Common Sense Systems

Whitepaper: Automated Testing of Complex Medical Devices

During development of complex medical devices, an enormous amount of testing is required. In addition to the usual quality assurance issues inherent in software and hardware development, quality system regulations like ISO 13485 and software specific standards like IEC 62304 require regression testing as part of the software development lifecycle.

Medical device developers have long since enabled automated device testing by writing test harnesses that allow remote operation of their devices. The next step, however, is to expedite the test process further by automating device setup, test artifact collection, and queuing of multiple tests.

We selected Jenkins, a powerful open-source application for executing and monitoring repeated jobs, as a tool to increase testing throughput. Many users use Jenkins to run software builds as well as to tightly integrate the build process with automated testing. We have focused primarily on using Jenkins for automated testing on remote devices.

Problem: Need to fully automate test execution

Running a test script involves several steps, including loading software onto the device, configuring software options, executing the test script, evaluating results, and collecting test artifacts. To maximize throughput, all of these tasks need to be managed without human intervention. If this can be accomplished, tests can run back-to-back, 24/7.

Solution: Use Jenkins jobs to execute complex processes

At its heart, a Jenkins job executes a set of scripted tasks. This means Jenkins is highly flexible – if a task can be scripted, Jenkins can execute it. In addition, a job may be broken into many steps, where each step is dependent on successful completion of previous steps. This means that a process can be terminated mid-stream if one of its steps fails.

Working with our clients, we have developed custom scripts to

- load software onto the target device
- set device options
- trigger execution of a test script
- collect test artifacts

All of these scripts are executed within Jenkins jobs.

Common Sense Systems, Inc.
18915 142nd Ave. NE, Suite 145
Woodinville, WA 98072 USA

Tel: 425.806.8744
Fax: 425.806.8407
Web: common-sense.com

Please send notification of errors, omissions or other suggestions for improvement to: documents@common-sense.com.

Problem: Need to maximize usage of scarce resources

When developing a complex medical device, it frequently doesn't make sense to spend a lot of money building lots of devices when the hardware is still under development. On the other hand, developers need to test early and test often, against the most current product version available, even if it isn't the final product. This can lead to a scarcity of resources available for testing.

Solution: Use Jenkins to manage a pool of devices available for testing

Jenkins excels at scheduling tasks for a pool of resources. Using Jenkins, a test pool of the latest device hardware can be kept in a centralized location and an entire team of developers can run tests against that pool. Instead of needing a device for every developer, where utilization of each system can be 50% or less, fewer devices can be shared without sacrificing access. In addition, maintenance and upgrade of the test pool is easier, because all devices are in a central location.

Problem: Need to route tests to devices with required resources

Many times, not all devices are equal. Some may have attached peripheral hardware that may be required for a particular test, or not allowed for another one. It can be a challenge to match tests to devices with appropriate peripherals.

Solution: Use Jenkins' built-in scheduling features to manage resources

Jenkins is highly extensible and configurable through the use of slave nodes, which are secondary computer resources available to the Jenkins server. Most importantly, each slave node may be labeled to identify node characteristics. Jobs may then identify required labels and when the job is run, it will execute on a node that has all of the specified labels.

We use slave nodes in a 1:1 relationship with devices in the test pool. By assigning node labels that correspond to the available peripherals for each device, we can ensure that individual test scripts identifying required resources will be automatically assigned to execute on a device that has those resources.

Problem: Need to modify resources available to Jenkins on the fly

Even the best developers sometimes create bugs. And while many errors can be recovered automatically, the complex nature of medical device development means that some errors require human intervention to recover. Therefore, there needs to be some mechanism to take a system out of the test pool if it becomes unable to run tests.

Solution: Use the Jenkins API to modify resource availability

One of the mode powerful aspects of Jenkins is its machine-consumable remote access API. We have extended the open-source Ruby gem available on GitHub, and we use this for programmatic control of Jenkins resources. Most frequently, we use it to remove a resource from the test pool if it is no longer responding to remote commands. However, we also use it for other tasks, such as automatically generating slave node labels when users set up or change hardware resources.

Problem: Need to archive test results and artifacts

In addition to recording the result of automated tests, it is necessary to save log files and other artifacts for developers to use when diagnosing test failures. These files need to be archived for anywhere from a week to years for use in debugging and process documentation.

Solution: Use Jenkins archiving functionality

Jenkins has built-in archiving functionality. Users can specify the types and locations of files in the job workspace to be saved. These files are then stored on the Jenkins server alongside the test result for a user-specified period of time. In addition, we also use Jenkins to run scripts that transfer test artifacts to long-term off-server storage.

Problem: Need to automatically notify interested parties of test results

Unfortunately, developers cannot always be relied upon to return to the test server to check on the results of tests they've scheduled, especially if those tests were in the queue for a long time or were launched automatically.

Solution: Use Jenkins' built-in email notifications

Jenkins jobs can be configured to automatically email test results to the user who launched a test and/or a fixed list of interested parties. The email content can be customized, but defaults to the test results and a link to the Jenkins test job. This greatly helps developers stay on top of test status and enables them to follow up on test failures quickly.

Problem: Need to stay in sync with the latest code changes

When continuous testing is a part of development, testing against a dynamic code base can be challenging. It is important to make sure that tests are run against the correct code set, especially when testing is occurring against several types of builds – developer, nightly, weekly, beta, and/or production – either simultaneously or during different project phases.

Common Sense Systems

Solution: Use Jenkins source control plugins to access the code base

Another of Jenkins' strengths is its extensibility. There is an active community of developers working on both the core application and a wide variety of plugins. There are plugins available to access most popular source control solutions, including SVN, git, and IBM Rational ClearCase. These plugins enable jobs to use the correct versions of code files from source control for testing.

Problem: Regulatory requirements for software tool validation

How well can you trust your test software? Regulations such as IEC 62304 require validation of all tools used in the software development process, in addition to final product code. Writing and executing a validation plan for software test infrastructure can be complex and time-consuming.

Solution: Common Sense Systems can help!

We are experienced in developing software tool validation plans and protocols. We can help develop a customized solution for your tool chain.

About Commons Sense Systems:

Common Ground, Common Purpose, Common Sense

Common Sense Systems was founded in 1996 by computer design engineer John Sambrook to help regulatory affairs and compliance engineering professionals at medical device manufacturing companies.

He believed (and still does) that a fresh, common-sense approach would radically improve existing methodologies and dramatically decrease the time required to bring new products into the competitive medical device marketplace.

Today Common Sense can boast of many successes in significantly and rapidly improved productivity and operational results for medical device manufacturers.

How do we do it?

Common Sense brings an expert team of software engineers and regulatory affairs experts to its clients. But just as importantly, we provide our clients with the management tools to report on the status of all regulatory affairs and compliance engineering processes and projects to all audiences within client organizations – from design to compliance engineering to engineering and executive management. This vital communication ensures that all product decisions are fully informed and aligned with overarching company goals.

Common Sense Systems

Is it magic?

In a word, no. In fact, the praise we are most proud of is when we are told by clients that, in the final analysis, our solutions are just common sense. To us, that means that our services seem natural, are non-disruptive to implement and deliver very obvious procedural and financial rewards.

We are located in the Greater Seattle technology hub, in Woodinville, Washington, USA. If you'd like to chat about how we can be of service, please contact us today.