

Peer Reviewed: Software Validation: Can an FDA-Regulated Company Use Automated Testing Tools?

By **Janis V. Olson** Jan 9, 2013 2:15 pm PST



Getty Images

ABSTRACT

Software test tools are very useful for development, validation, and maintenance of computer systems in regulated industries and can absolutely be used. Key aspects in an approach for use of these tools are provided. Two examples of tools are described. The use of software testing tools will increase the quality of software applications by efficiently detecting functional, performance, and security issues.

INTRODUCTION—WHAT ARE SOFTWARE TEST TOOLS?

Software test tools help development and testing teams verify functionality, ensure both the reliability and security of the software they develop, and investigate software bugs. Off-the-shelf tools are available for all stages of software development. Examples include static code analyzers, record and replay, regression testing, and bug tracking. Some software testing tool vendors offer an integrated suite that starts with the gathering of requirements and continues through software development and testing throughout the life of a project, including supporting the live system. Other vendors concentrate on a single part of the application development life cycle, such as just testing.

HOW DO YOU USE A TOOL SO THE US FOOD AND DRUG ADMINISTRATION WILL ACCEPT IT?

FDA will accept use of a software testing tool if the following are done:

- Know, understand, and document the intended uses of the tool
- Test and verify the tool in the context of the intended uses
- Document the validation of the tool
- Maintain the validated state

Know, Understand, and Document the Intended Uses of the Tool

Companies, project teams and individuals only get what they want if they know what their requirements are for how they are going to use the tool. So the first task is to understand the intended use requirements and get agreement that this is how the tool will be used. Understand the native functionality of the tool when used “as is” and what needs to be configured or customized to get the performance needed.

Plan what to do in a validation plan or a project plan. Identify what will be done by whom when. Then, go to work on setting up the tool. Documentation of the process is the key to success. Document what can be done with the tool's native configurations and any customizations needed to assure that the tool will work as expected. Review the native functions and the configurations and map them back to the intended use requirements to assure that all the needs of the tool as stated in the intended use requirements have been covered. Document the intended uses of the tool in the context of the processes that will use the tool in written procedures. Procedures need to be written at the level needed by the people in the organization who will use them.

Test and Verify the Tool in the Context of the Intended Uses

One should use code already tested or verified you have through other means. Determine the amount of testing/verification and the extent of testing that needs to be done based on the risk of the failure of the tool. Low-risk tools still require testing and verification that the tool will meet one's intended uses. Document your testing with test cases or scripts. Document the results; including the objective evidence of the actual results that can be compared to the expected results. Do not just record that the tool worked "as expected" or "passed". Use good and poor code to test the tool. Keep those pieces of code as part of the test documentation. Do fault insertion to assure that the tool behaves properly in all the expected scenarios. Do testing in the context of the intended use. Understand that risk is based on the intended use of the tool. Know the tool's limitations. Each tool will solve problems but also create potential issues. Understand what these potential issues are and their consequences. Mitigate the possible risks possibly as part of the procedures for the use of the tool.

Document the Validation of the Tool

Validation is more than just the testing. It is the linking of all the activities and documentation to one another. The intended use requirements are mapped to the tool's native functions and any special configurations and/or customizations. They are also mapped to the testing scripts. The executed scripts have the documentation of the objective evidence of the results that can be compared to the expected results. Once all this documentation is together and linked, this is the assembled evidence that the tool is validated for its intended uses. Write a summary report of the effort. Train the users and use the tool.

Maintain the Validated State

Once the tool is validated, monitor the use and the results of the tool. If the users find that they have additional uses that were not validated, those will need to be validated prior to use. Likewise, if users find new "expected scenarios" that were unknown at the time of the original validation, additional documentation of the behavior of the tool under these new scenarios must be placed in the validation file and tested. Monitor for additional issues and their consequences and evaluate the risks. Document new risks as part of the risk analysis.

TWO EXAMPLES OF COMMONLY USED TESTING TOOLS

Static Code Analyzer

A static code analyzer is a commonly used code review tool. FDA recommends the use of static code analyzers in high-risk applications, especially for medical device software. Static code analyzers will find issues that software engineers know are not associated with the basic code. Many false positive issues are found that software engineers then have to evaluate and investigate. Static code analyzers have coding rules that they enforce. If the code does not conform to their rules, the software engineers will get many issues that need to review and resolve. Once the software engineers have resolved them, this information on the false positives routinely identified by the tool can be used so that the next time the false error is found, in the same way by the static code analyzer there is no need to do another investigation. Some companies change their coding standards to conform to the tool for all future development. However, because they often have built code on old code bases, they maintain a list of false errors they will ignore or not investigate fully. Naturally, this is based on the risk of not investigating these errors.

Static code analyzers also allow software engineers or the tool vendor to do custom configurations. These customizations tell the analyzer to ignore some of the false errors that the tool finds but that the developers know are not issues with the code and can be ignored. A company should do extensive research to determine the best static code analyzer for their needs. Code modules with common errors, complexities, and poor coding practices as well as modules with good code should be developed if not already available to the software engineers. Several different manufacturers' static code analyzers should then be evaluated. The tool that found the most types of errors and issues should be chosen. Limitations of the static code analyzer should be documented so other verification methods can be used to find errors the static code analyzer missed or

was not intended to find. A list of common false errors the tool found should be compiled. The false code errors, the investigations into each one, the results of the investigations, and what the software engineers would do with each false error when found in their code modules was prepared. These became part of their standard operating procedures in the use of the tool.

“Record and Replay”

Another commonly used tool is an automated “record and replay” testing tool. These tools allow testers to record each keystroke as they are running a test manually. The testing team then can schedule testing to be performed any time in the future and the tool will replay each keystroke as if a tester was doing it manually. The advantage of this approach is that the documentation accumulated while recording, such as screen shots or other documentation, is used not only to show that the code passed the test when run manually but also used to compare the tool output the very first time it is run and demonstrates, in fact, that the tool is running the manual tests and running them correctly. To optimize testing, many of these tools allow testers to modify the keystrokes rather than rekeying them, insert negative testing, or change if the code is changed and requires changes to the testing steps run by the tool. Modifications to the keystrokes outside the record feature of the tool should be documented and independently reviewed to assure that the changes are done appropriately and correctly. The documentation of the keystrokes is the code for the tool. After validation of the tool, testers can have the tool continue to provide the documented evidence of the test results, or chose after to only have the tool record that the tests passed or how they failed. All testing failures will need to be investigated and resolved.

SOFTWARE TESTING TOOLS ARE BUSINESS DRIVERS AND HAVE BENEFITS

The use of software testing tools will increase the quality of software applications by efficiently detecting functional, performance, and security issues. Compared to manual testing, which may be inconsistent because of human error, testing tools also are more consistent and repeatable in following the documented testing procedures/processes. Many of the tools can measure software performance throughout development, testing, and maintenance of the software product. Other tools can assist in conducting and documenting risk analysis and benchmarking against other software. The use of these tools can improve a company’s time-to-market or time-to-implementation because of more frequent and consistent testing, finding, and fixing of issues and bugs. This improves product lifecycle management processes and reduces your total cost of ownership.

Once the tool is in a validated state, use it over and over. Companies can be confident that the tool is giving them the information they need to demonstrate, to FDA, to their customers, and to all their stakeholders, that the software is being tested consistently and effectively. **JVT**

RELATED: What are the Most Important Competencies for Computer and Software Validation Projects?

Source URL: <http://www.ivtnetwork.com/article/peer-reviewed-software-validation-can-fda-regulated-company-use-automated-testing-tools>