**Question 1:** How must the regression testing process change under the agile development methodology?

**Response:** Many organizations are adopting agile development practices, but the same organizations are not modifying their testing strategy to match the development methodology. Agile development provides a focus on smaller units of the application in minidevelopment cycles. Code and test cycles are performed, but regression testing requires predefined test suites that are often overlooked in the rush to implement agile coding methods. Agile regression testing fits well with code and test cycles if test suites for the units of the application are planned and written in advance of the coding. Regression test suites can then be quickly exercised repeatedly during the unit coding process. Code can be quickly modified and retested until the unit meets the functional requirements.

One good approach to regression testing with agile development is record and playback. Functional tests can be quickly performed after code changes to a specific area of an application, and results can be visually monitored or recorded to a log for later manual or automated review. Data-centric applications can be regression tested, and the resulting data can be automatically checked for accuracy after changes to the code are made.

Agile coding can also benefit from automated testing if the coding process includes writing test interfaces to the units of code. Standard methods of unit testing can be quickly automated with a few lines of code, and nothing in the agile approach prohibits this small, additional effort. The benefits of supporting automated unit testing are large.

**Question 2:** Are any new tests ever introduced during regression testing?

**Response:** Regression testing is commonly viewed as the repetition of previous tests to ensure that changes to code to repair defects has not introduced new defects. There are many situations where existing tests are not sufficient during the regression testing process. Repair of a defect might require substantial rewrites to some areas of the code. These changes might render existing tests ineffective if a new approach to the functionality is adopted. In this case, new regression tests may be required to appropriately test the new code.

Another situation requiring new regression tests is the introduction of new functionality. Regression testing is commonly used with new versions of
FAQ: Regression Testing

software to ensure existing functionality has not been broken by new code introduced in other areas of the application; however, the new functionality most certainly requires its own testing. New test suites would be prepared in this case, and regression tests based on these new tests would later be used for regression testing during the test and repair cycles.

A third situation requiring new regression tests might occur if existing tests are determined to be inadequate for application coverage. Finally, new regression tests may be designed to monitor the operation of software in production environments as new factors regarding performance or other critical operations are examined.

**Question 3:** What help can automated testing tools provide for regression testing?

**Response:** Automated testing has a cost. To perform automated testing, additional code is often necessary, and additional time is required to add this code during the implementation phase. Automated testing also often requires the preparation of test scripts and testing of the actual test routines. If the regression tests are repeated often, the benefits of using automated test tools usually outweigh the costs. For small projects, the cost of implementing automated testing is usually not worth the benefits.

There are some types of automated testing that do not require as much effort but still provide some benefits. Record and playback regression tests can be prepared relatively quickly and are useful for basic tests of functionality. Automated tests can also be prepared to check database files, and load testing can also be automated easily in some cases by feeding an application with user interface (UI) responses or data feeds.

Interpretation of results also commonly includes some level of manual interpretation, so the benefits of automated program operation to test outputs may produce more data than can be effectively analyzed. This overproduction situation is common with performance profiling tests; however, data-driven testing can often be focused on key areas of functionality to make the effort worthwhile.
Question 4: What important quality metrics should be collected during regression testing?

Response: An important concept to keep in mind is that regression testing is a repeat of test suites used during the system testing process. As such, regression tests generate the same metrics as the system tests; however, there are some key metrics to measure as a result of the continued regression tests that can help determine the quality condition of the application.

One primary metric is the number of defects discovered during each cycle. Measuring the discovery rate can help determine when the application is reaching defined quality goals. Another important measurement is the number of defects introduced as a factor of the quantity of code touched or modified. This metric helps determine the quality of code produced during test and repair cycles and also helps predict the number of defects likely to be introduced during each repair cycle.

Unit and application susceptibility can be quantified by tracking and analyzing the number of defects introduced in other areas of the application when one unit is modified. The level of interaction between different areas of the program addresses the level of independent design used in the application. Higher levels of cross-unit susceptibility suggest a higher risk of making changes anywhere in the code. A poorly designed program will be more likely to have undiscovered defects after changes have been made, and this type of frail program should be tested more thoroughly to reduce quality risks.

Question 5: Is regression testing the same thing as system testing?

Response: Regression testing serves to validate parts of an application that have been modified. To serve this role, regression testing commonly consists of repetition of certain test suites used during the system testing process. The difference between regression testing and system testing is that regression testing uses repeatable tests whereas system testing can utilize test suites that are not necessarily designed to be repeated. An important factor in regression testing is that tests are repeatable so the effect of changes in the code can be measured.

Regression tests may also be more focused than system tests. System tests will often test general functionality of an application. Regression testing can focus on specific units in a program, or it may also strictly measure key
application outputs. Specific inputs for units or functions in an application can be provided through unit regression tests, and the expected outputs can be measured. As code is developed, more regression tests are added so when the application is complete, a thorough set of regression tests are available. System tests are typically based on the requirements specification and prepared during the implementation phase but independent of coding.

Regression tests provide broader coverage than system tests. Code execution is one of the key focus areas of regression testing rather than functionality of the complete application. Regression testing also utilizes automated testing techniques that may require code in the application, but system testing is more outside the code and does not require code modifications.

**Question 6:** How can I determine which tests to perform during regression testing?

**Response:** To be completely safe, a test-all strategy could be employed in which all functionality and code is tested during a regression test cycle; however, even if all the regression tests are automated, the complete suite of regression tests does not usually make good sense. Large applications may require too much time to execute a complete set of test suites each time a modification is made. Test selection is an important consideration to reduce the time required for regression test cycles.

An appropriate test selection methodology starts with a good understanding of test coverage. If the regression tests are mapped to the code appropriately, changes to the code can be quickly tested by selecting the test suites that map to the modified area of code. Defects uncovered in other areas of the application would not be covered by this selection technique. For this reason, interrelationships in the code must also be understood. A useful mapping of test suites against code should include a reference to related areas. These references would help the testers determine which additional regression tests should be executed in addition to those used to directly test a specific area of code. Sometimes, these interrelationships become so complex that there is little choice than to execute a full suite of regression tests.

Data and process flow diagrams can also assist in the regression test selection process. These diagrams help the test team understand the relationships in the application. Following the flow of these diagrams will help trace changes in one area of the application to other areas. Regression tests
can be selected to cover the path through the diagram and reduce the risk of undiscovered defects.

**Question 7:** Must each cycle of regression testing cover the entire application?

**Response:** A complete cycle of regression testing can be time-consuming and typically is not required. The complete regression test cycle is called the *safe method* because all tests are executed. For small applications, this approach will usually be appropriate, but for large applications, the safe approach is usually too time-consuming. Some situations do require complete regression tests.

New versions of an application might introduce significant new functionality, and most areas of the code may be touched during development. In this case, there is little choice but to execute the complete suite of tests. This situation generally introduces new tests to the regression test suite and not in the typical regression test cycle. New versions of applications usually undergo unit testing during development so by the time system testing is complete, only selected regression tests must be executed during test and fix cycles.

Some applications are so critical to the operation of a business that they must undergo a complete set of regression tests any time a change is made. Financial operations or software that might put human safety at risk would fall under this category. These risk factors must be taken into account during the regression test-selection process.

**Question 8:** What risks are involved when regression tests are performed?

**Response:** Regression testing serves a vital role in ensuring the quality of an application. When changes are made to the code, either to repair previously discovered defects or to add new functionality, defects are likely to be introduced. Regression testing can help uncover these defects and avoid the introduction of defects into the production build of the program. There are some potential risks involved in regression testing too.

Regression testing can generate a false sense of assurance that all defects have been uncovered. Automated testing in particular gives the impression that an application is being thoroughly tested because of the large amount of output that can be generated by these tests. The organization and speed of automated tests leads testers to believe that coverage is deep and thorough,
but in fact, the tests are only as good as the test design, which may be flawed or incomplete.

Regression testing also relies heavily on proper test selection. The human part of the test-selection process requires an understanding of test coverage and interrelationships in the code. Any errors in the test-selection process could result in missed coverage and missed defects. Changes in the code might also render a regression test ineffective because of new code execution paths. The test process will not provide feedback about missed coverage and may still appear to be a good test.

**Question 9:** How sophisticated and complex are regression tests?

**Response:** Regression testing is often neglected because of a perception that this form of testing is complex. Automated testing can be a complex process of writing test scripts and modifying code to support the tests. Analysis of some regression test results can also be a complicated and time-consuming process that some test teams do not find worth the effort. Some effective regression tests can also be simple.

Regression testing is repetition of previously executed test suites. These tests can range from very simple, manual unit tests to fully automated, complete code execution path coverage. Most regression test suites have some very simple tests as their foundation. Inputs are supplied to some unit of code, and outputs are measured and examined for correctness. Consider a simple unit of code that takes one input and is expected to produce a clearly measurable output. A programmer can perform this regression test in a matter of minutes using a debugger. Minimum, maximum, and out-of-range values can be defined and tested with the suite. In this case, the entire test case is simple and easy to execute.

In summary, regression tests can cover the entire range from simple to complex, depending on the nature of the code to be tested. Many useful, simple regression tests can produce substantial coverage with minimal effort.

**Question 10:** When is regression testing necessary?

**Response:** There are many situations when regression testing is appropriate. The first, and most common, is when a specific area of the code has been modified in an effort to repair one or more defects. In this case, a set of regression tests should be selected that cover the code that was modified.
These tests would validate the correction and also test to ensure additional defects were not introduced. In addition, coverage should include other areas of the application that might be affected. This use of regression testing is typical during the test and fix cycles of software development.

Another situation requiring regression testing is when new functionality is added to an application. The new functionality will certainly undergo regression testing during its own test and fix cycles, but other parts of the application should also undergo regression testing to detect new defects caused by the new functionality, even though these other parts of the application were not changed.

System monitoring is another good application of regression testing. Behavioral metrics of an application are often an added bonus with many regression test suites. These metrics can be used to determine if environmental conditions not present during testing have introduced problems in the system. Complex systems often rely on various component availabilities as services to the application. Regression tests can identify broken communications between the interrelated parts of a system and can include notifications to those who must service the system.