FAST WHITEPAPER

Greenliff's Automated Testing Framework Solution

Management Summary

Today IT has become a mission critical business risk. The risk can be reduced drastically using new test methods and approaches. However, heterogonous test infrastructures and highly fragmented test environments prevent the efficient use of test results within the entire organization. Greenliff's FAST solution is a light-weight, open and easy-to-deploy framework that can integrate some or all of your existing test environments into one centrally-managed infrastructure, increasing the effectiveness of your test investments and raising quality awareness across the organization.

What is testing?

A company's survival in the rapidly changing and competitive world of today depends on the soundness of its IT infrastructure. Software testing is a technical audit discipline to assure function and quality. However, if test methods and framework implementation are not considered carefully, it can easily detract from a company's profits. Test design strategies need to have precise and accurate outcomes to provide any value for the money being spent. A test framework used uniformly will successfully streamline test activities across multiple and significantly different environments. This paper introduces such an automation test framework.

Automated testing

The most basic description of automated testing states: *"Automated testing is where technology is used to replace testing workflows once done by humans."* There have been many books and articles written, tools developed and web sites dedicated to the topic of Automation of Software Testing. Some sources suggest that automation of Unit testing realizes the multiple benefits of early bug detection and resolution. Other sources say automated integration-level smoke tests decreases development time and margin for error. However, there is one type of test that is generally agreed upon, to generate the most benefits from automation. This is the Regression Test.

Regression testing

Checking functionalities that have already been tested is what regression testing is about.

Code changed in one area of the application may inadvertently affect other unchanged areas, resulting in negative side effects to the overall software functionality. Regression testing may also be required in situations where the source code has not changed, but the operating environment has. Regular regression testing, like an automated nightly test run, will reliably identify broken functionality or degraded performance.

Lastly, automated regression testing is a prerequisite for Agile development and Continuous Integration. Developers feel more confident in making modifications knowing that errors will show up in the regression tests. In this way, regression testing becomes the safety net allowing for fast software change and shorter time to market.

GREENLIFF

Data driven
Automated software regression tests that are executed repeatedly with different input data often use a data-driven test approach. Data-driven testing utilizes input data that resides outside of test scripts in a separate database. This approach is tremendously efficient when comparing maintenance efforts. Any script modifications will automatically carry over to every test case in the spreadsheet. On the flip-side, scripts that include the input data inside each test case require repeated modifications for every test case. This can be extremely time-consuming if there is a large amount of input data.

Consider a spreadsheet of possibilities and combinations of possibilities for input in a software application field. This can be used to validate the software's expected behavior when using every row of input data.

	Possible Input 1	Possible Input 2	Possible Input 3
Test Case 1	x	y	z
Test Case 2	y	z	x
Test case 3	z	x	y

A data-driven approach to testing would be a perfect way to repeat the tests with a different row of data each time.

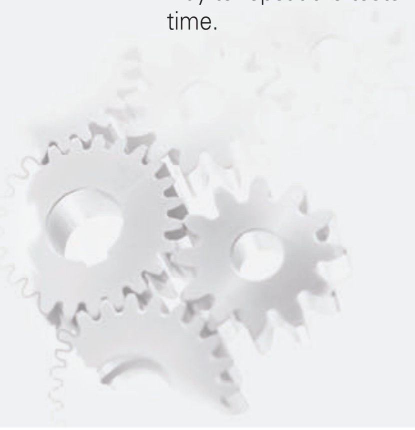
Greenliff Automated Framework Solution
The most successful approach to automated software testing is based on an automation framework and data-driven tests. FAST is Greenliff's framework for automated software testing designed for the management of automated test cases that run on many different sub-systems, integrating them into one centrally-managed infrastructure.

FAST manages the scheduling of tests and the display of results data from legacy or new test systems. Even though most test environments are extremely heterogeneous, employing various scripting languages, vendor tools, test drivers, and measurement devices, FAST is able to combine these disparate systems into one integrated solution.

FAST consists of variations on the following core components:

- **Test Engines:** automated testing sub-system
- **Job Manager:** job distribution to different test engines
- **Test Job:** job submission XML interface for initiating test execution
- **Execution Monitor:** test process monitoring
- **Collector:** test results database
- **Results Application:** Web display for results
- **Reporting Tool:** unified result reporting

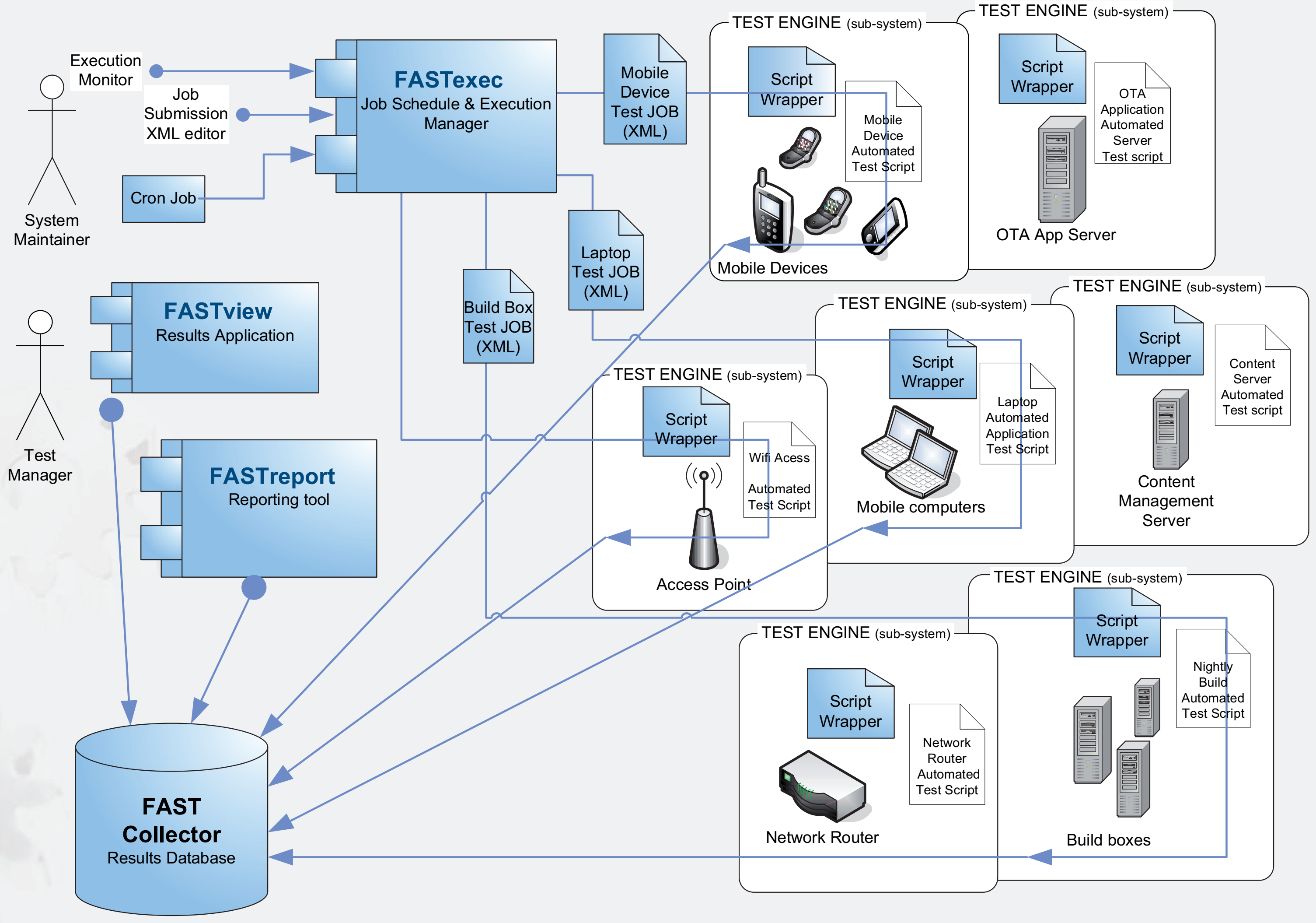
Using the component flow diagram you can identify all the FAST components in blue.



Make your
test results visible –
stimulate quality awareness

GREENLIFF

FAST Component Diagram



GREENLIFF

Test Engines
Automated sub-systems (3rd party automation applications or script files) that actually perform the automated test execution are the FAST Test Engines. The engines usually require unique environments in order to run test scripts.

Job Manager
The FAST Job Manager component knows the location of all the separate Test Engines on the network and can distribute jobs to the appropriate engine. It understands that scripts which require MS Windows with a Perl interpreter go to Engine A, scripts that require a Unix environment go to Engine B and Test Director macro tests go to Engine C. The Test Job Manager also takes care of test progress monitoring and communication with the test execution scripts.

Test Job
The FAST solution starts with a Test Job submission using either a Web interface or an automatic test scheduler. The Test Job is an XML file with test case dependencies (prerequisites that need to be fulfilled prior to running the current job), test engine name, and an executable command, typically a call to launch an automated test or script.

Execution Monitor
Test processes are monitored on the sub-system of the test engines after the Test job is submitted. It shows what tests have been completed and what tests are still in progress. When the test process finishes it receives a result code of either PASS or FAIL. If the test process does not finish for some reason, it receives a result code of ERROR.

Test Script
The script file uses a software wrapper to talk the common language, whatever that may be, of the Test Engine to start running the automated test.

Collector
The results are collected in a relational database. This is achieved by parsing the output and mapping it into a standardized schema.

Results Application
Testing results are stored by the Collector and are displayed in a Web browser through the Results Application. The Results application allows browsing of test results, drilling down on single cases, and linking to additional information such as log files, bug reports, test specifications, and screen shots.

Reporting Tool
The reporting tool provides unified reporting including charts and diagrams of the testing results.

GREENLIFF

GREENLIFF

Technoparkstrasse 1
8005 Zürich
Switzerland

Phone +41 43 20 40 800
Fax +41 43 20 40 808

www.greenliff.com

Benefits of Greenliff's Automated Framework Solution

Companies can get a jump-start at establishing test automation early in the software development process. Greenliff and FAST increase the productivity of your existing automation tools. Since the approach is interoperable and scalable, it is adaptable to almost every software development environment. Test Managers get full automation and immediately reap the rewards of results capturing and comparison with FAST. Historical data stored in the FAST database are readily available to create on-demand test reports of status and quality information for Product Managers and Project Leaders.

Integration of standard test tools like Junit, NUnit, Maven, Quality Center, Rational testing tools etc. are readily available out of the box. Less common tools and in-house developed test drivers are also easily integrated through the FAST collector plug-in architecture.

Test execution is managed, scheduled, and monitored by **FASTexec**. Test results are available immediately after test execution to all interested parties, while at the same time being stored in normalized format both for real-time result evaluation and long-term archiving.

Test results are accessible from everywhere and over the Internet. **FASTview** allows for online progress monitoring, result drill-down, historical views, graphical trend analyses, and performance evaluation. User profiles and advanced filtering guarantee the right results at the right time. **FASTview** includes alarm-based notification channels as diverse as RSS feeds, SMS notification, and email compendiums.

FASTreport makes printed results available for both standard tests reports and advanced test analyses enriched with bug reports, test specifications, surveys, metrics etc. Furthermore, current and historical results can be exported into any 3rd-party report generation tool.

Conclusion

IT departments have seen tremendous growth in software quality as a consequence of large investments in professional test processes, testing tools, and training. The challenge today is to increase the effectiveness of these testing activities in order to scale effectively across the entire IT landscape while keeping costs down.

Automated testing combined with accurate and easily accessible results reporting, is not only key to increasing the effectiveness of these testing investments but is a necessity in today's market-driven environment.

FAST takes the idea of execution and management of automated test cases to the next step. The FAST framework enables easy integration of all of your existing test environments and brings them together into one centrally-managed infrastructure.