



RadView Software White Paper

Elevating the Intellectual Property of Test Engineers

By Ilan Kinreich, Chief Strategy Officer and founder of RadView Software, Ltd.

Do you believe software quality and the intellectual property (IP) of a software application is important? If yes, then what about the IP of the software application testing process? I believe this testing process IP is just as important. If a company is committed to producing quality software, then it should aim to preserve the IP of the testing team, just as it would its development or manufacturing team.

I believe the IP of the testing team is the documentation of a well-defined quality assurance process that captures the test plans, executions, and results of a successful test team. When these pieces are documented, analyzed, revised, and improved, the test process is more than “bug hunting”; it’s a fundamental plan that is part of the overall business goal to produce and deploy successful software applications.

To protect and elevate the process that enables this success, you must be able to capture and illustrate it. And to illustrate the process, you must understand what each component adds to the test and QA organization’s IP. In this article I will define what I believe are the critical elements of test process IP and what can be done to elevate and improve these elements to produce higher quality software.

Based on my experience, I believe the key components are test scripts and reporting, the construction of the testing system itself, and the documenting of all the testing elements. To elevate these individual testing components to the level of IP, the adoption of Open Standards is crucial. It simplifies the writing of test scripts and reporting and it will ensure interoperability and the exchange of code and data between testing systems, something that rarely happens today. This will lead to standardization in the way that software is tested, evaluated and reported on, ultimately leading to higher quality software.

Benefit of Open Standards

The two main areas that will benefit from the adoption of Open Standards will be test scripts and reporting. Test scripts are at the heart of many test automation systems – defining both single test cases as well as complete test automation execution. Test engineers today use many computer languages to write scripts. While some test systems offer proprietary scripting languages, other systems use programming languages or open script level languages. I would argue that an open script language such as Python or JavaScript should be used to access the vast availability of the resources of an Open Standard language and to avoid the unnecessary complexity of a programming language.

Another function that would benefit from the use of Open Standards would be the reporting of results. Imagine if there was a standardized data structure allowing you to compare results between different systems. XML offers an excellent choice for this function with its ability to define the meaning and structure of data. We could start to share a common way of judging the performance, integrity, and overall quality of a software application. Furthermore an XML-based Test Database system could also offer a standard for sharing test execution and test planning information. A new test engineer introduced to an already existing testing system could easily augment information stored in such an XML-based database.

Building Blocks

Today a lot of application testing is done through the execution of the application’s User Interface (UI), commonly referred to as GUI testing. This is an excellent

example of the power of a standard as the UI provides a common interface to any kind of application, and testing from the user point of view is an excellent way to verify application behavior.

But, this idea calls for a second-generation improvement. Today, in many test systems an overuse of the record/replay function tends to create long, not well-documented and non-maintainable test code. It is far more efficient to build a test system based on a "Building Blocks" approach. Each Building Block concentrates on one test case or specific UI operation. The largest test plan is then constructed of a combination of these Building Blocks, thus providing a well-documented, flexible, and maintainable test system. This approach saves time, promotes re-use of test elements and ultimately ensures a higher quality software application.

Application Testability

Then there is the issue of the testability of the application code itself. Currently, it is often the case that the application code poses huge burdens on the testing system. For example, application UI objects are built with no identification or by using non-accessible elements that makes the testing phase very difficult, less productive, and sometimes impossible. With the high importance of software quality today, application code needs to be designed with test in mind.

To use an analogy, a car designer would not use a component that the manufacturing team could not assemble and in the same sense application developers should verify that the GUI component they are incorporating can be properly tested. Available tools that can "check for testability" would be a great help in the process, ensuring a smooth path for a quality software release.

Version Control Software

Use of version control software for test scripts and building blocks is also key. This tracks the evolution of the scripts, when they were used, and the results- all important information and part of the documentation of the test process. Within the version control system you could integrate the testing process right into the software build system to increase test automation.

The following are key areas for organizations to focus on to increase the value of their testing IP:

- Insist on using Open Standards - use XML for any interchangeable or reported data. Use openly available scripting languages to control the execution of your test system.
- Insist on building your test system from integrated elements - each of these elements being well documented and easy to maintain.
- Make sure your Test System endures changes easily, minimizing the impact of a single change to a single element and maintain version control of your test components.
- Insist on application testability as a design rule. Each element used by your software needs to be accessible and testable by your testing system.
- Accept the need for different skill sets within your test organization. Understand that you need test planners, test architects, designers and test engineers in your organization playing the different roles in building your test system, just as you have these functions in your application development organization.

- Promote industry standards for every key element of the testing system – and encourage others to adopt the same principals.

These are what I consider to be the important elements of test and QA department's IP and what I believe it takes to produce high quality software today. While some of the issues in this article fall outside the realm of an individual, such as the industry-wide adoption of an open standard for a scripting language, companies can take steps in that direction. Certainly documentation, application code written for testability, and use of building blocks to comprise a complete test system are easy to implement within a company with a little discipline. The time required for these activities will pay off in more efficient testing and higher quality software. In short, this work is what needs to be encouraged to advance the way we develop and deploy software today – to ensure high quality software applications.

RadView Software, Inc.

7 New England Executive Park
Burlington, MA 01803
Phone: 781-238-1111
Fax: 781-238-8875
Toll Free: 1-888-RADVIEW

RadView Software, Ltd.

14 Hamelacha Street
Park Afek
Rosh Haayin, 48091 Israel
Phone: +972-3-9157060
Fax: +972-3-9157683