Is Your Automation Agile or Fragile?

Building Flexible Automated Tests
For a Flexible Environment

BUSINESS CASE FOR TEST AUTOMATION
MAKING A PROJECT SPECIFIC CASE AND COMMUNICATING ROI

LAYING A FOUNDATION FOR TEST AUTOMATION
POSITIONING YOUR AUTOMATION TO LAST

X MARKS THE SPOT: BUILDING AN OBJECT MAP WITH XPath
WHICH ONE ARE YOU?

“Black Hat events are the premiere cyber security events you can attend... (with) the hottest, most relevant speakers and leaders in the field.”
—Jim Christy, Director of Futures Exploration, DOD Cyber Crime Center

THE BAD GUYS KNOW YOUR TRICKS, DO YOU KNOW THEIRS?

Are you the kind of security professional that craves the freshest information? Are you the kind of hands-on network defender that needs to be the first to know? If you stand out from the crowd because you’re seriously committed to security, then you’re our type — the Black Hat type.

The Black Hat Briefings are the most important technical security events on the planet, and we create them for people just like you — people who know security and want to be part of the solution. Join us in Las Vegas for 10 full tracks and over 60 intense hands-on training sessions of timely, actionable security information.

Black Hat USA 2009
July 25-30 Las Vegas, NV
CONTENTS

COVER STORY

IS YOUR AUTOMATION AGILE OR FRAGILE? 22
In order to create an effective test automation regime, you must be able to bend and adjust to the environment with which you are presented. Lack of flexibility will ultimately lead to broken test automation, therefore, it’s important to learn how to be flexible.  By Dion Johnson

FEATURES

THE BUSINESS CASE FOR AUTOMATED SOFTWARE TESTING 12
Articulate the expected Return on Investment (ROI), and employ strategies and techniques for developing the automated software testing business case for your project.  By Elfriede Dustin

LAYING A FOUNDATION FOR AUTOMATION 28
Overcoming management resistance to test automation, and laying the groundwork for lasting success.  By Randall W. Rice

COLUMNS & DEPARTMENTS

EDITORIAL 4
Scripting Calisthenics
Keep your automation skills in shape with Calisthenics.

AUTHORS AND EVENTS 6
Learn about AST authors and upcoming events.

SWEAT THE TECHNIQUE 8
XPath Marks the Spot
Learn how to create an Object Map feature with XPath.

OPEN SOURCERY 11
Tool Enthusiasts Unite!
Small groups are meeting all over to country to discuss test automation and the open source tools they love.

UP-TO-DATE WITH ATI 32
Find out what’s happening on the Automated Testing Institute’s Online Reference.

HOT TOPICS IN AUTOMATION 34
IAST Is Hot!
Read this review of Elfriede Dustin’s newest book entitled Implementing Automated Software Testing.

Automated Software Testing (AST) Magazine is an Automated Testing Institute (ATI) publication that serves as a companion to the ATI Online Reference (www.automatedtestinginstitute.com).

For more information regarding the magazine visit:
http://www.astmagazine.automatedtestinginstitute.com
Test automation is like a resistance training workout program; the more you do, the better and stronger you get. The dilemma with test automation projects, however, is that they are very often fleeting. When a test cycle ends, test automation often doesn’t resume until much later after a stable version of the next release comes to the test team. Other times, a project or contract ends, and the next contract does not utilize your test automation skills at all. Either of these situations could have an automator away from the test automation keyboard for an extended, even indeterminate period of time, which presents a serious challenge. Anyone that is used to weight training knows that if you stop for an extended period of time, your muscles begin to atrophy or waste away. Then, if you later attempt to resume working out with the same weight or intensity that you previously enjoyed, you risk injuring yourself. The same applies to test automation. If you stop using your test automation skills for an extended period of time, they begin to atrophy. You forget basic scripting syntax and techniques, and if you try to restart a vigorous test automation effort, you risk getting injured. Injury in the automation arena is seen in the form of failed test automation implementation, loss of credibility, loss of confidence, increased stress due to overwhelming expectations, or even job loss.

When recessing from an extremely intense automation effort, the best thing to do is go into maintenance mode, to maintain the knowledge and skill that you’ve attained.

To support you in your maintenance, this article defines a program of Automation Calisthenics. Calisthenics in physical training are exercises consisting of a variety of simple movements, usually performed without access to the heavy duty equipment that we previously used. This heavy duty equipment may include the Application Under Test (AUT) that we were automating, the commercial tools we used, and/or framework in which test automation was performed. Automation Calisthenics don’t require any of this, but instead can use some light weight equipment including scripting languages, open source tools, and freeware applications.

**Scripting Calisthenic Exercises**

Automation Calisthenics involve the development and implementation of various scripting components. **Class development and implementation** is a powerful exercise for maintaining scripting fitness, because it requires a certain level of comfort with objects, properties and methods. Remaining comfortable with these concepts is useful, because they are at the core of several other concepts that are important to successful test automation implementation.
concepts such as object maps, AUT object models, runtime objects used by automated test tools, and the properties and methods native to each of these objects.

Control flow function creation is an excellent exercise, because much of the scripting that takes place during test automation involves the use of control flow functions. After you learn the objects associated with the AUT, and the statements in the automated testing tool that allow you to manipulate those objects, the main thing that is left for you to do is control the execution of those statements with control flow functions. Therefore, simple development of scripts that utilize if-then-else or case branching constructs, and for-next or do-while looping constructs will keep you sharp in this area.

Developing and implementing classes will provide you with a chance to create functions in the form of class methods, but this is not the same type of exercise as the creation and implementation of a function library. This typically requires that you organize the library of functions in some sort of logical grouping, load the library in the script in which it is to be used, and then use the functions in the proper way. These are all important actions to many real life automation efforts, which is why developing function libraries is an excellent exercise.

The creation of Assertions is excellent for keeping automation verification skills from cooling. Assertions are basic verification points that allow your scripts to identify failures in expected outcomes. The most basic assertions may be created using branching constructs that signal a positive or negative result, depending on the condition being evaluated. To get the maximum burn from this exercise, however, it is best to implement the assertion from an external function library. Don’t just add a branching construct within a script. Create the assertion within a function with arguments, and return values. This will allow you to exercise the automation skills associated with AUT verification, and error handling.

File Manipulation is an exercise that may be deemed the “squats” of automation. Anyone that regularly does physical exercise knows about squats. A squat is a lower body exercise that is touted as one of the best resistance exercises for toning and building muscle all over the body. File manipulation in automated test scripts can have the same affect on your automation skills. File manipulation normally involves the instantiation and use of operating system objects, which allows you to exercise skills dealing with objects, properties and methods. In addition, since these objects allow you to check and perform actions on various operating system components, they behave much like an AUT. Also, you can use these objects to create and write information to simple text files, which allows you to exercise your automation reporting skills.

Conclusion
Maintaining test automation fitness is paramount to a test automator, and with Test Automation Calisthenics, downtime doesn’t have to result in the loss of skills. On the contrary, it could actually give you a chance to tone your skills so that you will be even healthier and more prepared for the next big automation effort that comes your way.

Many of the exercises that may be conducted as part of Automation Calisthenics are detailed in the Techniques section of the Automated Testing Institute’s Online Reference. For access to this Techniques section today visit:

http://www.techniques.automatedtestinginstitute.com
Elfriede Dustin provides the blueprint for making a business case for test automation. Elfriede has authored various white papers on the topic of automated software testing, teaches testing tutorials and is a frequent speaker at various software testing conferences. With a Bachelor of Science degree in Computer Science and over 15 years of IT experience, Elfriede has been responsible for implementing automated testing for several projects. She is the coauthor of several books such as Effective Software Testing and Automated Software Testing, and among her many accolades now graces the Automated Software Testing Magazine with an excerpt from her latest book entitled Implementing Automated Software Testing.

In the lead feature, Dion Johnson provides guidance for keeping your test automation agile in an Agile development environment. Dion has several years of experience in providing IT services to both government and private industry in a manner that has demonstrated expertise in multiple areas of the software development lifecycle. With a Bachelor of Science degree in Electrical Engineering, he has spent much of his professional career as a consultant in the areas of quality assurance (QA), quality control (QC), software process improvements, requirements analysis and software test automation. As a regular conference speaker and presenter, Dion has delivered award winning and highly acclaimed presentations at many of the most prestigious industry conferences, including the StarEast and StarWest International Conference on Software Testing, Analysis and Review, the Quality Assurance Institute Conference and Better Software Conference. He also teaches college and business level classes relative to testing and test automation, and has several published articles in various IT publications.

Randall W. Rice is a leading author, speaker and consultant in the field of software testing and software quality. He has over 30 years experience building and testing mission-critical projects in a variety of environments and has authored over 50 training courses in software testing and software engineering. He is a popular speaker at international conferences on software testing and is also publisher of The Software Quality Advisor newsletter. He is coauthor with William E. Perry of the book, Surviving the Top Ten Challenges of Software Testing and Testing Dirty Systems, published by Dorset House Publishing Co. Randy also serves on the board of directors of the American Software Testing Qualifications Board (ASTQB). You can contact Randy at rrice@riceconsulting.com, or from his website at www.riceconsulting.com.
ATI 2009

AUTOMATION HONORS

Celebrating Excellence in Test Automation

www.atihonors.automatedtestinginstitute.com

Nominations: April 1 - June 30
Voting: August 1 - October 31

Nominate the Best Today!
As you probably already know, an Object Map is a file used in test automation to maintain a logical representation of each Application Under Test (AUT) object. This file serves the purpose of separating application object property data, known as the Physical Description, from the automated script. Without an Object Map, referencing an object in a script would be done using the physical description of that object, similar to the following statement:

```
GetElement("Class:=Button", "Text:=O Button").Click
```

The `Class` and the `Text` properties make up the physical description of the object represented in the above statement. With an object map, the same object would be identified using a Logical Name as in the following statement:

```
GetElement("OvalButton").Click
```

This separation helps increase an automated test framework’s robustness, because it is likely that an object is referenced in several locations among the automated test scripts within that framework. If the properties of the object change in the application, an Object Map ensures that this change only needs to be reflected in a single location as opposed to every place in the automated test scripts in which the object is referenced.

Not all tools come with an Object Map feature, however, so reaping these benefits often requires that you build your own. An excellent way to accomplish this is by using XML and XPath. XML – extensible markup language – is an open standard for structuring information. XPath is a language used to navigate through elements and attributes in an XML document in order to find specific information. In essence, XPath queries for information in an XML document much like SQL queries for data in a database. Why use XML? XML is written in a text file, which simplifies data storage, and simplifies sharing across and porting to various platforms. In addition, information storage and retrieval become standardized. Following are steps for building an Object Map:

**Step 1. Identify the basic types of objects the repository will support**
- Identify the basic types of objects the repository will support
- Create the XML Object Map structure
- Create the utility script(s) that use(s) XPath to read the repository
- Update automated library functions to reference the XPath utility script(s)

You can build from this list of objects as
necessary.

**Step 2. Create XML Object Map structure**

You have a lot of freedom regarding how you choose to create the structure, since you’ll also be responsible for writing the script that will interpret that structure. The XML Object Map discussed in this article will comply with the following set of rules:

- XML must contain one element – the root element – that is the parent of all other elements. This element may be called `<Object_Map>`.
- Each object in the Object Map is defined within an `<Object>` element. This element is a child of the `<Object_Map>` element, and it has an attribute known as ‘logicalName’ which describes the name that will be used to reference the object in a script. This name should be unique, to prevent conflicts with other objects.
- Each Object will have three child elements that will collectively define a single object property:
  - `<PropertyClass>` – Defines the type of object being used (i.e. textbox, listbox, etc.)
  - `<PropertyName>` – Defines the name of the property
  - `<PropertyValue>` – Defines the value of the property

Note: This particular set of rules is most effective when one property is defined per object. There are many approaches to defining an Object Map, but for simplicity this is the definition that we will use in this article. With these rules, the Object Map may appear shown in Figure 1, which defines three objects:
- A `UserName` textbox object that is defined in the application by an `id` property that is equal to “userID”
- A `Password` textbox object that is defined in the application by an `id` property that is equal to “pass”
- A `Login` button object that is defined in the application by a `value` property that is equal to “loginButton”

**Step 3. Create the utility script(s) that use(s) XPath to read the repository**

This is where a little scripting knowledge and knowledge of XPath come in handy. While this article will not be a full-blown XPath tutorial, we will discuss some of the basics.

```ruby
def parseXML(objName, objMap)
  xmlDoc = WIN32OLE.new("Microsoft.XMLDOM")
  xmlDoc.async="false"
  xmlDoc.load(objMap)

  #""Write Batch File Lines for 1st Environment

  propertyNameNode = xmlDoc.selectNodes("/Object[@Name='" + objName + "]/propertyName")
  propertyValueNode = xmlDoc.selectNodes("/Object[@Name='" + objName + "]/propertyValue")
  propertyClassNode = xmlDoc.selectNodes("/Object[@Name='" + objName + "]/propertyClass")

  propName = propertyNameNode.item(0).text
  propValue = propertyValueNode.item(0).text
  propClass = propertyClassNode.item(0).text

  propData = [propName, propValue, propClass]
  return(propData)
end
```

Figure 2: XPath Utility Script (in Ruby)

Essentially, we want to be able to query for the object by its Logical Name, and then have the property class, name and value of that object return. To accomplish this, we need to understand and use basic XPath syntax as it applies to our specific Object Map. Following basic XPath statement that will be used:

```
/Object[@logicalName='Login']/propertyClass
```

The `/Object` portion of the statement will obtain all Object elements in the XML document. Adding `[@logicalName='Login']` will obtain all elements in the document that have a `logicalName` attribute that is equal to ‘Login’. Finally, the `propertyClass` portion will obtain all `propertyClass` elements that are children of elements that have the ‘Login’ `logicalName` attribute. Ultimately, this statement would return “button” from our file. To return the `propertyName` and `propertyValue` similar statements are constructed.

The way you use these statements in your code, will largely depend on your platform and scripting language. If using Ruby in a Windows environment, you may have a function similar to the one shown in Figure 2. The method shown in the figure, called parseXML has two arguments:

- `objName` – The logical name of the object in the Object Map
- `objMap` – The full file path of the Object Map XML file

Lines 3 through 4 load the XML file so that it may be queried, while lines 9 through 11 obtain the `propertyName`, `propertyValue` and `propertyClass` nodes of the desired object (based on the logical name assigned to the `objName` argument). Lines 14 through 16 obtain the actual text associated with the `propertyName`, `propertyValue` and `propertyClass` nodes, while lines 18 through 19 return this text in the form of an array.

Therefore, if we assigned “UserName” to the `objName` argument, and set the `objMap` argument equal to the path of our Object Map, this function would return the following array:

(Continued on page 31)
How Do You Network?
Whatever your choice, the Automated Testing Institute is there!

Stay up-to-date with test automation by following the Automated Testing Institute on your social networking site of choice.

For more information, visit http://www.networking.automatedtestinginstitute.com
If you’ve been tuning into the Automated Testing Institute’s video news series entitled “This Week in Automation”, you’ll recall that we’ve been reporting on various open source meetups that have been taking place all over the country, beginning with an event known as OSTATLI, conducted by Ms. Elisabeth Hendrickson on February 5th, in Pleasanton, CA. OSTATLI, which stands for Open Source Test Automation Tools Love-In, was created for the purpose of allowing participants to express their “mutual love of open source test automation tools”.

According to Hendrickson’s blog, the meeting was sort of free form, relying on sticky notes, and small groups to set and carry out a spontaneous agenda. Later in the day the group gathered for a demo conducted by meeting participant Ken Pier, Product and Quality Assurance manager at Socialtext, highlighting the test harness that his company has built around the Selenium open source automated test tool. After that demo a second demo was conducted, then the meeting was moved to a local pub in which the participants could continue with an informal conversation about “geek stuff with a beer in our hands”. From the accounts of the meeting from participants - including Hendrickson’s account on http://www.testobsessed.com, and Dave Liebreich’s account on his blog site, http://www.daveliebreich.com/blog - the meeting seems to have been well received.

That gathering also seems to have been successful at touching a nerve and inspiring some other folks to conduct similar events. Al Snow, self described tool developer and automator, announced on his twitter site, http://twitter.com/jasnow, that he’d organize a similar event on March 7th called the Atlanta Open Source “Test Automation Shareathon, aka AOSTAS. But while the “Love-In” took place in California, the “Shareathon” took place in Atlanta, Georgia.

Now, Frank Cohen, a “leading authority for testing and optimizing software developed with Web, SOA, AJAX and REST designs and implementations” will lead a slew of open source meet-ups starting in late April in Texas and concluding in early June in Las Vegas, Nevada. For more information on these meetings, visit http://www.pushotest.com/services/workshop.

The open source meeting craze seems contagious! Get your shots, or you might be the next one to conduct a meeting.

To see the “This Week In Automation” video series and other automation news, visit: http://www.news.automatedtestinginstitute.com
THE BUSINESS CASE FOR AUTOMATED SOFTWARE TESTING

By Elfriede Dustin
As discussed throughout our book *Implementing Automated Software Testing* (IAST), Addison Wesley, March, 2009, there are many benefits to implementing Automated Software Testing (AST). Chapter 2 of IAST discusses in detail the why’s of automated testing, i.e. “Why Automate?” If applied correctly, those automated testing benefits can include reducing the time and cost of software testing, improving software quality, and many other factors that will positively impact your software test program. Understanding these potential benefits and being able to explain them are important when introducing the idea of AST to your management and test teams. Additionally, it is important to have buy-in from upper management and the various stakeholders affected in order to secure the funding required for AST. It is important to be able to articulate for your particular project the specific business needs that will be met, the expected Return on Investment (ROI), and any other considerations beyond ROI. This article is an excerpt from the IAST book and is focused on strategies and techniques for developing the AST business case for your project, so that you can gain the necessary buy-in and support required.
DEFINITION OF THE BUSINESS CASE

Various definitions of business case exist, much has been written about how to write a business case, and various related templates are available. A quick search on Google for the term brings up thousands of references. The purpose of this discussion is not to redefine what a business case is, but to use existing definitions to develop a business case for AST. Our definition is a combination of the existing ones. To summarize, the purposes of a business case are to

- Identify the business needs—provide the reasoning for initiating a project or task, in this case AST
- Justify the business needs in terms of costs and benefits—calculate the ROI
- Include the expected risks

An example of a summarized business need for AST could be that an AST program will support the timeline decrease of the software testing lifecycle, facilitate defect identification, and thus help improve quality. Naturally, improved quality through minimal defects leads to improved customer satisfaction.

In our experience, large-scale AST projects are much more successful when the team has thought through the specific areas and reasons for implementing AST, is able to describe the expected benefits and costs, and has also taken into account what is needed to be successful in terms of time, skills, and processes. Additionally, it is important that upper management and other stakeholders buy in to the business case associated with the AST effort.

On the other hand, too often we see projects that start out with the purchase of an automation tool without any proper automation skill acquisitions or real analysis of what the tool can reasonably be expected to accomplish, other than speeding up testing. No thought has been given to how this “speeding up” is to be accomplished. The automation tool is provided to a team that has little or no experience or training with the tool, with an expectation that some automated tests will soon be running. The team then identifies the tests that look like the easiest to automate so they can run some with the new automation tool, and through hard work and perseverance they get some tests automated. Then the questions come: “Now that we have these tests automated, how much will this save us in the future? How much better testing did we do?” The answers are disappointing because the tests that were automated were selected because they were easy, not because they have a big impact on the test program, or they result in minimal savings because they are not even run very often because they cover low risk areas. The consensus becomes “We tried AST once but it didn’t really make much of a difference for our project.” The team blocks existing or future automation because of a bad experience based upon the wrong ingredients and false starts.

We have found that spending the time up front to identify how AST can support a project’s goals and what the expected benefits are to be an important step that is often left out. The process of developing the business case is a discipline we highly encourage when introducing AST to your project.

IDENTIFYING THE BUSINESS NEEDS

As we identify the business needs and justify them in terms of cost and benefits, we should ask ourselves a few questions, to narrow down our specific needs. For example:

- Do we want to reduce the lifecycle cost of a software product?
- Do we want to deliver software products with fewer defects, and have extremely high confidence that certain functionality or capability is working as expected; for example are some tests nearly impossible to run manually, requiring AST?
- Do we want to remove the mundane tasks for members of the testing team, whose skills are best applied on more complex areas of the AUT than rerunning manual tests over and over?

Most often the answer to these questions is “All of the above!” We want to deliver faster, at a lower cost, with fewer defects and higher confidence and at the same time increase test team efficiency. In this scenario, then, the consideration of how to best apply AST to a project should take into account all areas that contribute to each of these factors, such as

- The need for speeding up the testing effort and increase the testing efficiency.
- The need for decreasing the testing cost
- The need for applying test team member skills most effectively
Need for Speeding Up the Testing Effort and Efficiency

How much would speeding up the testing really impact the product delivery timeline? If a particular test activity is not really in the critical path, automating it will probably not impact delivery time much. On the other hand, if a test or set of test program activities can be identified that are truly on the critical path to improving a product delivery timeline, automating it, can improve software delivery time. For example, a test setup activity, i.e. restoring a test baseline and setting up the required pre-conditions for a test are very time-consuming, but automating these activities would reduce this activity by 60 percent. Using these types of “potential time savings” metrics allows for an assessment of the potential impact on schedule of reducing the execution time and can be used for developing the business case. Generally we can speed up the testing effort and decrease cost if we increase the testing efficiency.

When assessing the testing efficiency, various questions should be asked: What category of software defects is most important to minimize? Which set of test activities, tests and test techniques have proven to be the most important in discovering these types of software defects? Which are the tests that are critical to be able to consistently repeat and also likely to be the ones you will want to run most often, because they cover a complex, high risk or problematic area and have the tendency to uncover most defects? Also consider the tests that manual testing can’t hardly achieve, because they are so complex, such as memory leak detection, concurrency, performance, and so forth and automation is needed. Consider what would the impact on software quality be if they were run more often as overnight regression tests? Do we have the right skills on board to allow for testing efficiency? Are we finding defects early enough or too late in the testing cycles? Do we have adequate schedules? The most efficient testing team can’t be successful if there isn’t enough time to implement and execute the test program’s best laid plans.

Test program activities should be tailored to your business needs and your specific project. Then, once you complete an inventory of the test activities and assess them against your business needs, you can provide an objective recommendation of the benefits of automation areas, which phases of the Software Testing Lifecycle (STL) to automate, which specific tests to automate, why you selected those areas and tests, and analyzed carefully and AST is applied to the appropriate areas.

The need for applying test team member skills most effectively

Chapter 10 in the IAST book talks about putting the right people on the project and describes the skills required of members of an effective test program. Manual testing can be tedious, cumbersome, error prone and very mundane. It can be exciting, when a new software release of the Application Under Test (AUT) is received the first time and testers start testing it to find defects. This might still be interesting to do when receiving the second and third release, but running the same mundane manual tests over and over on the fourth and n° AUT release can become boring very fast. During the fifth or later releases, often testers decide they tested an area previously and thought that certainly it will still work the same and they don’t repeat the test again on the n° release. This decision to not rerun a test often can be detrimental to the quality of a product and that’s how often times defects slip into production. AST takes away this guesswork as to whether to rerun a test and is ideal for this type of mundane, cumbersome and tedious testing. Test team member skills are best applied in the more complex areas, while the automated tests play back the mundane tests behind the scenes. AST can help improve tester morale, because not only can they learn new AST skills, but also, they can focus on new or more interesting activities, rather than rerunning a manual test for the n° time.

For calculating the business benefit, you want to determine how much testing currently costs.

Need for Decreasing the Testing Cost

If testing is sped up and efficiencies are increased, one of the resulting benefits can be a decrease in testing cost. For calculating the business benefit, you want to determine how much testing currently costs. If you break down the cost of testing into the various phases such as unit testing, component testing, system testing, and other testing phases discussed in Appendix B in the IAST book, which phase costs the most? Which tests add most value, i.e. cover the highest risk areas and absolutely have to be run? In other words, are the tests that cost the most to execute adding the most value? Of these tests, which ones are repeated the most often? How much time and effort are required to run these tests? What would be the impact on confidence in the software delivery if these tests could be run more often because they are now automated and the test coverage from these tests could be expanded? Testing cost can be decreased if the test activities are
Weekly Automation Tube Featured Videos

Check out this week's featured Automated Testing Videos from the web. For more information about ATI click on the header links, or click here.

Weekly Featured Articles

An article is featured in each of the following categories:
- Lifecycle/Methodology
- Frameworks
- ROI
- Techniques/Toolsmithing
- Performance/Load
- Security and Unit

Open Source Test Automation

**Summary:** This paper introduces the reader to Open-Source Test Automation (OSTA) and outlines the procedure for implementing.

ATI's Latest And Greatest

- Espionage and Software: U.S. Vulnerable to Cyberspies?
- Microsoft Powerpoint Defect Posing Security Risk
- ‘Software Problem’ Results in Shocking Letter To Parents of Dead Teen
- Benerator Releases 0.5.8
- Software Testing: The Fastest Growing IT Service
- New Year, New Level: What’s Next in Automation
- Catching a Code in the Summer
- HP Release QTP 10.0
The Automated Testing Institute (ATI) Online Reference is the web’s foremost comprehensive online resource for obtaining, exchanging and certifying industry standard test automation knowledge, information, and techniques.

Visit us today!
implementing AST. This section describes how to calculate the potential cost savings or ROI when implementing AST in comparison to performing the same tests manually. ROI accounts for not only the time savings from automation but also the investment it will take to implement AST to give you a complete picture.

AST ROI can also be non-tangential, i.e. automation pays off for areas of the test program that are difficult to be described in metrics and numbers, but are simply more beneficial to automate. Some of these areas are described in Chapter 3 of the IAST book. For the tangential areas, we have developed an ROI “calculator” that allows for entering all the factors needed to determine AST time and cost savings. As presented in the IAST book, the ROI calculator is broken down into a set of worksheets focusing on each of the major activities associated with testing:

- test setup,
- test development,
- test execution, and
- test evaluation.

Each of these categories are described in detail in the ROI section of the IAST book.

Other ROI Considerations

Personnel Requirements

When calculating the AST cost savings, keep in mind that different skills are required to implement an effective automated testing program from those required for manual testing. The cost of possibly training existing personnel or the cost of hiring qualified AST engineers (aka software developers) needs to be considered and included in the ROI considerations. Chapter 10 in IAST discusses the skill set required for implementing an effective automated testing program. During this ROI consideration also compare the number of people required for manual versus automated testing efforts. Although you might have to hire software developers to implement the AST tasks, it might be possible to retrain the testers who run the various labor-intensive manual regression tests over numerous days to do some of the AST activities. Additionally, those test personnel can focus on additional testing tasks or projects which is an additional ROI, but often too difficult to put into an actual metric that shows the gains.

Lab Equipment Requirements

Also, as part of AST ROI considerations, you should consider additional equipment required, considering the most efficient way to support AST, such as evaluating the feasibility of implementing the test lab using VMWare or other OS image/virtualization software.

Test Case Maintenance

So far we’ve discussed original test case setup, but test case maintenance should also be included as part of the ROI considerations. When evaluating AST ROI, it is also important to compare test case maintenance for the manual versus automated testing efforts.

Intangible Automation Savings

Some intangible or difficult-to-measure automated ROI can be achieved. For instance, it is difficult to measure the impact on an automated testing effort when test coverage is increased via automated test data generation and increased test scenario coverage that only automation can achieve. It is difficult to measure the impact of increased test coverage if using different data but, for example, using the same data paths, resulting in different scenarios. Here we need to evaluate whether running additional test data scenarios really will benefit the testing effort or if we are just duplicating the same test path without adding any value—ignoring the rules of equivalence partitioning, for example. Note: Wikipedia has a good definition of Equivalence Partitioning: Equivalence partitioning is a software testing technique in which test cases are designed to execute representatives from each equivalence partition, i.e. a partition of input values undergoing similar treatment. In principle, test cases are designed to cover each partition at least once. This has the goal:

1. To reduce the number of test cases to a necessary minimum.
2. To select the right test cases to cover all possible scenarios.

How many data combinations to run through the same data paths is a strategic decision, depending on complexity and implementation criteria. An additional difficult non-tangential benefit is to measure automated test success that manual testing couldn’t achieve, i.e., tests that were previously not run because they would have been too cost-prohibitive or almost impossible to execute, such as memory leak detection, stress testing, performance testing, concurrency testing, and so forth. ROI (Continued from page 15)
needs to be added for a high priority defect found using AST that a manual test could not have uncovered because one didn’t exist. Additionally, some of our customers tell us that currently they don’t run a regression test as often as they would like to, because it’s labor-intensive and a lot of setup is involved. However, once the regression test has been automated, they want to run the test more often. How do we measure “peace of mind”?

RISKS
Automated test program assumptions, prerequisites, and risks need to be understood and presented as part of the business case. These include any events, actions, or circumstances that will prevent the automated test program from being implemented or executed successfully, such as lack of skills, late budget approvals, the delayed arrival of test equipment, fluid and constantly changing requirements and software, or late availability of the software application.

The risk description needs to incorporate ways that help minimize it and prevent failure. Failure is defined in terms of cost overruns, schedule slippage, undetected critical software errors, rework required, and so on, despite the automated testing effort.

Most risk is caused by a few factors.

• **Short time-to-market** - Risks include a short time-to-market schedule for the software product. As mentioned previously, testing budgets and schedules are often determined at the onset of a project, during proposal efforts, without any “real” or “actual” estimates or inputs from testing personnel, based on past experiences or other effective estimation techniques. However, a test manager can quickly see when the short time-to-market schedule does not allow for adequate testing. As part of the business case, this high-risk issue needs to be pointed out. Either schedules should be adjusted, or risk mitigation strategies should be developed.

• **Lack of adequate skills** - Other risks may exist because new skills are required to implement the automated testing effort and currently adequate skills are not available in-house. The limited availability of engineering resources on the project would be considered another potential risk to AST success.

• **New technology** - If new technology is implemented, the risk can be high that no automated testing tool exists and the tool needs to be developed in-house.

• **Difficult to automate requirements or features** - Some system features exist that are not documented, are complex, and most often are the biggest problem area. Learning curves of the functionality, subject matter expert support, and documentation requirements need to be considered, before claiming a requirement or feature can successfully be tested in an automated fashion. Functional or nonfunctional requirements that are difficult to be tested because of complexity and lack of documentation will also pose a high risk to an AST effort.

As risks are identified, they need to be assessed for impact and then mitigated with a strategy for overcoming them should they be materialized as part of the business case. Carefully examine potential risks in order to try to prevent them from being realized; hiring people with the right skills for the job, reassessing the deadline or listing all the requirements needed in support of a successful AST effort, are ways to mitigate those listed risks. However, if risks are realized even though all precautions have been taken, a mitigation strategy needs to be in place.

OTHER CONSIDERATIONS
In earlier sections we described the benefits
that AST can offer, ideas for how to evaluate business needs and assess AST against those needs, and estimating ROI. This section lists other ways to approach the topic and other questions to consider in relation to AST.

AST may be the solution if you don’t have an answer to some of the following questions:

1. How will your test team keep up with the ever increasing complexity of new technologies implemented?
2. How will your test team’s productivity keep up with the rate of productivity improvement expected from the development team?
3. What is your testing strategy to support the increasing level of software and component reuse that is planned?
4. What artifacts can you provide to demonstrate the software was tested to meet specified standards? How quickly can those be produced and reproduced for each test run?
5. How do you capture and reuse the knowledge of the subject matter experts your test program currently depends on?
6. What is your plan for being able to document and re-create problems found during testing for a development team that is geographically distributed?
7. When the software product can be installed remotely, what is your approach to providing verification or diagnostic tests that can be executed remotely?
8. How will you run tests that manual testing can hardly accomplish, such as memory leak detection, stress testing, performance testing, and concurrency testing?
9. Will you need to justify having testers work night shifts, weekends, and overtime when an automated tool can just be kicked off at day’s end and results viewed the next morning?

Other such questions can be added, and in considering these questions and the future of your test program, AST should be evaluated as part of the potential solution.

So far we’ve discussed the importance of a business case and addressed how to identify the business needs, how to justify AST in terms of cost and benefits, and how to identify risks. The strategic approach to implementing AST and measuring the key success indicators also needs to be defined as part of the business case. Strategy is discussed in Chapter 6 of “IAST,” and the keys to success and metrics are discussed throughout this book.

**SUMMARY**

In order to get buy-in from upper management and all stakeholders involved, the business case for AST needs to be made. Our experience has shown that if buy-in exists (i.e., all stakeholders have bought into the automated testing effort), all parties feel responsible for the success of that effort. If all contribute, chances for success are much higher. It is therefore an important goal to develop the business case and get approval.

The market demands that software products continue to be delivered faster and cheaper with increasing reliability. Testing is a big part of the time, cost, and quality equation, and without changes in how testing is done, software projects will not be able keep up. In fact, there is much evidence that this is already the case. AST can play a big part in addressing these needs. More specifically, AST provides an opportunity to reduce the time and cost of software testing, improve software quality, help identify defects earlier in the development lifecycle, and improve your software test program in measurable and significant ways.

In order to implement AST successfully, an approved business case can set the right tone. We are now off to a good start.
Submit Your Stories!
Now’s Your Chance to be Heard

Have something to say?
ATI is here to listen

The Automated Testing Institute believes that one of the best ways to improve upon test automation is to listen to the lessons learned by others. So, if you have an interesting story that you’d like to share with the world, contact us at stories@automatedtestinginstitute.com to find out how.
Is Your Automation Agile or Fragile?

Building Flexible Automated Tests
For a Flexible Environment

By Dion Johnson
Recently, I came across a question posted on an online message board that asked whether or not test automation was possible in an Agile environment. I suggested that success would be tied to the ability to effectively customize the automation approach to the environment in which it was to be implemented, and therefore may result in a framework that is scaled down in structure. This didn’t really sit well, however, with those that believe a heavily structured and extremely robust framework is the answer to every automation question. It’s not hard to understand why this belief is held, though; it’s a direct result of the fact that test automation is too often treated as an afterthought. It is relegated to a spare time event that calls for no planning or skills beyond the ability to record manual test execution; and this treatment has led to failure in test automation implementation time and time again. Therefore, the response by many automation professionals may be summed up by the law of physics which suggests that “to every action there is an equal and opposite reaction”. I bet you never thought you’d have to dust off that physics book to understand test automation, did you? This law, known as Newton’s third law of motion is useful, however, in explaining our current dilemma. The frivolous nature with which many consistently treat test automation is an extreme action that is naturally met with the opposite and equally extreme reaction of trying to apply an intensive test automation framework to every test automation situation. It’s like a tug of war in which no one wins, because what many automation loyalist don’t realize is that both of these actions are prone to failure.

Don’t get me wrong, I am a strong advocate for structured test automation frameworks, and push for them whenever possible, but I am also a realist. And reality lets me know that there is no one-size-fits-all solution to successful test automation implementation. There are basic principles that can be applied in every situation, but you need to be flexible in your approach, both in an Agile environment and outside of an Agile environment, because project constraints always require you to bend and adjust what you plan on doing. Lack of flexibility in your automation approach will ultimately lead to that approach breaking.

I’ve worked in a couple of Agile projects that have used the Scrum methodology and even took the “Scrum Master” training. Why they call it “Scrum Master”, I don’t know, because I definitely didn’t feel like a master when it was over. I understood it well enough to do my job, however.
Scrum involves the compilation of a prioritized list of features planned for a product Increment. This compiled list, known as the Product Backlog, is evaluated at the start of each Sprint—a timeboxed period of software development that is often 15 or 30 days—and the highest priority items that are deemed attainable are transferred to what is known as the Sprint Backlog. Project members form a Scrum Team, often consisting of 5 to 9 people, and the team breaks down the prioritized functionality into detailed tasks. A short meeting is conducted at the beginning of each day to obtain progress and set priorities for that day. If work is not completed at the end of the Sprint, it is placed back on the Product Backlog until the next Sprint comes along. The point of all this is to produce constant, measureable results, and to facilitate change.

The purpose of this article is not to assert or deny the merits of Agile or Scrum. That’s for others to do. Instead, this article attempts to address how automation may survive in a rapidly changing environment such as one that uses Agile.

The “Sprinting and Scrumming” summary provided above tells us something very important about Scrum and Agile environments, and that is that everything is extremely dynamic. Priorities are set and changed often, teams are small and its members are constantly shifting, activities occur within a small window of time (timeboxed), and progress is constantly checked and measured. This presents the following challenges for highly structured test automation frameworks:

- **Difficulty Obtaining Buy-in** – Obtaining buy-in for an effective test automation program is challenging under any circumstance, but it can be extremely difficult in Agile environments. This is because many people still don’t see the value of actually breaking up automated framework development and implementation into tasks that can be tracked and burned down on a daily basis. And if it can’t be burned down on a daily basis, it’s difficult to justify having time in a Scrum environment devoted to getting it done.

- **Difficulty Securing Time** – Even if you are able to get buy-in for a highly structured test automation framework, it is difficult to secure the time that it entails, particularly frameworks for system test automation. Test automation frameworks require an up-front investment in time as well as money. The investment is normally paid back in a positive return on investment, but the investment is necessary, none-the-less. The investment is spent in analysis, design, development, documentation and implementation, and there is a good chance that you will not have much time for each of these items. In addition, you may also be assigned other project tasks that need to be completed in concert with the automation tasks, resulting in the automation tasks getting pushed by the wayside.

- **Difficulty Sustaining Participation** – It is very difficult getting agreement for a dedicated automation team, yet the project team is expected to have multiple people involved in automated test creation and implementation. Each person that will need to implement test automation has a million and one other daily tasks to consider, so it’s highly probable that they are not going to be interested in learning the framework that you’ve developed, obtained or customized.
While these challenges are not specific to Agile, they do seem to get amplified in an Agile environment. Two common responses to these challenges are:

- Throw up your hands and say, “We just won’t automate until the environment stabilizes”

or

- Dig in your hills and say, “We need to make the framework even more robust to handle the changes.”

The first response never works, because the environment never stabilizes. The second response may appear to work at times, but it’s probably an illusion. Strong frameworks require strong support from upper level management and your project team; absent that, the framework will probably not be effective.

**Project Scenario**

This scenario describes an automation experience that I had with a Scrum project on which I worked. I was hired primarily for test tool support, but was immediately thrown into several other tasks. Although I was focused on several tasks, I kept my eyes open for areas that could be supported by tools. For example, there was a request for assistance in creating test users through a GUI separate from the application under test (AUT). Someone on the team previously tried to automate the user creation process with the functional automated test tool that was on the project’s proverbial shelf. The attempt failed because the tool’s record & playback feature couldn’t record actions in the user creation application. The test team lead, therefore, enlisted about 3 people to manually create more than 700 users that needed to be created, a task that would collectively take at least a full day. Upon assessing the situation, I found that although the application couldn’t be recorded, I could still use a combination of Send keys and the application object model to automate the user creation process. The automation took no more than 30 minutes and all 700 users were successfully created within 2 hours, so not only did the process take less than a full day, the other two resources didn’t have to suffer through manually implementing the repetitive task of creating users.

Another opportunity presented itself upon learning that the team was regularly producing a requirements traceability matrix (RTM) manually. The team owned a test management solution, but it did not produce the RTM in the desired format, when used directly out-of-the box. It’s worth noting that it wasn’t officially called an RTM, because the project didn’t like the word ‘requirements’. The official name was Acceptance Criteria Traceability Matrix, but everyone called it an RTM. Anyway, since I was hired for tool support, I was able to customize the tool to produce the RTM in the desired format, a move that was very well received by all. Unfortunately, once I began producing the RTM directly from the test management solution, it became clear, based on the useless data coming out of it, that the tool was not being used properly. Since management was now aware of the usefulness of the test management tool, they decided to allow me dedicated time to set up the tool, and produce detailed documentation for the framework so others could understand and use it. When I presented this to the team, they loved it! They congratulated me on creating lasting scripts, and on producing thorough documentation. The problem arose when it was time for others to actually start developing automated scripts. They had no interest in learning the guidelines for developing automated tests within the framework, nor did they have any interest in using the documentation. The automated test framework was unsuccessful at being institutionalized into the project, so the automated tests were ultimately still fragile.

What ended up being more successful was when the project team got together and made the decision to slightly segment the testers into two groups. One group focused on the daily Sprint testing and interfacing heavily with development, while the second group focused on big picture validation and interfaced more with the user. The Sprint focused testers focused on manual testing of acceptance criteria, while utilizing

...project constraints always require you to bend and adjust what you plan on doing.

...
small, quick and easy to develop functional automated tests created by developers for front-end “unit” testing. The tests weren’t very maintainable, but nobody cared (at least initially), because the focus was on quick returns. The big picture testers focused on gaining a broader understanding of the system beyond what was stated in acceptance criteria, ensuring the customers expectations were being met and on developing maintainable automated tests and test components that would cover some of the acceptance criteria in a traceable fashion. These automated tests would primarily be used in later Sprints and during FIT (Factory Integration Testing). Again, I must reiterate that this article is not meant to assert or deny the merits of this specific approach, but it is meant to illustrate the importance that agility has to successful automated test implementation.

Lessons Learned

Many might suggest that the team’s lack of interest in a working framework is their own personal problem, but I saw it differently. I wasn’t interested in simply displaying my ability to create a framework, but instead interested in creating something that would be effective. So it was important to assess the above case study to determine what worked and what didn’t.

What didn’t work:

• Too Much Structure – I was no stranger to being able to evaluate an environment, and design a framework that works best with that environment. Unfortunately, I’d underestimated the resistance that a highly structured framework would have in an Agile environment (at least the ones in which I worked),

What worked:

• Broader view of automation – In the beginning, my focus was more on the broader view of test automation. This view defines test automation as tool support for all aspects of a test project, and not just automation of test execution. Approaching test automation from this point of view worked well, because it provided quick returns, which is what Agile thrives on. With its small teams, short cycles, and daily Scrum meetings, Agile is built on receiving regular, small bursts of information and progress. Anything that automation can do to facilitate that is a plus. And as the case study revealed, this approach may also result in more freedom being given to test automators to explore and attempt more robust functional test automation solutions, as it did for me.

• Reduction in structure – In the name of maintainability, robustness and a host of other quality attributes, we as automators can sometimes create frameworks that aren’t used enough to justify the effort put into ensuring these quality attributes are met. Sometimes the conventional wisdom that is used to determine when a given quality attribute is important, just doesn’t apply. For example, an organization that is extremely dynamic and has many environments in which testing must occur, typically calls for a framework with increased maintainability and robustness. That didn’t work in the scenario described in this article. By focusing more on usability, and reducing the structure, the automation was much more useful for the Sprint focused team.

Making Your Automation More Agile

One of the most powerful approaches to making your automation more agile involves expanding your view of what test automation is. As previously stated, the broader view of test automation defines it as tool support for all aspects of a test project. Some examples of implementing test automation in the broader definition are as follows:

• Data Creation – The creation of test data often involves highly repetitive, yet simple tasks. Therefore, it is a great candidate for automation.

• Report Generation – One of the most costly, time consuming overhead items in IT is the production of reports, yet this also is very repetitive. Automating the
process of generating reports will surely make you a hero on the team.

- File Manipulation – Shuffling, renaming, combining files is a task that many projects find themselves involved in. These tasks can often be easily reproduced and executed more quickly in a script.
- Test Setup – Test cases typically have preconditions that must be met prior to the execution of its procedures. These preconditions often involve setting up data and bringing the system to a controlled stable point. This can usually be automated quite simply, because the automated tests don’t have to be concerned with complex verification, but merely exercising the application.

Another powerful approach to making your automation more agile is with low structured, simple automated tests. This may include:

- Simple Data-driven Scripts – Automated tests that increase overall test coverage by automating the verification of boundary conditions and equivalence partitioned data that might not be verified manually due to lack of time.
- Scripts That Exercise Functionality – Normally, experienced automators frown on automated scripts that don’t perform much verification. At times, however, there is something to be said for scripts that merely exercise functionality with some really high-level verification, as long as those scripts are meant to supplement existing manual tests, not substitute for any of them. Such scripts may increase confidence in the AUT.
- Decrease the Maintainability and Robustness of Framework – Sometimes the people and processes of a project dictate reductions in quality attributes that technical goals may suggest are necessary. Doing so could reduce your estimated ROI, but creating a framework should always be about finding what will be most effective in the environment with which you are presented.

So in conclusion, whether working on an Agile project or not, one thing holds true: If your automation is not agile, it’s fragile.

---

No Time For Training?

With Rice Consulting’s e-Learning courses you can build your skills any time, any place at an affordable price!

- ISTQB Foundation Certification
- Basic Software Testing
- User Acceptance Testing
- User Requirements
- Plus many other courses!

Visit riceconsulting.com for details.
Over the years, I have had the pleasure of helping many people in a variety of organizations implement software test tools. In that time I have observed what works and what doesn’t work.

Most people agree on two big challenges of test automation:

1. Unrealistic expectations and
2. Getting management support for test automation.

Of course, there are other challenges, but these are the big ones.

On one project in particular I experienced first-hand how my client skillfully handled both of these challenges in one effort.

A Little Background

First, let me explain a little about the business domain. This domain is highly complex both in terms of technology and business application - the financial sector. This organization was dealing with four different technology platforms and already had test automation in place for three of them.

I was hired to assess the level of test process maturity and the readiness to introduce test automation on the platform with no automation in place. To their credit, this client chose to get the process assessed before we embarked on a tool search.

One final piece of information is that the test automation in place was pretty dysfunctional. The automation on one platform had not been maintained along with the application, so it was unusable. There was no integration between the tools, and the technology transfer between
the consultants that created the automation and the client staff wasn’t working, either.

Why is This Important?
Place yourself in the role of a senior VP in this organization for a moment. One of your test managers comes to you and says that in order to keep up with the rapid pace of delivery - a pace of over 500 releases in the main system per year - test automation is needed and it will probably cost over $250,000.

You know that the company already owns three tools, one of which isn’t currently in use, and can’t be used for the proposed situation. The two that are being used are not being used very effectively.

So you fold your arms and ask the test manager to make his case.

Now, place yourself in the role of the test manager. How would you make your case?

Here’s What My Client Did

1. He didn’t promise quick results

Instead, he promised an effort that would start small and grow well. He told senior management that it would take 12 to 18 months to lay a good foundation for test automation. Understanding the value of early successes, however, my client also established interim milestones that would help build momentum and solicit buy-in for continuing effort.

2. He didn’t oversell the benefits

Instead, he gave the reasons why the tool was a necessity instead of a convenience.

3. He got the process and skill sets in place first

Instead of placing the focus on a tool and what the tool could do, my client concentrated on understanding how the tool would be applied. He also knew that the tool would only be as successful as the people who used it.

Why Did This Work?

1) My client had credibility and management trusted him.

This is why I say that credibility is probably the most important attribute a tester or test manager can possess. Credibility equals trust.

2) My client had the courage to deliver an unpopular message.

It wasn’t easy for my client to temper expectations, but he knew that unrealistic expectations were a set-up for disappointment.

3) There were at least three past examples in the company of how diving immediately into using a tool had failed or delivered less than good results.

This gave some rationale for trying a different approach - start small and grow.

4) My client backed up his approach with good information and frequent updates to his management.

Being forthcoming and timely with updates to management, is the reason they found my client to be highly credible.

5) The company invested in their people to build skills.

The team hadn’t been trained in years. We laid some basic foundations in test design and understanding what makes a good test. We also trained the team on designing tests with automation in mind, and on how to work well together.

6) We did a proof-of-concept (POC) instead of a quick evaluation.

This was huge. In our first POC, the consultant provided by the vendor of the tool we chose for test automation actually completed the automation of an entire application! It was a win-win-win. The vendor sold tools, my client looked great to his management, and I looked great for suggesting the idea.

7) We were able to show some quick wins to build momentum.

These early successes allowed us to learn lessons with lower risk. The momentum from those early wins let us ramp up even faster. By
going a little slower at first with automation, we got faster quicker.

You Can Do This, Too

You may be thinking that you can’t picture your management accepting this approach. Believe me, I understand. However, someone has to manage expectations.

I don’t have a formula or process to build this kind of foundation, but I do have some principles:

1) Be open and honest with your management.

When there’s bad news, get it out fast while there’s still time to develop a reasoned response. In my experience, bad news usually does not get better with age.

2) Keep the lines of communication open.

You may need to remind your management that you are building a foundation or framework.

3) Go for the quick wins and publicize them.

These successes don’t have to be big. Look for situations where time was saved, testing was done better, and/or people are freed up to do other work.

4) Get the process and skill sets established first.

Before you get on stage and start playing the instrument, learn a song first. I know that’s not about testing, but it’s a principle that also applies to testing. The process and skills guide your decisions about what, when and how to test.

The tool will likely force changes to your process - and that’s fine. Tools, people and processes usually grow together.

5) Be flexible.

Be willing to try new approaches, even in the process of deploying the tool. Listen to the team and try their ideas.

6) Learn from your mistakes (and those of others).

I like to learn from others’ mistakes, since that’s less painful than learning from my own. However, we all make mistakes, so we should learn from them. Unless you have a time and place to reflect and discuss these lessons learned, you probably won’t do this. A great book on the topic is Project Retrospectives: A Handbook for Team Reviews by Norman L. Kerth.

Summary

I hope you’ll take this case study to heart. It overcomes two of the biggest challenges in test automation - unrealistic expectations and getting management support.

You don’t have to be super-human to do something similar. You do have to educate management and work with them to understand the issues, risks and benefits of test automation.

If you can keep expectations in line and keep management on your side, you will find that in a year or so, you will be in a place where the elite minority resides - in the successful implementation of test automation.

X-Path Marks the Spot (Continued from page 9)

[“textbox”, “id”, “userID”]

GetElement(“Class:=Button”, “Text:=O Button”).Click

you’ll use a statement that contains the logical description, such as the following:

GetElement(“OvalButton”).Click

In this example, an update to the GetElement function is required.

Conclusion

Object Maps are very powerful tools for adding robustness to your automated test framework, so if your tool/framework doesn’t offer the capability, you may want to consider developing your own. If you do decide to develop your own Object Map to reach the treasure of enhanced robustness, remember that X… or rather, XPath marks the spot.

For code samples of this and other techniques visit http://www.techniques.automatedtestinginstitute.com
This Week In Automation Series

*This Week In Automation* is an ATI produced video series that summarizes current events relative to test automation. With an often tongue-in-cheek approach to discussing the latest news in tools, events (both small and large) and concepts, this series serves as the video companion to ATI’s newsletter, while keeping you abreast of what is going on in the world of software test automation. See the latest edition of *This Week In Automation* at http://www.news.automatedtestinginstitute.com.

SEE No Evil, HEAR No Evil

Have you seen the latest video tutorial or podcast produced by ATI? If not, visit http://techniques.automatedtestinginstitute.com for a tutorial on calculating Automation return on investment (ROI). Then, when you’re finished with the tutorial, navigate over to http://podcasts.automatedtestinginstitute.com and listen to the latest podcast featuring an interview with Ms. Elfriede Dustin. In this interview, she discusses her latest book entitled, “Implementing Automated Software Testing”.

It’s NEWS To Me

Over the past few weeks ATI has brought the user community some interesting and very popular news stories related to test automation. These stories include:

- ‘Software Problem’ Results in Shocking Letter To Parents of Dead Teen
- Espionage and Software: U.S. Vulnerable to Cyberspies!?  
- Software Testing: The Fastest Growing IT Service

For these and other stories visit http://www.news.automatedtestinginstitute.com.
Wanna know which of the hundreds of indexed tool and article archives are the most popular? Well, here goes. For the purposes of this section, popularity is based on the number of hits and the rating received. Two of the most popular indexed tools on the site are JSystem and Selenium. Two of the most popular indexed articles are “Test Automation ROI” by Dion Johnson and “Test Automation: The Promise Versus The Reality” by David Dang. Be sure to visit http://www.tools.automatedtestinginstitute.com and http://www.articles.automatedtestinginstitute.com for the latest in the increasing index of tools and articles, and be sure to rate them after you have read them!

Test automation is an integral part of ensuring the production of quality systems in a world where software development is becoming increasingly fast-paced. And one way to help elevate test automation so that it can be more effective is to celebrate it and its practitioners. Thus, the ATI Automation Honors have been created to celebrate excellence in the discipline of test automation from the corporate level down to the practitioner level. This celebration specifically pays tribute to:

- Those that have displayed leadership in moving and keeping test automation in its proper place as a distinct IT discipline,
- Those that drive innovation within the field, and
- Those that display excellence in automation implementation, thus playing a big role in the delivery of a high quality product to customers and/or production.
- Tools, individuals, publications, websites and more.

The nomination period is from April 1 through June 30, 2009. Finalists will then be identified by August 1, at which time voting will begin and commence through October 31, 2009. For more information on the Automation Honors, the nominating process or the honoree categories, visit http://www.atihonors.automatedtestinginstitute.com.
Wanna know what’s hot in test automation?
Just do a Google search on the term “Implementing Automated Software Testing” (IAST), and you’ll find out quickly. With approximately 3 million search results, this is clearly a very popular term, but not only that, it is now the title of a new book written by Elfriede Dustin, Thom Garrett and Bernie Gauf. We at the Automated Testing Institute aren’t sure if the title is coincidently identical to one of the most popular test automation terms, or if it was a genius marketing ploy to get more attention drawn to the book; whatever the reason, it is definitely aptly named. This successor to Automated Software Testing (AST) – a 1999 book also coauthored by Dustin, and also a term with approximately 30 million Google search results (wink, wink) – wastes no time in picking up where its predecessor leaves off. AST largely focuses on how test automation may fit into the overall software development and testing lifecycles, IAST lends more of its attention to the test automation effort itself and the details of managing an automated software testing effort.

IAST does an excellent job covering the following Automation Body of Knowledge version 1 (ABOK v1) skill categories (visit http://www.abok.automatedtestinginstitute.com for more information on the ABOK):
• Skill Category 1: Automation’s Role in the Software Testing Lifecycle (STLC)
• Skill Category 2: Test Automation Types and Interfaces
• Skill Category 3: Automation Tools
• Skill Category 4: Test Automation Framework Design
• Skill Category 5: Automation Framework Design
• Skill Category 12: Automated Test Reporting

Part I does an excellent job addressing Skill Category 1 of the ABOK in that it addresses the ‘what’ and ‘why’ of test automation. Chapter 1 sets the tone of the book with a “unified” definition of test automation; a definition that also works to distinguish test automation from manual software testing. This chapter, along with its reference to Appendix B, also addresses ABOK Skill Category 2 with a discussion of the “typical testing types that lend themselves to automation”. Chapters 2 through 4, round out Part 1 of IAST, and continue to further address ABOK Skill Category 1 with topics including: reasons for automating, making the business case for test automation with return on investment (ROI) calculations, and how to avoid typical test automation pitfalls. The latter half of Chapter 4 also addresses ABOK Skill Category 3, by discussing tool evaluation and selection. This discussion is aided by Appendix C which expands into tools and tool requirements not only for automating tests, but also for automation of processes such as requirements management, defect tracking and security testing.

Part II of IAST graduates from the ‘what’ and ‘why’ of test automation and delves into the ‘how’ of test automation. Not ‘how’ in terms of how to develop scripts, but ‘how’ in terms of how to create, implement and monitor an automated test framework; and these are items found in Skill Categories 4, 5 and 12 of the ABOK. Part II of IAST begins with Chapter 5, which discusses how to define requirements for your test automation effort. Chapters 6 and 7 build on this by discussing how to take these requirements and develop a compatible strategy and framework. Chapters 8 and 9 cover the next logical steps of defining metrics to track test automation progress, and the implementation of the automated software test framework processes. Then finally, Chapter 10 closes the main portion of the book with a broad discussion of the skills required for test automation implementation.

In summary, IAST is strong with respect to test automation from a macroscopic level. This makes it a great resource for managers, leads, and anyone that is responsible for or will take part in defining and implementing a test automation effort.

The Automated Testing Institute (ATI) conducted a podcast interview with Elfriede Dustin, where IAST was discussed. To listen in, visit http://www.podcasts.automatedtestinginstitute.com
Tired of searches that return irrelevant information?

The Automated Testing Institute has partnered with Google to create a software test automation search engine. If you’re looking for software test automation information, you will get the best results by using this search engine, because it only searches the sites that matter most to automators.

www.googleautomation.com
WHICH ONE ARE YOU?

“Black Hat events are the premiere cyber security events you can attend... (with) the hottest, most relevant speakers and leaders in the field.”
—Jim Christy, Director of Futures Exploration, DOJ Cyber Crime Center

THE BAD GUYS KNOW YOUR TRICKS. DO YOU KNOW THEIRS?

Are you the kind of security professional that craves the freshest information? Are you the kind of hands-on network defender that needs to be the first to know? If you stand out from the crowd because you’re seriously committed to security, then you’re our type — the Black Hat type.

The Black Hat Briefings are the most important technical security events on the planet, and we create them for people just like you — people who know security and want to be part of the solution. Join us in Las Vegas for 10 full tracks and over 60 intense hands-on training sessions of timely, actionable security information.

Black Hat USA 2009
July 25-30 Las Vegas, NV