Manager's Guide to GUI Test Automation

Yury Makedonov, CGI

Software Test & Performance Conference,
New York, November 03, 2005

CGI

Target Audience

- This presentation is for everybody who makes decisions regarding GUI test automation:
  - Managers of functional test teams
  - Managers of GUI test automation teams
  - GUI test automation specialists

CGI

Agenda

1. Introduction of mainstream GUI testing tools
2. Major test automation approaches/frameworks
3. Test automation process
4. Managing GUI test automation

Backup slides:

1. Alternative GUI testing tools/frameworks
2. Offshore outsourcing of GUI test automation

Automated Testing vs. Regression Test Automation

- "Automated testing" = testing by a computer;
  In this case the design of a test procedure and identification of expected results are being performed by a computer.
  - "Dumb Monkey" for Rational Visual Test
- This presentation is about:
  "**Regression GUI test automation**"

  "Regression GUI test automation" = automation of the execution of a manual functional test which was developed and documented by a human
- Functional test scripts have to be created before the start of GUI test automation.

GUI testing tools

1 – Introduction of GUI testing tools

GUI Testing Tools to be discussed

- The most efficient GUI test automation tools are the ones from major vendors:
  - Mercury WinRunner and QuickTest Pro,
  - Segue Silk,
  - Compuware QARun,
  - IBM Rational Robot,
  - etc.
- These tools are used by independent testing teams to automate functional test cases.
- The focus of this presentation is on the efficient implementation of these tools

Myth #1 – Commercial GUI testing tools are expensive.

- Under the influence of this myth some people:
  - try to develop their own testing tools to save money,
  - use scripting languages like Perl and Ruby to automate functional test cases and
  - do not consider test automation at all.
- GUI testing tools are indeed much more expensive than such development tools as Microsoft Visual Studio or Borland JBuilder,
- But are they really expensive?

High prices of GUI testing tools are just a myth!

Reality – GUI testing tools are cheap

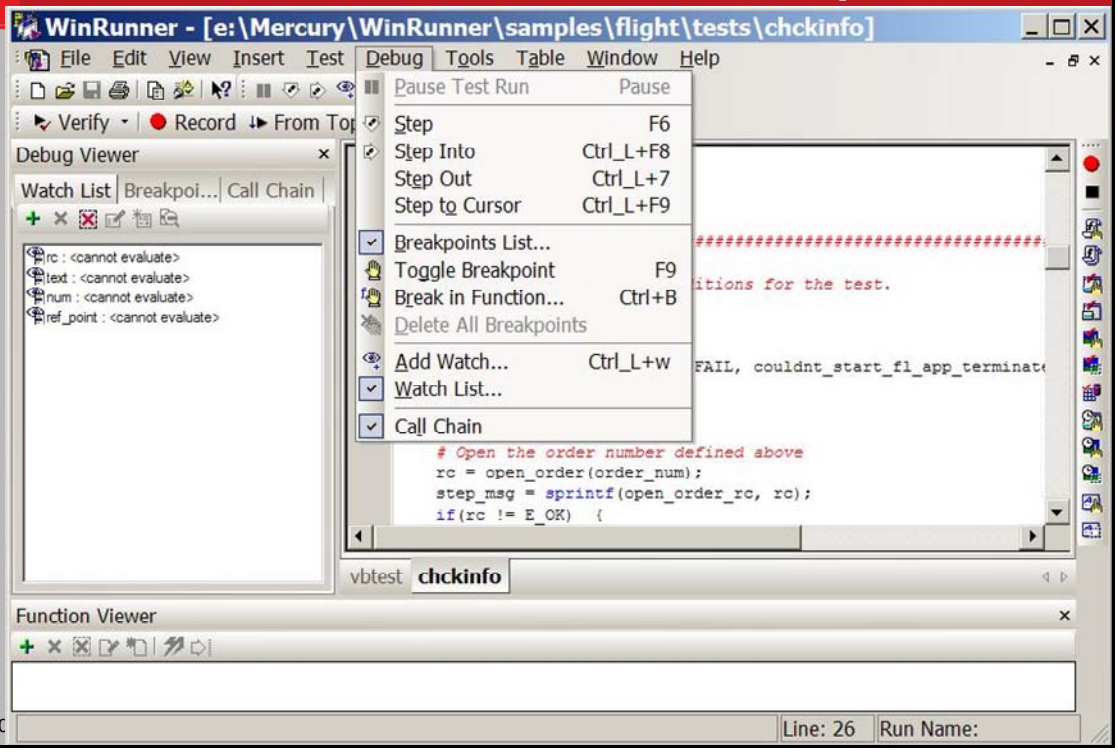
- Per seat license for one of the most expensive GUI testing tools is about $8,000.00.
- This tool will be used for two years (and then thrown away or put on a shelf). Cost of this tool is $4,000.00 per year.
- Cost of test automation person with overhead is $80,000.00 per year.
- The cost of a GUI testing tool is just 5% of the cost of the person who will use it.
- The increase of productivity of a person who uses these tools is more than 5% compared to a person who uses other alternatives.

Record and Replay Tools

Record and replay tools:

- Tester executes manual test script while GUI testing tool records all his actions:
  - Mouse clicks
  - Key presses
  - Pauses/delays
- Then GUI testing tool replays the script exactly as it was recorded, including:
  - Timing of script execution
  - Data entered into different fields

GUI test automation tool – an example



GUI test automation approaches

2 – GUI test automation frameworks

Record and Replay test automation – Pros

- The record and replay approach is relatively simple and fast
- The record and replay approach might be useful when you change something beyond GUI, e.g.:
  - Back-end configuration parameters,
  - Back end OS,
  - Version of back-end product
  - etc.,

Record and Replay test automation – example

- Example of successful implementation:
  - Verification that online bill looks exactly the same after a new version of a back end Bill Presentment system is implemented.
- Vendors were extremely sensitive about the exact appearance of their bills:
  - Corporate logos should be valid up to a pixel,
  - The layout of an electronic bill should be pretty close to the paper version; it should be exactly the same regardless of all changes to the back end code.
- The same user IDs, account numbers and bill amounts were used for every test run.

13 Software Test & Performance Conference November 3, 2005, New York City

Record and Replay test automation – Cons

- Only one functional test case can be automated by one automated test script.
- What about changes in:
  - GUI
  - Target URL,
  - User ID/Password
  - Business data (e.g. price, order number, invoice number, posting date, etc.)?
- When anything changes (new version of a product, new build, bug fixes, patches, test data, etc.) – the script has to be rerecorded!
- **Poor maintainability**
  Very limited usefulness

14 Software Test & Performance Conference November 3, 2005, New York City

Record and Replay test automation – conclusion

- The success story I described above (verification of online bills) might be only an exception.
- Applicability/utility of “record and replay” approach is very limited.
- Typically I recommend it only for training purposes.

Test automation approaches / frameworks: Functional Decomposition – a challenge

- Let’s assume that you have 27 scripts.
  Each script starts with the same statement:
  ```
  web_browser_invoke(IE, http://confut.cell.ca);
  ```
- You need to execute these scripts against different environments, e.g. QA, Mirror, Staging
- What would you do with a “classical” record and replay approach?
- You develop a script for every target URL, e.g.:
  - Verify_Access_QA,
  - Verify_Access_Staging,
  - Verify_Access_Mirror

The only difference between them – the initial URL!

We have uncontrolled multiplication of test scripts!

Test automation approaches / frameworks: Functional Decomposition – an example

- We start with the recorded script
- Then we introduce ***functions***
- We remove this statement from all scripts:
  `web_browser_invoke(IE,http://confut.cell.ca);`
- And substitute it with a function:
  `UT_Site_Open();`
- This function is described only in one place.
- When we need to execute these 27 scripts against another environment all we have to do is change the code in just one place!

Test automation approaches / frameworks: Functional Decomposition – sample code

Main test script:

```
.....................
 load("Lib_UT_Functions");
 UT_Site_Open( );
 UT_Select_Language("English");
 UT_Login("a327012", "12345");
 .....................
```

Function:

```
function UT_Site_Open( )
{
 #Opens a new UT site in a new browser window:
 web_browser_invoke(IE, http://confut.cell.ca);
 return 0;
}
```


Functional Decomposition – pros

- We can use a “Functional decomposition” approach when:
  - Different test scripts include identical actions (groups of test steps) e.g. “Login”.
- We substitute these repetitive groups in all scripts with ***functions***.
- When script is broken due to any change of GUI or business logic we have to modify script code only in one place.
- **Pros: High Maintainability**
- The functional decomposition method could be adequate when we have only a few parameters which can have only a few different values (e.g. target URL, user ID, browser, etc.).

Test automation approaches / frameworks: Data Driven Approach

- We start with a recorded script
- Then we substitute the recorded values with those retrieved from a data file
- One functional test case corresponds to a record in the data file
- To execute the same script with another set of test data we have to use a different record in the data file.

Test automation approaches / frameworks: Data Driven Approach – sample code

URL	Browser	UserID	Password
http://confut.cell.ca	IE	a327842	123
.....	.....	.....	.....

```
vTestEnvXls = "C:\Test\UTTestEnvironment.xls";

# Retrieve environment variables from a file:
ddt_open(vTestEnvXls);
vURL = ddt_val (vTestEnvXls, "URL");
vBrowser = ddt_val (vTestEnvXls, "Browser");
vUserID = ddt_val (vTestEnvXls, "UserID");
vPassword = ddt_val (vTestEnvXls, "Password");
ddt_close(vTestEnvXls);
```


Test automation approaches / frameworks: Data Driven Approach – an example

- Service address validation:

- Some “location types” were generating errors

Test automation approaches / frameworks: Data Driven Approach – an example

- 44 location types
- Add-hoc test automation
- Script read all address fields from a data file
- One script covered 44 functional test cases
- Script was executed against several builds until this defect was completely fixed.
- This script was also executed several times for regression testing.

Data Driven Approach – conclusion

- We can use a “Data Driven” approach when:
  - The workflow is identical and
  - There are several different sets of test data for the same test procedure (workflow)
- Benefits of a “Data Driven” approach:
  - 1 script automates several test cases and
  - Test data can be prepared/verified by a functional tester or a Business Analyst.
    (everybody can use EXCEL without a training!)
- Pros:
  - **High Efficiency**

Combination of the functional decomposition and data driven approaches

- This is a combination of two previous approaches:
  - Script is broken into functions *and*
  - Test data are being retrieved from a data file
- This is the current industry standard
- Most successful GUI test automation teams use this approach
- Pros:
  - **High Maintainability**
  - **High Efficiency**

Test Automation Process

3 – Test Automation Process

First rule of GUI test automation

- Probably you have already heard many times the first rule of GUI test automation:

  It makes sense to automate a test case when it will be executed about 10 times (against 10 builds/releases of an Application).
- This is a simplified rule.
- It's a good starting point for test planning.
- Software development shop with daily/weekly builds vs. customized implementation of a commercial product.

27 Software Test & Performance Conference November 3, 2005, New York City CGI

"Good enough" test automation scripts

- We can not justify the same quality requirements for a GUI test automation script (which will be executed only 10 times) as for an application which sells in the millions of copies
- We can tolerate more bugs in test automation scripts than in mass market software products
- How can we possibly test an automated test script?
- The failure of a "good enough" test automation script doesn't necessarily mean that a defect of an Application Under Test is discovered
- Often a script fails because of a defect in the test script itself or in a test data set

28 Software Test & Performance Conference November 3, 2005, New York City CGI

De facto standard of test case documentation

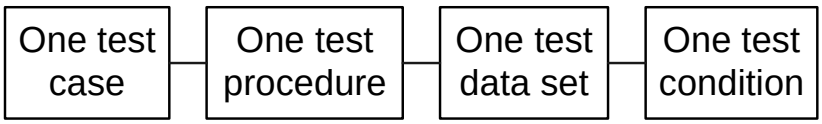
Sample functional test script:

1. Click on “User ID” field
2. Type in “a327012”
3. Click on “Password” field
4. Type in “12345”
   – 5 stars “*****” should be displayed in the “Password” field
5. Click on “Login” button
   – main application screen should be displayed.

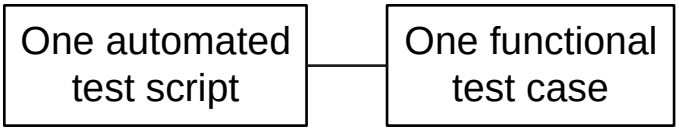
Test procedure mixed with test data!

Current status of test documentation may lead to inefficient test automation

- There is a de facto standard of test documentation which incorporates a test procedure and test data in one so called “test case”:



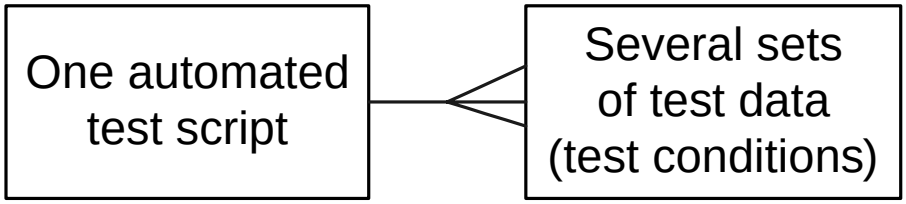
- Such functional test documentation may lead to inefficient test automation:



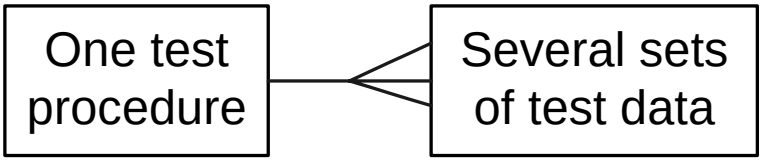
- It’s not a fault of the “IEEE 829-1983 Standard for Software Test Documentation”.

More efficient test automation

- It is more efficient to cover several test conditions with the same automated test script:



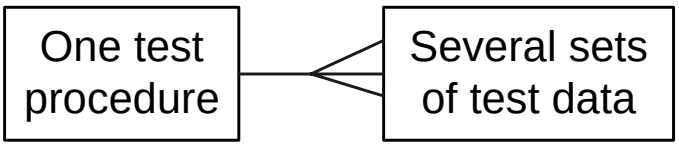
- To use this approach, we need the functional test documentation in the following format:



31 Software Test & Performance Conference November 3, 2005, New York City

Efficient test automation starts with efficient test documentation

- To ensure efficient GUI test automation you have two options:
  1. Convince your functional test team to use a better approach to test documentation *or*
  2. Convert the available set of functional test cases (yourself) into the following format:



32 Software Test & Performance Conference November 3, 2005, New York City

Commercial test management tools and their integration with GUI test automation

- Pros:
  - Provide better “control” and an audit trail for test execution, coverage, etc.
- Cons:
  - Less portable and less maintainable scripts.
  - A lot of network traffic.
  - Typically these tools tend to produce very granular scripts like:
    - Click on a field “…”
    - Type in “…”
    - Click on “Save” button
- Hard to integrate with data driven framework.
- A type of test documentation that leads to Inefficient test automation!

Myth #2 – The Test Automation Process is simple

- A tools sales person will try to convince you that the following is a real GUI test automation process:
  - Record a script.
  - Enhance a script (add functions and/or data driving)
  - Execute scripts
  - Report defects
- This process looks pretty simple and straightforward.

This simple process is just a myth!

Test Automation Process – reality

- Real test automation process:
  - **Choose a test automation approach and develop a test automation framework**
  - **Design test automation scripts (efficient, maintainable, portable, etc.)**
  - Record a script.
  - Enhance a script (add functions and/or data driving)
  - Execute scripts
  - **Analyze causes of scripts' failures**
  - Report defects
  - **Repair failed scripts**

35 Software Test & Performance Conference November 3, 2005, New York City CGI

Myth #3 – GUI testing tools are simple - Every functional tester can use them

- This myth is promoted by the tool sales people.
- This myth assumes the "click, click, click", record and replay approach.
- Under the influence of this myth a test manager proudly can report:
  - "All our testers are using GUI test automation tools".
- Indeed:
  - You do not need a developer's skills to develop a simple test automated script with a "click, click, click", record and replay approach.

But, this simplicity is just a myth!

36 Software Test & Performance Conference November 3, 2005, New York City CGI

Reality – a programming background is required to use GUI testing tools

- The efficient implementation of GUI testing tools typically requires functional decomposition + data driving approach.
- **You need to have some kind of a programming background** to implement this approach.
- GUI test automation scripts are not as complex as a classical C/C++ or Java back-end development.
- Definitely, an experienced software developer or a Computer Science university graduate would make a good test automation specialist.
- At the same time a science or engineering education would be more than adequate for GUI test automation.

Cooperation with functional testers

- The most important success factor is good cooperation with functional testers.
- Typically, test cases are documented with the assumption of enough knowledge of the Application Under Test.
- Test automation folks need a lot of help from functional testers to understand functional test documentation.
- When functional testers understand that the goal of GUI test automation is to help them test, and when they see real results of test automation they become interested in the success of test automation and are willing to provide enough support.

Cooperation with functional testers – an example

Functional testers:

- "Here is a list of functional test cases I am bored to death with executing. Can you automate these test cases?"

Test automation specialist:

1. "I can automate these test cases easily."
2. "These test cases would require several times more effort from me to automate."
3. "These test cases are rather complex and time consuming to automate. I doubt I will be able to do this in the allotted time."
4. "Look! In addition I can also automate some more test cases. It will take very little of my time and will be almost free for you!"

The role of a manager is to make the final selection after considering all trade-offs.

CGI

Managing GUI Test Automation

4 – Managing GUI Test Automation

CGI

Managing GUI test automation - Test Planning

For GUI test automation, we must define:

- The version of the Application Under Test to develop test scripts with
- The target version of the Application Under Test for script execution / maintenance
- The test environment to develop/execute automated test scripts
- A set of regression test cases to automate
- Test data (User IDs, accounts, invoices, etc.)
- When to start (GUI should be stable)

Managing expectations
Reality vs. 'Click, click, click" approach

- Higher managers typically believe in the simplistic "Click, click, click" approach that does not include some phases of the real test automation process.
- You have to educate them and explain real phases and real tasks that are being implemented.

Phases of a GUI test automation project

- At the beginning of a GUI test automation project:
  - Tool selection / Proof of Concept
    (Integrated vs. standalone? Can this tool drive your GUI?)
  - **Development of a basic framework**
- For each version/release:
  - **Design of test automation scripts**
    (efficient, maintainable, portable, etc.)
  - Development (and debugging) of automated test scripts
- For each build:
  - Execution of automated scripts / **analysis of results** / reporting of anomalies/defects
  - **Maintenance of automated scripts - fixing of defects in scripts or test data.**
- **Assign all phases of a test automation process to the same person / team!**

Test environment and test data

- **Pay attention to the test environment and test data during the whole process from development of test scripts to execution against consecutive versions/builds of an Application to be tested.**
- In a simplified case you will be able to use the same test environment and the same set of test data for the whole test development and test execution process.
- Otherwise you have to understand how these scripts will be executed against another test environment when:
  - The old set of User IDs/passwords is no longer available,
  - Old business data (account numbers, invoice numbers, etc.) are no longer available.

Test automation metrics – script development

- How can we track progress of test automation?
- During the development of automated test scripts use **The number of functional test cases which were automated** as a metric.
  This would lead to the implementation of efficient automated scripts.
- Do not use the number of developed **test automation scripts** as a criteria of progress.
  This would lead to development of zillions of small simple inefficient automated test scripts.

Test automation metrics – script execution

- During a phase of execution of automated test scripts use the following metric:
- **The duration of execution of all test scripts since the moment new build become available until all results are analysed and anomalies/defects are reported.**
- An example:
  - Build became available at 2:00 AM.
  - Testers started execution of automated test scripts at 9:00 AM.
  - Test execution was completed at 12:00 AM.
  - Results were analyzed and anomalies/defects were reported by 2:00 PM.
  - In this example test execution took 12 hours from 2:00 AM when new build became available until 2:00 PM when all discovered anomalies were reported.

An ideal case of GUI test automation

- When functional testers arrive at 9:00 AM they should have anomalies discovered by automated scripts already reported to them!
- How we can reach this goal:
  - Start the execution of a set of automated test cases immediately after a build process was successfully finished, Application was deployed and restarted. This could be done by:
    - A build manager starting execution of automated test cases manually.
    - Integration of an automated build process and test execution.
  - Test automation team can start analysis of results of test execution one or two hours before the start of the work day of a functional test team.
  - Use a set of additional workstations and test tool licenses specifically for test execution.
  - Each test automation specialist should have at least two workstations and two GUI testing tool licenses.

GUI Test Automation - Summary

- GUI test automation tools and frameworks are evolving
- The following is the current industry standard for GUI test automation, which ensures more efficient test automation and a better ROI:
  - Mainstream GUI test automation tools (Mercury WinRunner and QuickTest Pro, Segue Silk, Compuware QARun, Rational Robot, etc.)
  - A test automation framework which uses a combination of “Functional decomposition” and “Data driving” approaches.
- Pros:
  - **High Maintainability**
  - **High Efficiency**

Appendices

Appendices

Further reading - 1

- You can find a lot of useful information on GUI test automation on the vendors' sites:
  - Mercury Interactive: http://www.mercury.com
  - Segue: http://www.segue.com/
  - Compuware: http://www.compuware.com/
  - IBM / Rational: http://www-306.ibm.com/software/rational/

Further reading - 2

- Keyword driven frameworks:
  - “Action words driven testing” by Hans Buwalda: www.sqe.com/archive/stacf2001/TestAutomation.pdf
  - “Keyword Driven Test Automation” by Carl J. Nagle: safsdev.sourceforge.net/DataDrivenTestAutomationFrameworks.htm
  - “Totally Data-Driven Automated Testing” by Keith Zambelich: www.sqa-test.com/w_paper1.html
  - “Certify” by Linda Hayes: http://www.worksoft.com/

References - 1

The following sources were used for this presentation:

- A diagram of a keyword driven framework from “Test Automation Frameworks” by Carl J. Nagle : http://safsdev.sourceforge.net/DataDrivenTestAutomationFrameworks.htm
- A screenshot of Certify from: http://www.worksoft.com/
- A sample Visual Test script from "Building an Automation Framework with Rational Visual Test“ by Thomas Arnold: http://www.automationjunkies.com/resources/article_build_auto_frame.shtml

References - 2

The following sources were used for this presentation:

- Sample Perl code from "Automated GUI Testing " by George Nistorica: http://www.perl.com/pub/a/2005/08/11/win32guitest.html?page=1
- A sample WinRunner keyword driven script by Larry Liu (CGI CTQ)
- A sample FITness script from: http://www.fitnesse.org/FitNesse.ActionFixture

Presenter's contact information

Yury Makedonov, CPS
Senior Technical Consultant
Centre for Testing and Quality
CGI services to BCE
125 Commerce Valley Drive West, Floor 6B,
Markham, ON, L3T7W4
(905) 762-2705 – office,
(647) 224-6619 – cell
ivm@ivm-s.com
http://www.softwaretestconsulting.com