

Brett Leonard

Indianapolis Workshop on Software Testing

Integrating HP Quick Test Pro and Empirix Hammer

About this Report and Presenter

About this report

This report was presented at the Indianapolis Workshop on Software Testing on September 26, 2008. The title of the workshop was “Testers who write code”.

About the presenter

Brett Leonard is the author and presenter of this report. He is a Senior QA Manager at a major financial company in the United States. He has over 17 years of experience in the financial services industry and 9 years of experience in software testing. He has a Bachelor of Science degree in Business Economics from State University of NY at Oneonta and has studied Software Engineering at the University of Cincinnati College of Applied Sciences. He currently leads a team that creates automated testing solutions for a variety of technologies such as web, mainframe, client server, database, web services, and voice response systems. His email address is cruisinqa@gmail.com.

Project Background and Testing Challenge

Project background

A company is engaged in a multi-year, multi-million dollar effort to upgrade their call center software application. The telephony functionality that sends a callers account information, authentication status, and other details to a phone representative (from the Voice Response System) is an integral component to the new software.

Testing challenge from Senior Management

Originally, automation of telephony was considered out of scope; however, the Senior Vice President of Technology Risk Management expressed concern due to the number of regression test cases requiring the functionality and challenged the QA Automation team to find a solution.

Two possible responses to the challenge

Yes, there is always a way to technologically solve a problem and there is a solution to this one. The critical question is whether the cost of implementing the solution exceeds the benefits derived. That question can be answered by further research and proof of concepts.

No, it is dangerous to suggest innovative approaches to Senior Management because you can pay a political price if you don't find a solution.

Lesson One Take the Dive

**** Lesson 1 **** Take the dive!! Take the risk and suggest new, creative, and potentially effective solutions (even to Senior Managers) but cover yourself politically by suggesting further research, proof of concept, and a cost/benefit analysis may be necessary before committing to a solution.

A quote to keep in mind *Anything the human mind can conceive, and believe, it can achieve –*
Napoleon Hill

Technology corollary 1 Technology solutions are restricted by the triple constraints of time, money, and scope.

Technology corollary 2 Some technology solutions may not be the best to solve the problem. There is a great risk of over engineering solutions – sometimes a simple solution is the best one.

Lesson Two Be Prepared and Use Effective Effort

**** Lesson 2 ****

Know your requirements and constraints, conduct your own automation research and development, form a vision using your knowledge, and intuition. Use effective effort to break through barriers. Don't listen to nay-sayers if you believe you are right, even supposed experts.

Our requirements

Telephony was to be invoked by Quick Test Pro.
Process needed to be integrated with Quality Center.
A number of interactions needed to be covered (authenticated, unauthenticated, wrong personal identification number, wrong customer ID, etc...)

Our constraints

No budget to buy additional software.
Initial implementation to be done within a month.
Needed to be implemented with existing resources.
Development team was simultaneously working on a solution.

**My vision
(Before investigation)**

Use Quick Test Pro to invoke a command line utility to kick off Call Master script on Hammer server.
Call Master script simulates interaction with user.
Quick Test Pro waits for user interface to indicate that call was received.
Quick Test Pro proceeds with test on desktop and reports results to Quality Center.

What is Effective Effort?

Components of Effective Effort

Tenacious engagement – Working hard. Working through difficulties.
Not giving up.
Intense focus on data and feedback – Paying close attention to information that indicates how well you are doing, thinking about what it will take to get better, what you must work on to improve.
Ongoing strategy formulation based on feedback – Listening to what the feedback is telling you and changing your approach accordingly by trying new ways to improve.

Continued on next page

Lesson Two Be Prepared and Use Effective Effort, Continued

Examples of Effective Effort

Component	Example
Tenacious Engagement	The expert user for Call Master/Hammer tool had doubts about the possibility of invoking scripts through a command line. My intuition told me that there must be a way. After doing some research and talking to the vendor it became apparent there was a way.
Intense focus on data and feedback	After installing the command line utility on my desktop, I was not able to invoke the script on the Hammer box. The feedback (error) told me there was something wrong with the network connection. The vendor told me to call a network administrator. The admin did not help but recounting the story to him gave me an idea – to check the Secured Sockets Layer (SSL) service on the server – after I turned it on I was able to invoke the script.
Ongoing strategy formulation based on feedback	The original vision assumed that a command line could invoke an existing Call Master script. After additional research it was discovered that new Hammer scripts needed to be created and Call Master could not be used.

Lesson Three Find Support for your Cause

**** Lesson 3 ****

Vendor product support representatives can be great allies but don't waste their time.

How to waste a support reps time

Don't read the manual before you call.
Don't read other resources on the support web site.
Don't read the support forums.
Expect the rep to solve your problem for you.

How a support rep can help

Confirm your vision.
Become an interested party and give you specific strategies.
Assist troubleshooting issues during proof of concept phase.
Provide ad hoc training

How a support rep helped us

Confirmed that there should be a way for us to achieve our goal (although he had not heard of anyone doing it before).
Confirmed that the software (command line utility) I identified through my initial research would do the job.
Provided us with additional documentation.
Helped us put together an initial command line string for our proof of concept.
Gave me ad hoc training on the functioning of the Hammer server software.
Gave me a small piece of code to start building our interaction model with the voice response system.

Lesson Four Prove Yourself

**** Lesson 4 ****

Proofs of concepts are essential to the development of new, creative, effective (innovative) solutions.

Advantages of successful Proof of Concepts

People with doubts can be won over.
People with little interest can become advocates and help implement the final product.
Opens minds and allows the creative process to move forward.
Uncovers risk involved in proposed solutions.
Forces others to take your work seriously.

Our proof of concept

1. Baby steps – Establish communication link to Hammer server. Use command line to invoke script that calls a cell phone.
 2. Integrate with QTP – Create a method through which the script in step one can be invoked through Quick Test Pro.
 3. Interact with voice response system – Create a script on the Hammer server that forces a call and account information to be sent to a phone representative.
 4. Put it together – Use QTP to invoke the script that sends a request to the Hammer server to initiate the call interaction and watch the call and account information come through.
-

Lesson Five Documents Sell

**** Lesson 5 ****

Document your findings, create a process, and get others to help. You can't do everything yourself. You need to socialize your solution throughout your team or anyone else that can help. Good documentation will give people the information they need to become familiar with your solution.

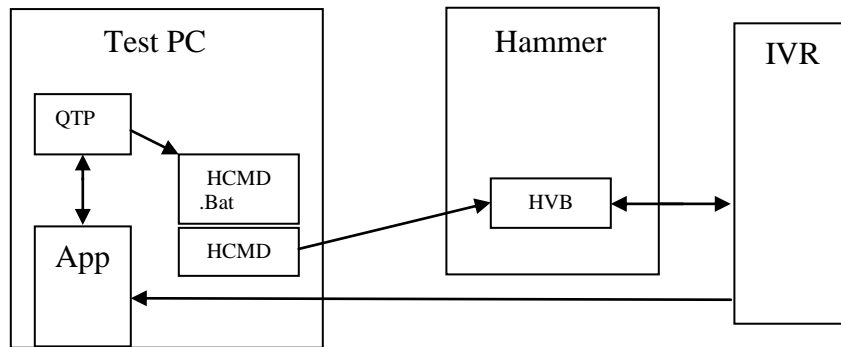
**Documents
be fancy.**

A good text readme document can be worth its weight in gold. Wikis are another good way to document new processes.

**The basic steps
(for automation
engineer)**

- Install HCMD application on desktop with QTP (TB_HCMTD_3_0_1.exe) it will install tcl and ask you to reboot
- Place batch (.bat file) in Temp directory (C:\TEMP)
- Open QTP and run the CallPhone script

**Test process
and
architecture**



**Process
description**

The following chart describes the test process.

Step	Action
1	QTP invokes the App and performs steps to get it into a state where it is ready to receive a phone call.
2	Call Phone action is called that invokes the HCMTD.Bat file. QTP waits for action on the screen to determine if call has come through.

Continued on next page

Lesson Five Documents Sell, Continued

Process description, (continued)

Step	Action
3	.Bat file sends command to the Hammer server to start a HVB script.
4	HVB script calls IVR and performs interaction to default to a representative.
5	HVB script gives IVR the extension of the test PC and the phone call transfers to the App.
6	QTP detects screen action and continues with test.

.Bat file

```
HCMD Start Script -mc HammerServerName -h HammerServerName -c 3 -n
C:\Hammer\libmgr\private\hammer\scripts\survey_params.sbl -i
4#s#PhoneNumber#%1#s#TFN#%1#s#SSN#%1#s#RepExt#%1 -sa 0
exit
```

QTP code

```
' Build command string
cmdStr = "cmd /K C:/TEMP/hcmd_params.bat 18661111111 8771111111#"
& " " & SSN & " " & "999"
Set objShell = CreateObject ("WScript.Shell")
objShell.Run cmdStr, 0, False
Set objShell = Nothing
```

Sample HVB code (.sbl)



```
J:\WORKGRP\QA\
Automation_Team_D
```

Appendix I

HCMD Start Script Command

Information about scheduling a Hammer Visual Basic test script through the HCMD command line interface.

Syntax

HCMD Start Script -mc masterController -h server -c channels -n testName
 {-i nbrInputVars[#varType1#varName1#varValue1] ...
 [#varTypeN#varNameN#varValueN]}
 [-l loop] [-ld loopDelay] [-ma maxActive] [-st staggerType] [-sv
 staggerValue]
 [-sm staggerMax] [-sa scheduleAction] [-t startTime] [-d startDate]

Parameter	Definition
-mc <i>masterController</i>	Name of the Master Controller on which to schedule the test.
-h <i>server</i>	Network name of the Hammer server whose channels the test runs on.
-c channels	Channel numbers the test runs on.

Continued on next page

Appendix I, Continued

Syntax, (continued)

Parameter	Definition
-n testName	Name of the .sbl HVB test file. You must include the full pathname. If the file name includes spaces, enclose it in quotation marks.
-i nbrInputVars	<p>Optional. The number of script input variables. For each variable N in your script, enter its type, name, and value using the varTypeN, varNameN, and varValueN parameters.</p> <p>Enter a # symbol to separate the number of input variables and each set of type, name, and value parameters. For example, to specify two input variables, enter:</p> <pre>-i 2#i#myInteger#5#s#myString#Good Morning</pre> <p>Input variables must be declared as input variables inside the script in which they are used. If input variables do not appear for a script in which you expect them, the input variables may not be properly defined. See Using Input Variables.</p>

Continued on next page

Appendix I, Continued

Syntax, (continued)

Parameter	Definition
varTypeN	Required if -i nbrInputVars is specified. For input variable N, the variable type. Valid settings are: i - Integer s - String f - Float
varNameN	Required if -i nbrInputVars is specified. For input variable N, the name of the variable.
varValueN	Required if -i nbrInputVars is specified. For input variable N, the value of the variable.
-l loop	Optional. Number of times to loop the script. If you do not specify this option, the script runs once. Specify - 1 (minus one) to loop indefinitely.
-ld loopDelay	Optional. Time in milliseconds to pause between loops. Default = no limit.
-ma maxActive	Optional. Maximum number of active connections. 0 for no limit. Default = 0.

Continued on next page

Appendix I, Continued

Syntax, (continued)

Parameter	Definition
-st staggerType	<p>Optional. Stagger is the time to wait between the start of a test on a channel and the start of the same test on the next channel. Valid settings are:</p> <ul style="list-style-type: none"> 0 - User-defined 1 - Random 2 - Automatic 3 - None (the default)
-sv staggerValue	<p>Optional. If staggerType is 0, the stagger value is the stagger time in milliseconds. If staggerType is 1 or 2, the stagger value is the minimum time of stagger in seconds. The -sv staggerValue parameter is not used with staggerType 3.</p>
-sm staggerMax	<p>Optional. The maximum time of stagger in seconds. Used only with stagger type 1.</p>
-sa scheduleAction	<p>Optional. Defines what action to take if a channel is busy when a test attempts to start. Used when you schedule a test to start at a future time by using the -t startTime parameter.</p> <p>Values are:</p> <ul style="list-style-type: none"> 0 - Wait for the channel to finish what it is doing, then start the scheduled test 1 - Immediately stop what the channel is currently doing so the scheduled test can start

Continued on next page

Appendix I, Continued

Syntax, (continued)

Parameter	Definition
-t startTime	Optional. Defines the start time for a test. The default is the current time;the test starts immediately unless a different start date is defined. Format is in military time hh:mm.
-d startDate	Optional. Defines the start date for a test. The default is the current date;the test starts today either at the current time or the time defined by the -t startTime parameter. Format is mm/dd/yyyy.

Example

The following command schedules an HVB test script on a Master Controller named Hammer. The test runs on channels 1 to 10 on the server Editt. The test file is in the C:\Tests folder and is named My_Test.sbl. The test takes two input variables: the integer called myInteger set to 3, and the string called myString set to helloWorld. The test is set to loop indefinitely with a loop delay of 1500 ms. The start time for the test is 1:15 PM on January 10, 2007. If a channel is busy when the test starts, the test waits.

```
HCMD Start Script -mc Hammer -h Editt -c 1-10 -n c:\Tests\My_Test.sbl
-i 2#i#myInteger#3#s#myString#helloWorld -l -1 -ld 1500 -sa 0 -t 13:15 -d
01/10/2007
```