Supporting Steps for Successful Test Automation Projects

Brian LeSuer

Overview

- Why do test automation projects fail?
  - Misconceptions
- Advanced planning is the key to success
  - Setting realistic goals
  - Evaluating existing staff skill sets
  - Evaluating testing tools
  - Evaluating the application to be tested

Overview

- Changing the way we test software
  - Test planning
  - Test case design
  - QA Department culture
- Decisions along the way
  - Choosing what to automate
  - Preparing test case data
  - Providing adequate equipment
  - Building a strong foundation
  - Implementing source code control
  - Developing standards

63% of Test Automation Projects Fail

- Misconceptions
  - Testing tools don't work
  - Testing tools are too difficult to use
  - My application is too complex
  - Our project schedules are too tight
  - Management will never support it
  - Automation will eliminate the need for all manual testing
  - Manual testing must have taken place before automating a feature

Setting realistic goals

- Rules of thumb are difficult to apply
- Conduct a small pilot project
  - Less than a month in duration
  - After creating “just enough” infrastructure, track time to automate each feature
  - Compare time to test manually
- Use this data for subsequent projects
  - Continue to update data
  - Automation will become more efficient after the first few projects

Evaluating Staff Skill Sets

- Need at least one member with experience
- Need at least one member with programming skills
- Methodology should be appropriate for staff skill sets
- Provide training beyond test tool vendor offerings

Evaluating Testing Tools

- Does the test tool support all of the required platforms?
- Is the learning curve of the testing tool and approach appropriate for staff skill set?
- Weigh the ease of script maintenance at least as high as ease of developing scripts
- Does testing tool provide a means for reusing automation components?

Evaluating Testing Tools

- Does the test tool provide a recovery mechanism?
- Is there a powerful scripting language?
- Does the test tool recognize application objects?

Evaluating The Application

- Are there aspects of the application that will be difficult to automate?
- Build testability into the application
  - Exposing application data
  - Modifying custom controls
  - Building custom controls – guidelines to follow
  - Choosing automation-friendly third-party controls

Building In Testability

- Proper naming of application pages and objects
- Addition of “hidden” controls
- Use of standard objects
- Build custom objects and choose third-party controls that are automation-friendly

Provide Unique Page Names

STAR QUALITY

Name Arrays of Similar Controls

Compare Items in the All Cameras Category

Let us help you find the item that best fits your needs.

Click the box next to the products you want to compare. Then click the Compare button to see the results.

Item Name	Item No.	Price	Select
Concord EyeQ Go2000 Digital Camera	578012	$79.99	☐
DXG 308 Digital Camera	566754	$89.99	☐
Fujifilm FinePix A205 Digital Camera	515573	$99.98	☐
DXG 321 Digital Camera	566755	$119.99	☐
Kodak Easyshare CX7300 Digital Camera	568420	$129.99	☐
HP Photosmart 435 Digital Camera	573965	$129.99	☐
Concord Eye-Q 3341 Digital Camera	582126	$129.99	☐
Olympus D-395 Digital Camera	564918	$149.98	☐
DXG 328 Digital Camera	578086	$149.99	☐
Fujifilm FinePix A330 Digital Camera	561920	$179.98	☐
Canon Powershot A400 Digital Camera	578635	$179.98	☐
Canon A60 digital camera	504728	$179.99	☐
Concord Eye-Q 4360z Digital Camera	582127	$179.99	☐
Toshiba PDR-4300 refurbished digital camera	569334	$189.99	☐

STAR QUALITY

Adding Hidden Html Objects

- Add a hidden html object for each control that returns the value of the control

```
<input type="hidden" name="qa_headerpage"
        value="qa_username=Tester,,qa_itemsincart=1
        item,,qa_cart_total=$13.79" />
```

Custom/Third Party Objects

- Publish useful methods (e.g. SelectCell)
- Publish properties (e.g. sClip)
- Add new properties for test verification
- Provide an hWin for each application control
- Copy-enable text fields
- Add hidden control (same color as background) to reflect value of rendered text

Custom/Third Party Objects

- Implement keyboard short-cuts and accelerators
- Follow platform standards for keyboard commands, e.g. <HOME>, <END>, <CTRL-HOME>, <SPACE>, etc.
- Use test applets from third-party vendors to verify compatibility before purchasing

Developing Test Plans

- Test plans structured by test objectives are optimized for automation
  - Results in more efficient automation
  - Promotes reuse of test components
- Test plan scenarios do not provide adequate structure
  - Results in automated tests that are long and complex
  - Results in automated tests that are too broad

Design Good Test Cases

- Test cases should have a single objective
- Test cases should result in one of two dispositions: pass or fail
- Test cases are independent
- No test case relies on the successful completion of another test
- Test cases start and stop at a known ‘base’ state

QA Culture

- Test automation will not be successful if it’s a ‘skunk-works’
- Automated tests need to be integrated with manual testing
- All QA staff members should have a role in test automation

Choosing What To Automate

- Good candidates
  - Short or simple transactions
  - Many data combinations
  - Expected results are stable or easy to generate at runtime
  - Tests that are executed regularly
  - Tasks that are difficult to do manually
  - Highest priority features
- Poor candidates
  - Long or complex transactions
  - One-offs
  - Unstable or difficult to predict results
  - Tests that cross multiple applications

Preparing Test Data

- Maintain control of test data
  - Establish data ownership
  - Strive to re-establish data state where possible
- Where data is dynamic, develop techniques to predict results at runtime
  - Search Engines
  - Retail Order Entry

Equipment Requirements

- Provide 2 machines on the desktop of each automated test engineer
  - One for developing new automation
  - One for testing/debugging recent automation
- Provide a lab for shared usage
  - Automation should run against every build
  - All automation should run within a reasonable time
  - Automation should run against all supported platforms

Building A Strong Framework

- Make test case maintenance a top priority
- Strive for early successes to gain management support
- Maximize reuse of automated test components
- Ensure that test cases run across supported platforms
- Ensure that test cases are not machine-dependent

Establishing Source Code Control

- Test code should be protected using the same standards as application code
- Any of the commercial packages will suffice
- Use the system being used for application code

Developing Standards

- Implement automation standards
  - Coding is faster and more efficient
  - Maintenance costs are lower
  - Learning curves are reduced
  - Staff can be easily redeployed

Developing Standards

- Capitalization
  - Data types are in upper case
    - STRING, INTEGER, BOOLEAN, etc.
  - User defined data types are all upper case
    - ORDER_ITEM, EMPLOYEE_DATA
  - Camel-back notation for variables
    - sLastName, rWage, bState
- Method Names
  - Standard names for common tasks
    - Invoke
    - Accept
    - Dismiss
    - Close

Developing Standards

- Coding Standards
  - All non-trivial code should be commented
  - Include counters to prevent infinite loops
  - Default clauses should be provided for all conditional statements like switch or select
  - Write clear and concise error message in all verification methods
  - File paths should never be hard-coded
  - Establish a data-dictionary

Questions?