Software Assurance

Test Automation
Key Service Name: Test Automation

Service Description:
The Test Automation offering enables clients to achieve improved productivity of their test resources, and to reduce the length of regression testing cycles while increasing test coverage. Test automation complements and vastly improves the efficacy of existing manual testing and integrates with the overall testing effort. Instead of executing basic system tests time and time again, test resources can instead concentrate on: test case design, execution of test cases via the use of automated tests, as well as execution of remaining manual tests. These are tasks that best utilize a tester’s domain expertise and knowledge of test methodologies and practices. The Cigital offering for Test Automation covers all levels of test at the various phases of the SDLC, from unit level to sub-system and system level.

We offer several types of Test Automation: GUI Automation, API level automation, and Data Quality/Integrity Automation. Automated GUI tests exercise an application through its user interface, API level tests directly call into the application business logic, Data Quality/Integrity testing calls directly into an application’s back end, which is usually a database. API level test automation and Data Quality/Integrity Testing address testing specific areas that are not accessible by black box testing. To develop such solutions, we use grey box techniques that utilize deeper knowledge of code and architecture. Often, these types of tests do not take as much time to execute as GUI tests and can be run earlier in the lifecycle.

Our unique methodologies maximize the use of available automated testing tools, but are not tied down to a specific product, script, or language. We can create automation frameworks and suites of tests using tools clients already have in hand, such as commercial GUI test automation tools like: Mercury WinRunner, Segue Silk, Empirix e-test etc. However our solution does not have to rely on expensive commercial test tools that clients must purchase. We can utilize open source tools to create custom tailored test automation solutions. To that end, for GUI Automation we have expertise with HttpUnit, Badboy and Wave Manager, etc.

For API level automation we have customized the open source X-Unit family of test frameworks like Junit, Nunit, VBUnit, CppUnit etc, and utilize the following programming languages: C/C++, VB 6, VB.NET, Java, Perl, Python, C# etc.

Database Quality/Integrity Testing leverages our knowledge of SQL and shell scripting languages along with the expertise we bring with the programming languages mentioned above.

All our automation solutions are designed to be maintainable, adaptable and reusable across software releases. Training on our solutions is included to ensure efficient technology transfer.

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.

- **Cost overruns / large percentage of rework occurring:** In situations where project undergoes a large percentage of rework and a large percentage of the testing time is spent on validating existing functionality, test automation could be designed to reduce the time to regression test.
In an environment of iterative builds, test automation could be used to quickly assess the quality of a build and perform regression testing, thus providing more time to test new functionality.

- **Insufficient test coverage:** In some situations the testing team may not have sufficient resources to obtain required test coverage. Test automation enables increased coverage by testing different data variations or test scenarios not covered thru manual testing.
- **Insufficient level of QA / testing too late in the SDLC:** Cigital Test Automation also means automating at the right level, not just the GUI level. API level testing can leverage more in-depth testing than just GUI testing alone.
- **High reject rate of defects during test, low quality of builds that go to test:** Not doing enough unit level testing may result in low quality of deliverables to test. Automating unit test enables development to achieve higher test coverage, via repeatable and reproducible methods.
- **Highly complex and/or configurable application that takes too long to test:** Test Automation is most effective for complex applications with a high overhead of manually executing tests. Test Automation can enable testing of more scenarios and permutations of data without adding extra time to test cycles.

**Service Tasks:**
The foremost task is to understand the project QA goals in general and the test automation goals in particular. These goals drive the tasks that are described below.

1. **Test Strategy Definition:** Test Strategy definition is geared towards aligning test automation goals with the overall QA goals and identifies how test automation will fit into the test SDLC and overall QA strategy. This includes the following activities:
   - **Test Planning**
     - A detailed test automation plan will be developed that would describe the milestones and deliverables of the test architecture; test harness development; test execution schedules; test results reporting schedules; change management and issue resolution mechanisms.
   - **Scope definition**
     - Identification of requirements for the test automation framework
     - 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)
     - Definition of test selection criteria for automation
     - Definition of test automation process workflows
   - **Test tools selection**
     - Identification of automation test tool: decisions concerning using existing tools; purchase new tools; build custom tools will be made and documented
     - Identification of complementary/enabling tools and techniques (e.g. code coverage, requirements traceability)
   - **Test Environment and Resource requirements definition**
     - Definition of test environments and dependencies on other environments
     - Identification of resource requirements
     - Definitions of integration points with build processes and external interfaces.
2. **Definition of Automated Test Architecture:** Test automation architecture depends on the underlying application architecture and the type of automation selected. For example, automation targeted at system level (which may require GUI automation) requires an in-depth understanding of the application usage, API level automation requires a deeper understanding of the interfaces of different components and the development environment. Data Integrity/Quality testing requires understanding the flow of data through a particular application.

In addition, test automation architecture identifies all the necessary components of the test automation framework including high-level designs, input data formats, dependencies, environments, and usage profiles.

3. **Development of test harness:** Once the test architecture is developed and a specific product or programming language is identified, test harness development will start. Test harness is comprised of test drivers that interact with the application and supporting test utilities that help in setting up test harness for execution and interpretation of output and test results.

Examples of test utilities include scripts to prepare input test data; output file comparators; log file checkers; test report generators etc. The development effort will follow an iterative development process and adhere to the same standards as any application development.

4. **Identification of Test Cases to be automated:** Not all test cases are suitable for automation. To maximize the use of automated testing, we select test cases from the existing test repository based on predefined selection criteria and design test cases as needed. All selected test cases will be reformatted as per the framework input data specifications. For example, in a data-driven test harness, test cases are encapsulated in Excel spreadsheets, which are read and executed by the automation scripts.

5. **Execution and Reporting:** Automated tests will be executed as per the schedule outlined in the automation test plan. Test results will be collated from the output of the automated tests and a test report will be generated and published. All problems found by the automated tests 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 test automation framework including, test automation strategy; high-level framework design; input and output data formats; dependencies like APIs, data, tools and environments.

2. **TestWare or test harness:** Includes source code for test framework; source code for test suites; test utilities; configuration files; and test data, necessary to execute automated tests. In the case where open source tools were used to develop test framework, the source of the test tool would be included in the deliverable.

3. **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.

4. **User’s Guide:** A document describing how to setup and configure test harness; prepare test input files; how to execute test harness; 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 test harness 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.

**Value Proposition:**
- Reduce test cycle time
- Increase coverage
- Improve test accuracy
- Accelerate time to market
- Provide Repeatability in testing
- Improve quality of software releases
- Expedite manual testing (in test data setup)

---

**About Cigital**
For over a decade Cigital has enabled some of the most well-known companies in financial services, communications, insurance, hospitality and e-commerce to reduce their mission-critical software business risks. Cigital consultants help companies protect some of their most valuable assets: company information, customer data, shareholder value and brand. Each client’s unique requirements are served through a combination of proven methodologies, tools and best practices. Cigital assures the reliable delivery and deployment of software that organizations build, buy and integrate. The company is headquartered near Washington, D.C. with offices in Boston, New York, Los Angeles and Delhi, India.

© 2008 Cigital All rights reserved. An electronic version of this document is maintained in a document management system. The reader of this printed page is responsible for ensuring its currency.