Software Assurance

Agile Testing
Key Service Name: Agile Testing

Service Description:
The Agile Testing offering enables clients to achieve improved coordination of their test resources with the agile development team by allowing automated tests to be developed in tandem with code development on the same set of requirements.

The methodology employed for the Agile Testing offering eschews the concept of building automated tests after code has been released, unit tested, and manually tested. Instead, the test team creates automated tests in parallel with the code development team for the same set of requirements. The test and development teams perform integration between automation tests and code continuously throughout the iteration cycle. Because all defects are found throughout the iteration cycle, all tests pass in the final test run at the end of the iteration, and the application code moves to the next iteration cycle with zero defects.

As with unit tests, the Agile Testing methodology allows the development team to experiment with design changes to previously implemented functionality, to refactor critical components, knowing that there is a solid automated test suite that will alert them of any unforeseen, possible business critical, side effects to the rework effort. However, unlike unit tests, which typically focus on discrete components of the code base, the Agile Testing methodology can be utilized to test at various perspectives, or interface points, of the application:

- End user perspective
  - by using the Agile Testing framework in coordination with test tools designed to allow API control of an application via the user interface
- Database perspective
  - by creating specific SQL calls to exercise database procedures and functions independently of the application
- Custom service perspective
  - by leveraging proven web service test tools, or creating custom tools as necessary for testing services with non-standard protocols

This approach has been proven successful on Web 2.0 projects, where other commercial automated test solutions (e.g., HP/Mercury WinRunner) fall short, by using test tools such as Selenium which are designed to test Web 2.0 applications. The framework can easily call any API based test tool.

Another advantage over typical unit tests gained by using the Agile Testing framework is a two-fold increase in parallel work effort. Instead of requiring the automation tester to write standard code such as C# or Java, the Agile Testing framework allows test staff to split the automation test tasks between test commands and test scripts. Test commands are support code that interfaces directly with the application, either via user interface, database, or other interface point. Test commands are typically authored by automation test staff familiar with software engineering such as Java. Test scripts, on the other hand, are written by test personnel (and in many cases, business personnel) in a high-level, business-action structure based on XML. This lets the tester to focus on business actions (e.g. AddCustomer, ModifyOrder, etc.) instead of low-level user interface actions (e.g. PressButton, TextField, etc.). Using this high-level scripting approach allow the creation of more tests with less scripts and which allows domain-knowledgeable business users to assist in the test creation process.
By using the Agile Testing framework, three parallel efforts of work occur during an iteration:

- Application code development  
  - The actual building of the application by the development team
- Test script development  
  - The creation of high-level test scripts focused on business functionality
- Test command development  
  - The building of commands used by the high-level test scripts

This is in contrast to the typical development/test team relationship, where the development team completes an iteration, then the build for that iteration is released to the QA team to test. For each iteration, QA is testing code developed for iteration N while the development team creates code for iteration N+1. Automation testing usually develops independently of the iteration cycle as a best-effort to stem the flow of manual tests.

Although this method offers efficiencies due to the pipelining effect, the development team only discovers defects after the code has been developed. Using the Agile Testing methodology, defects are discovered as the code is being developed; this approach minimizes rework for the development team.

Figure 1 depicts the typical development/test cycle, where test efforts are always an iteration behind development. Figure 2 depicts the Agile Testing iteration cycle; note that the development and QA efforts, which include automation, are developed in lockstep.
Problem that the Service addresses:

- **Project delays:** In situations where project delays are primarily due to regression testing not completed on time and/or testing delayed due to labor-intensive tasks, test automation could be leveraged to reduce test cycle time. Agile Testing focuses on test automation from the beginning, not as an afterthought.

- **Defects cause by rework:** In situations where project rework causes defects that go undetected because of insufficient testing. Agile Testing follows the agile development credo of developing a “safety net” of tests that allow the developers to rework and refactor code with confidence.

- **High rate of defects:** Using the Agile Testing methodology produces code each iteration with zero defects, because the code and tests are integrated throughout the iteration cycle. Although this ideal may not be met for every iteration, the result is a much lower number of defects per iteration than with other methodologies.

- **All issues solved with Test Automation:** Cigital’s Agile Testing service is a subset of the Test Automation service, and therefore encompasses all of the benefits of that service.

Service Tasks:

1. **Test Strategy Definition:** Test Strategy definition is geared towards aligning the Agile Testing methodology with the overall QA goals of the project. This includes the following activities:
   - Test Planning
     - A detailed plan will be developed that would describe the deliverables of the Agile Testing methodology; use of the Agile Testing framework software; integration with the agile development environment; system integration points for which tests will be written; test execution schedules; test results reporting schedules; change management and issue resolution mechanisms.
• Scope definition
  o Identification of requirements for the Agile Testing framework
  o Selection of test metrics to gather from test runs as well as target metrics (e.g. 99% statement coverage, release to production only when 100% tests pass)
  o Definition of test selection criteria for automation
  o Definition of test automation process workflows
• Test tools selection
  o Identification of test tools to be used with the Agile Testing framework; decisions concerning using existing tools; purchase new tools; build custom tools will be made and documented
  o Identification of complementary/enabling tools and techniques (e.g. code coverage, requirements traceability)
• Test Environment and Resource requirements definition
  o Definition of test environments and dependencies on other environments
  o Identification of resource requirements
  o Definitions of integration points with build processes and external interfaces.

2. Reporting: Test results will be collated from the output of the test script run within the Agile Testing framework. A test report will be generated and published. All problems found will be reported and recorded as per the QA processes requirements.

Deliverables:

1. Test Strategy and Architecture: A document that identifies all necessary components of the Agile Testing architecture including, test strategy; installation of Agile Testing framework software; input and output data formats; dependencies like APIs, data, tools and environments.

2. Test Reports: All test reports will be delivered to the client when they are generated as part of automation runs. The test report would include the types of tests executed and their results; descriptions of problems found during the runs; and a trailing history of problems found and resolved between runs.

3. User’s Guide: A document describing how to setup and configure the Agile Testing framework; prepare test scripts; how to execute test scripts; generate test reports from test results; how to interpret test results; and how to troubleshoot problems during test execution. The manual would also contain information on current capabilities of Agile Testing framework and desired future enhancements. A section would be dedicated to discuss the development details of test harness to aid in maintenance of the test harness by another team.