WORKSOFT WHITEPAPER

Future Proof Your Test Assets: Protecting Against Obsolescence

— Linda Hayes, Chief Technology Officer, Worksoft, Inc.
The only certainty in technology is change. New hardware platforms, operating systems and network protocols emerge that deliver more power, programming languages are created to exploit them, architectures are designed to leverage these advanced capabilities, then new applications are written to convert their potential into operational efficiencies and competitive advantages.

With all of the benefits come costs, of course. Aside from raw acquisition expenditures, employees require training in new skills, learning curves must be endured, development investments have to be made, and data conversions may be required.

Because of these costs, changes are rarely made for their own sake: the promised reward must exceed the investment. This explains applications that are still operational today despite the fact that they were developed decades ago; in many cases, they have been given facelifts in the form of new interfaces, but their core capabilities are left intact, memorialized in COBOL or Assembler or their language of origin simply because they still work.

In some cases, however, changes must be made involuntarily. When a vendor sunsets a particular product, for example, without providing a forward-compatible path to a replacement alternative, companies may find themselves being forced to take on the costs of change without an offsetting benefit.

An example of this predicament is occurring right now in the test automation industry. Vendors including Mercury Interactive, IBM Rational, and Compuware have all announced replacements for their test automation scripting tools that have no migration path for test assets developed with their previous versions. And while none of them have formally discontinued support for their legacy products, their decision to only provide support for new development platforms in their replacement products has the same result, since most enterprise business processes necessarily span both old and new applications and therefore cannot be effectively tested without changing tools.

This white paper explores the reasons why these tools are uniquely vulnerable to changes, the cost implications for enterprise users of these tools, and a future-proof strategy for assuring that these costs never have to be repeated.

**Why Test Tools are Vulnerable to Change**

Unlike programming languages that are used to develop applications, test automation scripting tools—used either standalone or as part of keyword-driven implementations—cannot satisfy their intended purpose of driving the execution of another application without being able to interact at runtime with the application under test. That is, by definition scripting tools derive their value from compatibility with other languages. This makes them especially vulnerable to technology changes.

Furthermore, scripting tools are themselves languages, and while usually proprietary at some level they often emulate the syntax and structure of more common development languages. The Mercury WinRunner TSL language, for example, resembles the C syntax while Rational Robot was built around Visual Basic and Compuware QA Run adopted a syntax of its own. Users of these tools must learn these proprietary languages and develop expertise over months or more of consistent usage.

Recently each of these vendors has released a new product using a completely different language syntax. Mercury began marketing QuickTestPro and Compuware released Test Partner, both of which are based on VBScript, while Rational’s new Functional Tester uses Java. And although there is nothing inherently wrong with the new languages they have adopted, they are completely different than the previous tools and the implications for their customers are drastic.

**Implications of Test Tool Changes**

Test scripting tools require that tests be written as code. A robust test library may consist of hundreds of thousands of lines of code; in fact, it may rival the size of the application under test.
Aside from the obvious training, learning curve, development and maintenance costs for this code base, the fact that tests are expressed in code means that if the vendor changes languages, a rewrite is required. Granted, utilities may be offered that promise to convert a percentage of the code, but moving between two completely different languages can never be totally automated. Ultimately such a change requires manual modifications, debugging and testing.

And not only must this massive code base be rewritten, but employees must learn the new language and experience a new learning curve. One large ISV estimated their cost to convert from WinRunner to QuickTestPro at $20 million.

Given the rapid rate of change in most organizations, just keeping test scripts current with the latest application version is challenging enough, but the maintenance effort at least yields a return in terms of test reuse for the next version. But spending the time to rewrite or convert the entire script library to a completely new language offers no such benefit, it is simply a cost to maintain what already exists. It is like rewriting an application in a new language without any improvement in functionality.

The immediate impact, then, of changing test tool languages is nothing less than a redevelopment of an existing code base without any offsetting return on investment.

But the long term implication is even more sobering. VBScript itself is already a sunset language, being replaced by ASP.NET. So enterprises who bear the cost of converting to QuickTestPro or Test Partner will soon find themselves in the same predicament within only a few years—having to convert their test libraries yet again.

The lesson is simple: Any time a test asset is stored as code, there is a virtual certainty that it will become obsolete. It is only a matter of time.

**Future-Proof Your Test Assets**

There is only one way to assure that your test assets are protected from technology changes, and that is to store them as data.

Think about it. Most enterprise IT assets—customer, employee and financial information—are stored as data that is accessed and manipulated by software. The software can be changed without necessarily changing the data at all. Shifting to a completely new application may require a data conversion to a new format, but the cost is only undertaken if the new system offers substantial offsetting benefits.

The only test automation solution on the market today that stores all test assets as data is Worksoft Certify™.

Certify employs a patented, open API that allows any test tool or language to be used to execute test cases automatically. The only required code is a compact, completely generic library of functions based on the class library of the application development language that in most cases is supplied and supported by Worksoft. This code contains no application specific content and is reusable across applications as well as enterprises. The cost of changing from one language to another may be zero if already supported by Certify, or a maximum of four to six weeks if completely new.

This radical new architecture also means that enterprises are free to mix and match technologies based on their needs, without being held hostage to decisions by external vendors. And if there is a benefit to changing languages, the cost is minimal; the test cases themselves are undisturbed.

Another benefit of Certify’s test asset database and open architecture is that complex business processes spanning multiple applications and platforms can be tested seamlessly in a single session. Certify simply mixes and matches the appropriate technology as needed. Thus, a single test session can use one language to drive a Web application, another to drive a mainframe, and yet another for an XML message layer – all without affecting the way the tests are defined, stored or managed.

The benefits extend to user training and learning as well. Certify users do not have to receive any additional training if the underlying technology changes: the interface behaves exactly the same. Just as driving a car would not be different to the driver whether the
engine had six or eight cylinders, using Certify does not vary with the underlying language. This means Certify users are free to focus on extending their test assets to improve coverage and quality without losing time and resources to costly code conversions or rewrites.

**Progression Testing: Moving Forward**

Aside from future-proofing your test assets and avoiding costly conversions, the Certify model is the only one that supports automation for both new features and regression tests, enabling rapid and agile development approaches.

Because Certify does not employ a record/scripting model, it does not require that the application under test be complete or even available in order for tests to be developed. User stories can be defined as Certify processes in the database repository, then linked later to an execution library when the application is ready. This means that tests can be developed concurrently with the software, reducing overall cycle time and improving communication with developers about expectations and deliverables.

The tests written for new features differ not at all from those developed for existing functionality. Both follow the same structure and content. This combination of verifying forward progress as well as protecting against regression is known as Progression Testing, and only Certify has it.

**Summary**

If your enterprise is like most, you are experiencing ever shorter delivery cycles for ever increasing functionality and complexity. You must focus your resources where you can get the maximum return. That means you must protect your investments from obsolescence and prevent incurring involuntary costs that yield no benefit.

Future proof your test assets with Certify and move your testing forward into the future with Progression Testing.