Test What You've Built

AXON iVY
digitalize your business

OpenLegacy

ca technologies

cm FIRST
Rethink Modernization

About Your Presenter

IBM i Professional for 16 Years.

Primary Focus is IBM i Engineering / Programming

Well Versed in 2E.

Well Versed in RPG (All Flavors)

Well Versed in CM Products such as Implementer

Up to Date on Newest SQL/RPG Enhancements.

What exactly is software testing?

Software testing is testing what you've built!
As obvious as this would seem it really isn't.
Can't supply comments to customers and
other colleagues like "Well it compiled it should work." Has happened to me.
Testing is as much an art form as it is a science.

5 Reasons we need software testing.

1. To ensure what we create does what it's supposed to do.

   Does the new F10 button actually work? Did you call the wrong function? Was the data actually written to the database correctly? Did what you introduce as new code break code that was already in existence?

5 Reasons we need software testing.

2. What works with one person may not work with dozens or hundreds of people.
   A good stress test of an application can ensure that your website or even a 5250 application stays up and running. Also you may encounter things like job contention on IBM i where there are not enough job queues or threads to handle the current load of requests. Not something you can easily see without stress testing.

5 Reasons we need software testing.

3. There is always the chance that the end user actually *will* do “that”.

   Just because you think they might not, does not mean that they won’t. Believe me they will. We all have enough horror stories about this.

5 Reasons we need software testing.

4. There are TONS of different devices, Operating Systems, and web browsers.
   An app written for one web browser will not necessary work correctly on another. A Java application written for one Java VM might not work correctly on another.

5 Reasons we need software testing.

5. We owe our user community the best quality software that we can possibly give them!
   We need to take pride in our work. Would you want to use junk? Of course not and neither do your customers.
   Producing quality software enhances pride in your work and delivers a better image to your user community.

OK OK you've won me over, but why automated testing?

CM FIRST Rethink Modernization

AXON iVY digitalize your business

OpenLegacy

ca technologies

Automated software testing saves time and money.

Software tests have to be repeated often during development cycles to ensure quality. Every time source code is modified software tests should be repeated. For each release of the software it may be tested on all supported operating systems and hardware configurations. Manually repeating these tests is costly and time consuming. Once created, automated tests can be run over and over again at no additional cost and they are much faster than manual tests. Automated software testing can reduce the time to run repetitive tests from days to hours. **A time savings that translates directly into cost savings.**

OK OK you've won me over, but why automated testing?

Improves Accuracy

Even the most conscientious tester will make mistakes during monotonous manual testing. Automated tests perform the same steps precisely every time they are executed and never forget to record detailed results.

OK OK you've won me over, but why automated testing?

Increase Test Coverage

Automated software testing can increase the depth and scope of tests to help improve software quality. Lengthy tests that are often avoided during manual testing can be run unattended. They can even be run on multiple computers with different configurations. Automated software testing can look inside an application and see memory contents, data tables, file contents, and internal program states to determine if the product is behaving as expected. Automated software tests can easily execute thousands of different complex test cases during every test run providing coverage that is impossible with manual tests. Testers freed from repetitive manual tests have more time to create new automated software tests and deal with complex features.

OK OK you've won me over, but why automated testing?

Automation Does What Manual Testing Cannot

Even the largest software departments cannot perform a controlled web application test with thousands of users. Automated testing can simulate tens, hundreds or thousands of virtual users interacting with network or web software and applications.

OK OK you've won me over, but why automated testing?

Automated QA Testing Helps Developers and Testers

Shared automated tests can be used by developers to catch problems quickly before sending to QA. Tests can run automatically whenever source code changes are checked in and notify the team or the developer if they fail. Features like these save developers time and increase their confidence.

OK OK you've won me over, but why automated testing?

Team Morale Improves

This is hard to measure but automated software testing can improve team morale. Automating repetitive tasks with automated software testing gives your team time to spend on more challenging and rewarding projects. Team members improve their skill sets and confidence and, in turn, pass those gains on to their organization.

Choosing an Automated Test Tool

Support for Various Applications and Platforms

Some automated testing tools support only one type of application, for instance, only Java or .NET applications, while other tools support more application types. In general, the best option is to choose a tool that supports all development tools currently used in your organization, or all tools that you plan on using in the near future. Even if you build your application with one compiler, you might use other compilers in the future.
Selecting one "multi-compiler" software testing tool instead of several "compiler-specific" tools may save you money, but you must decide which is more important and whether it will save you time and energy when creating tests for future application versions. Purchasing and using several software testing tools requires additional time for training and creating tests. Do you have that much time? Does your company have enough budgeted to pay for this?

Choosing an Automated Test Tool

Support for Various Operating Systems

Choose an automated testing tool that supports as many operating systems as possible. Make sure that the tool supports recent operating system versions. The fact that your customers do not run your application under Windows 8 does not mean that they will not use it in the future. If your application works under this operating system, your software testing tool should be able to work under it as well. In other words, the more Windows versions the software testing tool supports, the better.

Choosing an Automated Test Tool

Support for Mobile Devices

If you need to test mobile applications, check if the automated testing tool provides support for them and if it supports the test operations you need. You may need to know the answer to the following questions: Can it simulate swipes and long touches? Can it simulate touching physical buttons? Does it support gestures (multi-touch events)? Can it install applications on a device? Can it access sensor data of a device? Do you need to prepare applications for testing or can the tool work with unprepared applications? Does the tool require root (superuser) permissions? Does it support emulators?

It is also important to check whether the tool can access an application's internal objects and controls. If it can, you will be able to create more flexible and accurate tests. Using object-based testing, you can, for example, change the text of a text box by using one command - there is no need to input text letter by letter.

Choosing an Automated Test Tool

Support for Various Software Testing Types

When choosing an automated testing tool, check whether it supports the test types that you are going to run. Can it simulate user actions over the tested application's user interface, in other words, perform functional testing? Can it run unit tests built into the tested application? Can it run tests created with NUnit, JUnit, DUnit or MSTest? Does it support distributed testing? Does it have special features for synchronizing tests that run simultaneously on several network workstations?

Also, if you test your products using both manual and automated methods, you may want to choose a tool that provides specific support for manual tests. Also, don't forget other important factors, such as, data-driven testing support and the ability to use the tool for regression testing.

There are many automated software testing tools that support only one or two testing types, for instance, only unit testing or functional testing. Some vendors require separate products for different testing types. For instance, support for distributed testing can be implemented with separate products or add-ons. If you wish to support these testing types, you will have to add the price of these products or add-ons to the price that you are going to pay for the product.

Choosing an Automated Test Tool

Creating Automated Tests Without Programming

An automated testing tool must let you create tests quickly and effectively. The test creation techniques must be sufficiently powerful to allow performing various automated testing tasks. At the same time the automated testing tool must be easy-to-use and clear even for inexperienced QA personnel. A good automated testing tool may support several ways of creating automated tests. For instance, it may provide a simple, easy-to-use visual automated test editor and some other features, like automated test scripts, for performing advanced tasks.

Many automated testing tools provide only scripting capabilities for creating automated tests. Creating automated test scripts requires programming experience and may be difficult for inexperienced testers. TestComplete provides a simple but powerful alternative to automated test scripts: keyword tests (also called keyword-driven testing).

Choosing an Automated Test Tool

Automated Test Scripting

Many automated software testing tools also provide scripting capabilities. Scripts are very important when performing automated testing. They let users create automated tests for specific application features, provide a variety of automated testing actions and perform operations that cannot be performed with standard features built into the automated testing tool. Automated test scripts are useful even if the automated testing tool provides easier ways to create automated tests, because script code typically provides more powerful functionality, it lets you perform actions that other automated testing types cannot.

If you prefer scripting for your automated testing, it is important to pay attention to the scripting language the automated testing tool uses. Using standard languages, such as VBScript or JScript rather than their dialects, will decrease the time needed to get acquainted with a product. Another benefit is that you can find information on these languages from third-party web sites as well as MSDN.

Choosing an Automated Test Tool

Recording Automated Software Tests

Functional (or user interface) testing implies simulating user actions on the application under test. Implementing these tests even for a small application is a tiresome process. When choosing an automated testing tool, make sure the tool can record and play back user actions on the tested application. This feature will significantly streamline test development.

Choosing an Automated Test Tool

Creating Cross-Browser Web Tests

One of the most important goals in testing web applications is to verify that the tested web application behaves correctly in various browsers. Creating a test that will work properly for different browsers is a challenging task, as the object hierarchy, element properties, user interface and browser behavior differ from one browser to another and even from one version to another version of the same browser. QA engineers have to keep in mind browser differences and use tons of conditional statements to make tests function properly. Sometimes QA teams even create separate tests for different browsers. All this makes tests complex and difficult to maintain.

Creating Cross-Platform Mobile Web Tests

Another important goal in testing web applications is to verify that the tested web application works correctly on various mobile devices. These devices have different screen sizes and orientations. Some web sites may display different content, which depends on the device you use. It is impossible to have all the devices your application will run on. That is why, it is important to be able to test applications by using a mobile browser emulator.

Creating Automated Tests That Are Resistant to Changes

Normally, you start testing your application while it is still in development. So, it is quite possible that after you make changes to the application, the current tests will become obsolete and you'll have to update them to work with the changed application windows and controls. Modifying the tests may take the same amount of time as modifying the application. That's why smart teams like to create flexible tests that will be resistant to changes made to the application under test.

Creating Automated Tests That Are Resistant to Changes

Normally, you start testing your application while it is still in development. So, it is quite possible that after you make changes to the application, the current tests will become obsolete and you'll have to update them to work with the changed application windows and controls. Modifying the tests may take the same amount of time as modifying the application. That's why smart teams like to create flexible tests that will be resistant to changes made to the application under test.

Choosing an Automated Test Tool

Supports Various Test Data Sources

When performing functional testing, it is a common practice to simulate user input into the tested application's data forms. Thorough testing implies that you feed varied data to the input fields and that the tested application processes it correctly. When choosing an automated testing tool, check which data formats it can use, such as text files, XML files, database tables, and others.

Choosing an Automated Test Tool

Running Automated Software Tests and Synchronization

The state of the application under test and the execution environment may differ from the state and conditions you expected. For instance, some network resource may be unavailable, or the desired window is not created; the web page loads very slowly or an unexpected modal dialog box appears. These and other similar factors may cause the automated software testing tool to decide that the test has failed and it should be stopped. A good automated testing tool must be able to handle these situations in the appropriate manner. For instance, it must be able to "wait" for the needed windows, "close" unexpected dialogs automatically and perform other actions that a human would do to continue the test.

Logging Automated Test Results

An automated software testing tool should keep a log of all testing actions and provide a detailed report for them. The log of a high-end software testing tool must contain different types of messages: errors, warnings and informative messages. In this case, you can easily determine what went wrong and why the test failed.

Exporting Reports

An automated testing tool should give users the opportunity to export test results to external files. For instance, it may be useful, when you need to view test results on other computers that do not have this testing tool installed. Also, you should check whether the testing tool can export the test log so that you can send the results to your boss or colleagues.

Decide What Test Cases to Automate

It is impossible to automate all testing, so it is important to determine what test cases should be automated first.
The benefit of automated testing is linked to how many times a given test can be repeated. Tests that are only performed a few times are better left for manual testing. Good test cases for automation are ones that are run frequently and require large amounts of data to perform the same action.

Test Early and Test Often

To get the most out of your automated testing, testing should be started as early as possible and ran as often as needed. The earlier testers get involved in the life cycle of the project the better, and the more you test, the more bugs you find. Automated unit testing can be implemented on day one and then you can gradually build your automated test suite. Bugs detected early are a lot cheaper to fix than those discovered later in production or deployment.

Divide Your Automated Testing Efforts

Usually, the creation of different tests is based on the QA engineers' skill levels. It is important to identify the level of experience and skills for each of your team members and divide your automated testing efforts accordingly. For instance, writing automated test scripts requires expert knowledge of scripting languages. Thus, in order to perform these tasks, you should have QA engineers that know the script language provided by the automated testing tool.

Some team members may not be versed in writing automated test scripts. These QA engineers may be better at writing test cases. It is better when an automated testing tool has a way to create automated tests that do not require an in-depth knowledge of scripting languages, like TestComplete's keyword tests feature. A keyword test (also known as keyword-driven testing) is a simple series of keywords with a specified action. With keyword tests, you can simulate keystrokes, click buttons, select menu items, call object methods and properties, and do a lot more. Keyword tests are often seen as an alternative to automated test scripts. Unlike scripts, they can be easily used by technical and non-technical users and allow users of all levels to create robust and powerful automated tests.

You should also collaborate on your automated testing project with other QA engineers in your department. Testing performed by a team is more effective for finding defects and the right automated testing tool allows you to share your projects with several testers.

Create Good, Quality Test Data

Good test data is extremely useful for data-driven testing. The data that should be entered into input fields during an automated test is usually stored in an external file. This data might be read from a database or any other data source like text or XML files, Excel sheets, and database tables. A good automated testing tool actually understands the contents of the data files and iterates over the contents in the automated test. Using external data makes your automated tests reusable and easier to maintain. To add different testing scenarios, the data files can be easily extended with new data without needing to edit the actual automated test.

Create Automated Tests That Are Resistant to Changes in the UI

Automated tests created with scripts or keyword tests are dependent on the application under test. The user interface of the application may change between builds, especially in the early stages. These changes may affect the test results, or your automated tests may no longer work with future versions of the application. The problem is automated testing tools use a series of properties to identify and locate an object. Sometimes a testing tool relies on location coordinates to find the object. For instance, if the control caption or its location has changed, the automated test will no longer be able to find the object when it runs and will fail. To run the automated test successfully, you may need to replace old names with new ones in the entire project, before running the test against the new version of the application. However, if you provide unique names for your controls, it makes your automated tests resistant to these UI changes and ensures that your automated tests work without having to make changes to the test itself. This also eliminates the automated testing tool from relying on location coordinates to find the control, which is less stable and breaks easily.

CM FIRST
Rethink Modernization

AXON iVY
digitalize your business

OpenLegacy

ca technologies

Questions? Comments?