation capability.

7b Code Generation

IDT – Innovative Defense Technologies

Elfriede Dustin has been doing some advanced research into the latest in automated software testing technologies. IDT has created a Test Automation Framework (TAF) that is based on the original now open source STAF framework. It is difficult to discern which generation of framework to classify the IDT TAF. However since STAF is technically fourth generation it starts there, and includes some leading edge components

for automated test code generation. The IDT framework is based on all open source component technologies.

Google Tech Talk:
GTAC 2008 Advances in Automated Software Testing Technologies.

Book:
Elfriede Dustin, Implementing Automated Software Testing (March 2009) ISBN-9780321580511.

Simply Testing

Simply Testing is a company that has been in the software testing industry since 1995. John Kent has written about advanced test automation frameworks and test automation code generation in Professional Tester magazine and at EuroSTAR. Simply Testing's line of products includes a framework, the Advanced Test Automation Architecture (ATAA) and a test automation code generator TestCoda.

7c Non Legacy Tools

Ranorex

Ranorex is Software Development Company that provides the next generation of test automation tools. When we say non-legacy, legacy means record and playback, or the use of libraries that can be included in your application. The tools are based on the .net platform and provide support for multiple technologies.

QualiSystems

QualiSystems is test automation pioneer founded by Aryeh Finegold of Daisy Systems and Mercury Interactive fame. We consider Aryeh the Godfather of test automation tools. Developers of the TestShell product suite, QualiSystems has taken the non-legacy approach to test automation tools as well. There is also some very unique functionality here along with scriptless test automation. QualiSystems' TestShell is an end-to-end test automation solution compatible for testing virtually any hardware, device or embedded system. TestShell provides code free test authoring, automatic test execution, and comprehensive test reporting and analysis.

8) Acronyms and Definitions

Action word – an action word is Logica's version of a keyword within a test automation framework. Action words are business specific actions developed by a test analyst then implemented by a test automation developer. Action words are used for both manual and automated testing.

AutoTester – A company that produced the first commercial test tool for the PC (on MS-DOS). Their tool was also named AutoTester and eventually worked on Microsoft Windows as well.

BPT – HP-Mercury's fourth generation test automation framework and tool.

Capture and Playback – an inference to test automation tools where a test analyst/engineer records the required actions for a specific application/system under test and saves the steps in a journal file. The script is typically used to model/implement a test case.

CAD – Computer Aided Design.

CAST – Computer aided software testing.

Certify – Worksoft CTO Linda Hayes latest codeless functional test automation tool and framework.

CMM – is a concept that was developed in the field of software development and which provides a model for understanding the capability maturity of an organization's software development business processes.

Codeless Automation – Worksoft's description of their test automation tool (Certify) and framework.

CVS – Open source versioning tool, Concurrent Versioning System.

ECAD – Electronic Computer Aided Design.

Hybrid Test Automation Frameworks – hybrid test automation frameworks, are a confusing term where some practitioners in the field of software testing are declaring frameworks that incorporate any combination of data driven, keyword driven, and library frameworks to fall under this category.

IDE – Integrated Development Environment.

JUnit – Open source test automation framework where the focus is primarily unit testing of JAVA applications.

MBT – Model Based Testing.

MTPF – Minimal Test process Framework.

QTP – HP-Mercury's functional record and playback test automation tool.

Quality Center – HP-Mercury's test management tool.

RBT – Roles Based Testing.

Record and Playback – see Capture and Playback.

Scriptless Automation – Advanced Test Automation technology where test automation scripts are built automatically based on a set of predefined rules and parameters such as GUI object libraries and generic code templates.

RUP – The Rational Unified Process.

STAF – Open source keyword/data driven test automation framework written by Carl Nagle of the SAS Institute loosely based on TestFrame.

Structured Test Method – A mature testing process with a well developed testing life-cycle including structured test management and sometimes risk based. Typically associated with TestFrame and TMap. Note that RUP and MTPF are not risk based.

TAF – test automation framework.

TestArchitect – LogiGear CTO Hans Buwalda's latest fifth generation functional test automation tool and framework.

TestFrame – Logica's structured test method and fourth generation test automation framework. Invented by CMG of the Netherlands.

Testware – Everything that is required for testing that can be reused at a later stage, I.E. test case test scripts etcetera.

TAKT – Sogeti's test automation framework, test automation knowledge and tools.

TSL – HP-Mercury's test script language, a C language derivative used in the WinRunner R&P tool.

TQM – Total quality management is the organization-wide management of quality.

TMap – Sogeti's structured test method invented by IQUIP of the Netherlands, test management approach.

Unified TestPro – Software Development Technologies latest fifth generation functional test automation tool and framework.

For further information about Advanced Test Automation Frameworks, please contact Jeff at TESTars, send an email to jeff@testars.net or Martin at martin@deanalist.nl .

Copyrights:

ATAA and TestCoda are copyright of Simply Testing

Certify is copyright of Worksoft.

The Ranorex tools and suite are copyright of Ranorex.

TestArchitect is copyright of LogiGear

TestFrame and action words are a copyright of Logica.

TestShell is copyright of QualiSystems.

QuickTest Professional, Business Process Testing and Quality Center are trademarks of HP-Mercury.

Unified TestPro is copyright of Software Development Technologies.

[i] Test Automation Awareness, Jeff Hinz, LogicaCMG, 2007
[ii] Integrated Test Design and Automation Using the TestFrame Method, Buwalda Et al., 1999, ISBN - 0201737256, Rotterdam
[iii] Unified TestPro Scriptless Technology, Vinay Thandra, SDT
[iv] Test Automation, From Record/Playback to Frameworks, John Kent, Simply Testing, EuroSTAR 2007
[v] Automate your testing, Sleep while you are working, Maurice Siteur, 2005, ISBN – 9039524424, Den Haag
[vi] Ghost in the Machine, Part 5, John Kent, Professional Tester 2004
[vii] Getting Test Automation Under Control, Hans Buwalda and Maartje Kasdorp, CMG, 1999
[viii] Where Did The Code Go?, Linda Hayes, Worksoft

www.testars.com