https://doi.org/10.48009/1_iis_2012_1-10
Issues in Information Systems
Volume 13, Issue 1, pp. 1-10, 2012

EVALUATING AND IMPLEMENTING LOAD AND PERFORMANCE TESTING TOOLS TO TEST ADOBE FLEX AND OTHER RICH INTERNET APPLICATIONS: A CASE STUDY

John J. Scarpino, Robert Morris University, scarpino@rmu.edu

ABSTRACT

The research summarized in this document describes the state of the Rich Internet Applications (RIA) industry, including industry demand, job growth and RIAs' impact on the software development industry, while creating a framework for evaluating and implementing load and performance testing tools to test RIA software – Adobe Flex, in particular. This document outlines a case study of a company which demonstrated time- and cost-savings and increase in efficiency due to a load and performance tool's implementation in an environment architected and developed in Adobe Flex.

Keywords: Adobe Flex; Load Testing; Performance Testing; Rich Internet Applications; RIA; Software Testing; Software Testing Tools

INTRODUCTION

When a software development environment uses Adobe Flex as a part of its architecture it can be difficult to find a load and performance testing tool that fits the testing objectives and is compatible with Rich Internet Applications (RIA), to ensure that development contains scalability and optimization. Adobe Flex is an RIA within the Adobe suite which can be coded using Adobe Flash, or manually within the Flex complier. It may be displayed to the user through a Web browser plug-in, or via virtual machines or Javascript. Adobe touts its Flex application as "powerful," with a user-friendly structure that nearly anyone can use to develop applications for mobile devices, tablets and traditional Web browser and desktop machines [1]. RIA can also run outside of the browser, using an explicit runtime setting; for example, Adobe AIR and JavaFX require a client to be installed on the end-user's computer [2].

According to StatOwl.com in August 2011, 95.71% of Web applications supported Adobe Flash, compared to 76.37% supporting Java, and 66.23% supporting Microsoft Silverlight [3]. The demand for RIA correlates with the growing prevalence of Smartphone usage. An article published in February 2011 states that sales of such devices in the fourth quarter of 2010 were up 87% above the fourth quarter of 2009, and 2010 sales of smartphones increased by 75% overall [4]. A wide increase in availability of mobile technology, the Internet and open source platforms have cut a pathway for creativity in software development [5], which has aided the increase in popularity of RIA. Demand for RIA even shows in sectors where such innovation is unprecedented. In the financial sector, for instance, eCommerce has morphed from a simple interface that facilitates a basic transaction to an interactive presentation that offers Smartphone app downloads, money management visuals, games and more [6]. The increase in availability of Web 2.0 software has also turned the idea of a "user experience" into an imperative rather than an enhancement [7].

The high demand for and support of RIA demonstrates the need for performance and load testing of Adobe Flash and Flex as crucial for application development success, scalability and optimization.

Rich Internet Applications' Impact on the Software Development Industry

The impact of RIA on the software development industry is felt on a global level. In countries like India, where Java has been the industry standard for years, RIA such as Adobe Flex are putting on the competitive pressure. India's community of flash designers and developers are starting to use Flex instead of Java (including those who were selfproclaimed Java-only programmers). One indication of this is the growth of an online group called “Flex India,” which saw its membership increase from fewer than 100 members to more than 1,000 members in less than one year [8].

A search conducted by the researcher on Indeed.com – a Web site which monitors job-posting and job-search trends across many industries – showed that demand for Adobe Flex positions in the U.S. had risen to 185,000% in January 2011 (it topped out at approximately 250,000% only about six months earlier), while demand for Microsoft Silverlight positions had risen to 75,000% in January 2011 and has slowly increased since then. This was in stark contrast to Java job growth, which barely capped 25% in January 2011 and hasn't shown any fluctuation since January 2006 (Fig. 1).

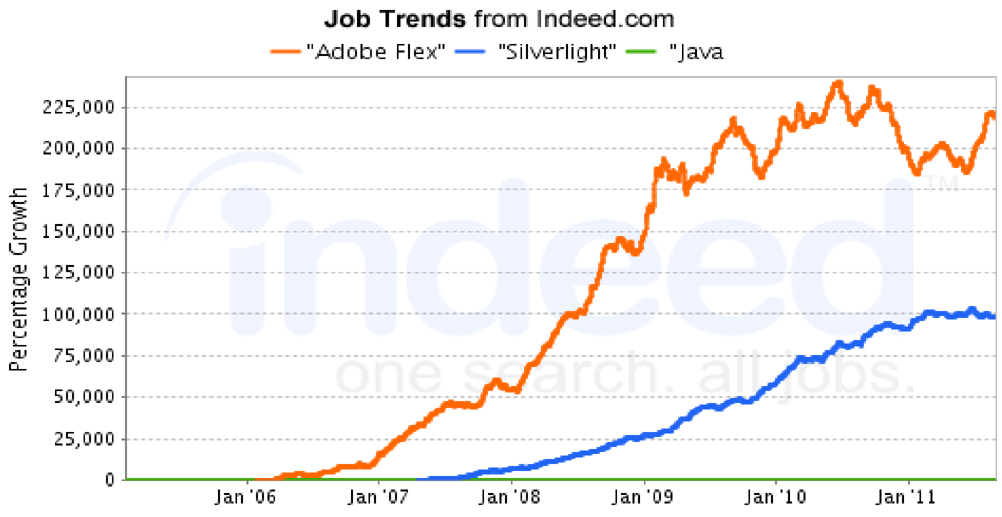

Figure 1. Industry Job Growth for Adobe Flex, Microsoft Silverlight and JAVA

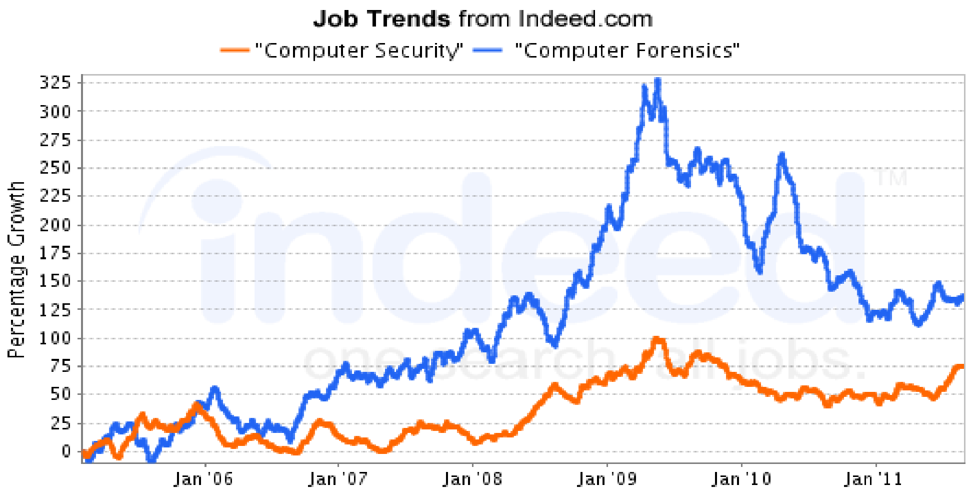

Figure 2. Industry Job Growth for Computer Security and Computer Forensics

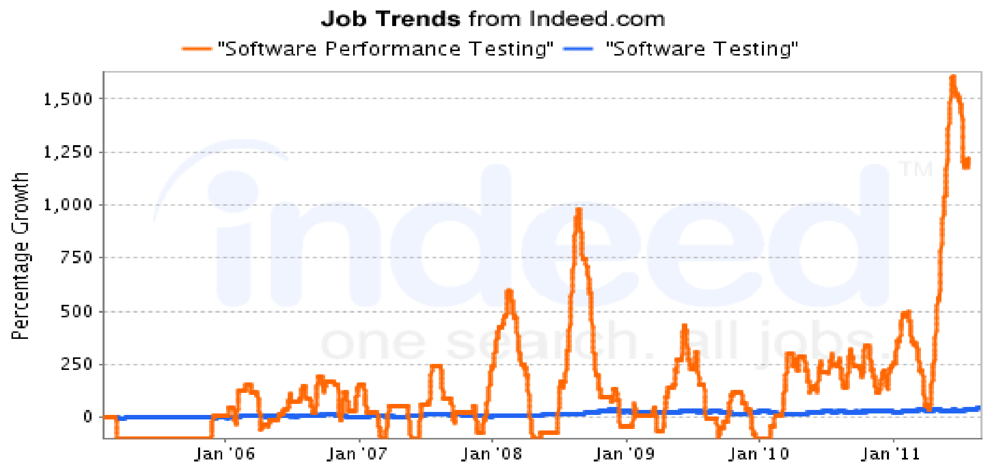

Figure 3. Industry Job Growth for Software Performance Testing and Overall Software Testing

To further demonstrate the impact of RIA on the software development industry, the researcher compared industry demand for Adobe Flex and Microsoft Silverlight to the recent boom and interest in Computer Forensics – a phenomenon that may be due to the popularity of television series such as NCIS, Cold Case, and CSI Miami [9]. According to Indeed.com, Computer Forensics job growth peaked in June 2009 at 325%, and decreased to only 125% in January 2011. The growth of Computer Security jobs was merely 50% in January 2011, and even at its highest point it was only 100%. (Fig. 2). Even with recent media hype pertaining to CSI-themed television shows, these percentages are extremely low compared to the percentages for RIA job growth; thereby demonstrating the difference between job growth due to a popular trend, and job growth due to a true industry demand for a skill.

Another search was conducted by the researcher on Indeed.com to compare the demand for RIA jobs against specific demand for software performance testers, and the overall demand for software testers. Both are "niche jobs" which support software development, and software performance testers specifically support RIA development. Job growth for software performance testers was only 500% in January 2011 and reached an enormous peak of 2,000% just six months later, but the demand for traditional software testers has remained steady at around 50% since January 2006 (Fig. 3). While the demand for these specific jobs is not particularly high (especially when compared with RIA demand), candidates who enter the software testing industry will have a prosperous future if they can demonstrate they have the skills necessary to support RIA. Data from Indeed.com indicates if demand for RIA continues to grow into the six-digit range it is likely that the demand for specific jobs that can service RIA, like Adobe Flex and Microsoft Silverlight, will follow suit.

RESEARCH METHOD

Case Study for Evaluation and Implementation of Load and Performance Testing Tools for RIA

A. Load and Performance Testing Evaluation

More and more businesses are dependent upon on Web applications to keep costs low while supporting various customer-related activities. The demand for Rich Internet Applications has put pressure on software development companies to keep their clients pleased by building applications that operate effectively in an unstable Internet environment [10]. Performance is the most crucial indicator of how well a software system or component meets its requirements for timeliness. According to Gan (2006):

> There are two important dimensions to software performance timeliness, responsiveness and scalability. Responsiveness is the ability of a system to meet its objectives for response time or throughput. The

> response time is the time required to respond to stimuli (events). The throughput of a system is the number of events processed in some interval of time. Scalability is the ability of a system to continue to meet its response time or throughput objectives as the demand for the software function increases [11].

There is no “one-size-fits-all” load and performance testing tool that is compatible with all RIA across the board. Not only do RIA engineering development methods change over time and are enhanced with new development standards, but so does how software is tested and the tools used for it. Using load and performance automated testing tools can help ensure that software load and performance tests gather key metrics and data more precisely and effectively than would be done manually by a group of people. Similarly, using a program that simulates the end user experience, or a “virtual user,” can combine test efforts and statistics to form a refined test case.

With so many Web 2.0 applications already crowding the market – and more entering it every day – there is an increasing need for consistent methods and tools to perform testing efficiently and correctly [12]. Therefore, it is important to conduct a thorough evaluation prior to purchasing a load and performance testing tool to ensure compatibility with the RIA interface.

Celerity Innovation Center, a software development and consulting company located in Pittsburgh, Pennsylvania, needed to purchase a load and performance testing tool solution that was compatible with RIA, and Adobe Flex in particular. The company provides turnkey and specialized solutions, such as: Project Management, Business Analytics, User Experience, Visual Design, Web and Mobile Development, Quality Assurance, and Support. Clients entrust Celerity Innovation Center to manage full implementation of software, from “ideation” through deployment and support. The company’s wide range of skills and offerings helps organizations on and off-site with fluidity and professionalism [13].

The researcher chose to conduct a case study at Celerity Innovation Center, studying the company’s evaluation of two products in contention for the best load and performance testing tool for RIA and Adobe Flex. The case study was conducted from May 2011 through August 2011. The researcher documented the evaluation process through which the company chose a compatible load and performance tool, including the evaluation criteria used and post evaluation results.

B. Main Evaluation Criteria

The researcher reviewed two performance and load testing tools for compatibility with Adobe Flex. The first was a market- and industry-leading testing tool (“Product A”); the second was a testing tool from a much smaller company (“Product B”). The evaluation focused on four criteria which best suited Celerity Innovation Center’s three main goals for its development projects. The three goals were: 1) meeting customers’ requirements; 2) supporting development’s need to know what and how to optimize; 3) supporting software testers in achieving efficiency and a high-level of software quality without sacrificing one for the other. The four main criteria for the evaluation were: 1) application compatibility; 2) tool usability; 3) documentation and reports; and 4) support, help and communication. After the tools were analyzed, the researcher realized a couple of additional criteria, which are expanded upon in this document.

1) Application Compatibility: The researcher evaluated how well the two load and performance tools read and layed-back Adobe Flex applications, rendered the code and gave a full, accurate analysis of the application’s development. Prior to choosing the two final tools for this evaluation, the company found that many load and performance testing products on the market have difficulty with accurately rendering Adobe Flex code to perform tests against it. There were several such issues with Product A. Product B, however, had a fantastic ability to read Adobe Flex applications, converting Flex requests to XML. It also had a better built-in collation, serialization and de-serialization of Action Message Format (AMF) than the market-leading tool. Product B’s support of browser polling and server streaming of AMF requests, its capacity to integrate the interactions of binary data variables for searching or identifying requests and responses, along with its ability to easily set up performance parameters, were all a plus. Because of these points, Product B also allowed for quick re-execution of tests, which made the tool very attractive to the company.

2) Tool Usability: The usability of both testing tools' interfaces was evaluated, taking into account the time it took for a user to understand how each tool worked. The user had to understand both tools well enough to be productive and efficient. Product B's interface, which created the load and performance test scenario code script, was very easy to understand and leverage. Product A was less powerful and more complicated overall, and did not have as many advanced features as Product B. The company saw the simplicity of Product B as a way to quickly and seamlessly create scripts as tests are launched, and track activity without added complexity. The testing status and analysis of testing activity for Product B were straightforward, easy to understand and translate. There was no report of Product B crashing, which was a concern of the company based on its past experience with similar tools – including Product A. Product B appeared to be more dependable, stable and reliable than Product A; it did not drag any of the load tests, nor did it lock the screen and require a computer reboot.

3) Documentation and Reports: The documentation from the executed tests of each load and performance testing tool was evaluated, including the formats available for disseminating the information and their capacity for customization. Product A provided a documented report that was easy to generate and read. Product B also provided an easy-to-read document, but with added illustrations. Both Products' reports summarized all factors that were set up to be tested and analyzed in an easy-to-export fashion, through Microsoft Word or .html format, for example. Once exported, the layouts could be captured, analyzed and modified.

4) Support, Help and Communication: Both tools were evaluated on the basis of how much corporate support was offered to back up each respective product, and how well each customer service team assisted the end-user. Product A offered a full online collaboration board, but no live chat or moderator. This resulted in several unanswered questions, and other questions receiving "unofficial" suggestions from users who had similar questions or problems. The online collaboration board was neither effective nor as helpful as Product B's live customer service. Product B had a reliable question-submission process in which the user could either call customer support and speak to a live person, or submit a question through a live Internet chat. The response from the customer service team was prompt and courteous, thoroughly resolving all issues and answering all questions.

C. Additional Evaluation Criteria

The following two additional criteria arose as the company evaluated the two load and performance testing tools.

1) Monitoring and Controlling: While evaluating the two testing tools, the company realized that there was a need for a monitoring and controlling function, particularly during testing and software production. Such a function is necessary to document, analyze and mitigate activities for WebSphere environments. While such technology is rarely used for Web testing, it harbors the potential to reduce network bandwidth and achieve a higher level of robustness as compared to conventional methods [14]. Product B happened to have such a function available that was more precise, easier to manage and easier to put into practice then Product A's solutions. In addition, Product B's ability to use monitors made it's integration more seamless than Product A. With Product B, no other add-in, secondary tool or installation was necessary – all that was required was to obtain the monitor license and start using it. The company decided to purchase several monitoring licenses for Product B to test and assure various development, test and production environments. The researcher found that using these monitors gave the company the ability to observe each server response, 24 hours a day, seven days a week – which helped the organization assure and certify its products while still in the development phase. Had the company chosen Product A, the time needed to implement the monitor and controlling function would have doubled.

2) Pricing and Ease-of-Purchase: When the price of the two load and performance testing tools was compared, the company found that Product B would save tens of thousands of dollars when taking into account the total cost of users during testing, plus the cost of virtual users, protocols and monitor licenses. Pricing information was very clearly outlined on Product B's Web site, thereby narrowing the margin for error in interpretation. When the company contacted Product B's sales team via e-mail, the inquiry was returned promptly with all of the information necessary for the company to purchase the tool. The sales representative followed up with the researcher to make certain that the company had what it needed to make an informed decision about the purchase, without being pushy about the sale. Even though Product A was the market-leading tool, its sales team was less "friendly" about the purchase, forcing the company to meet with them several times and during which additional items (largely unwanted by the company) were pushed as a part of a package deal. The motive was clear, and it was not long before the company began to feel like it was being taken advantage of by Product A's sales team.

DATA ANALYSIS

Product Selection and Tool Usage Analysis

Product B was finally selected as the tool of choice for load and performance testing. It out-ranked Product A in nearly every criteria except for documentation and reporting, in which both products were deemed comparable.

After the evaluation and purchase, Product B's first load and performance test was a single script of a large piece of functionality impacting the company's system under development and integration. This single script test would ensure system performance and integration responses to other systems. The total time dedicated to creating this script was 16 hours, or two working days. During this time, one test-automator created and executed the script, then made sure it read Adobe Flex and RIA objects and brought back accurate responses to the system, integration points and the company's infrastructure team. The infrastructure resources (one network resource, one database resource and one WebSphere resource) also made sure that the test script was accurate. After the creation and analysis of the test script, the automation test was used to simulate the action of actual users, as if they were testing the system's performance and capturing the results. Due to the virtual users, no infrastructure resources were required to ensure the test after the script was created, which eliminated the need for three additional resources plus the manual testers who would create the load test scenario manually.

Product B executed 11 consecutive, identical tests of a single developed test script, with different virtual users for each test execution. Table I shows a breakdown of components for the load and performance test automation scenario, with the number of resources needed (one test automator), the minimum number of virtual users needed, total pages and total hits executed, and the actual number of virtual users launched for each test. Each load performance automation test consumed only a half-day (approximately four hours) to execute and analyze, per resource.

Table II outlines an estimation of a manual test automation scenario for Product B, including how many live users and the number of hours it would take to execute 11 consecutive, identical tests manually. Each manual test would need a full day to execute and analyze. It is estimated that each manual test would require one test coordinator to execute the script, one network resource, one Database resource and one WebSphere resource, plus an average of 33.5 manual testers to complete the "virtual load."

Table III demonstrates the time and resources saved by executing an automated test scenario with Product B instead of a manual test scenario. The difference between the two equals a savings in total resources and hours; 2,908 hours and an average of 32.5 resources were saved by the automated test scenario.

Table I. "Product B" Load and Performance Test Automation Scenario

Test No.	Test Date	Virtual Users	Total Pages	Total Hits	Virtual Users Launched	Total Resources	Hours
1	5/9/2011	10	311	1289	24	1	4
2	5/12/2011	10	3997	14703	226	1	4
3	5/12/2011	25	9306	34122	517	1	4
4	5/12/2011	50	9972	36564	554	1	4
5	5/13/2011	50	81756	299772	4542	1	4
6	5/19/2011	50	13860	50304	770	1	4
7	5/20/2011	50	12780	45701	710	1	4
8	6/1/2011	10	250	660	10	1	4
9	6/1/2011	10	250	660	10	1	4
10	6/3/2011	10	250	660	10	1	4
11	8/3/2011	50	13176	47482	732	1	4
		29.5 (average)	**145,908**	**531,917**	**8,105 (total)**	**1 (total)**	**44 (total)**

Table II. Estimated Time and Resources Needed For Manual Test Scenario

Test No.	Resources Needed	Hours to Execute
1	14	112
2	14	112
3	29	232
4	54	432
5	54	432
6	54	432
7	54	432
8	14	112
9	14	112
10	14	112
11	54	432
	33.5 (average)	**2,952 (total)**

Table III. Actual Time and Resource Savings for "Product B" Load and Performance Test Scenario

Test No.	Resources Saved	Hours Saved
1	13	108
2	13	108
3	28	228
4	53	428
5	53	428
6	53	428
7	53	428
8	13	108
9	13	108
10	13	108
11	53	428
	32.5 (average)	**2,908 (total)**

Table IV. Test Efficiency Comparison: Estimated Manual Test Scenario for "Product B" VS. Actual Load and Performance Test Scenario for "Product B"

Test No.	Resources Saved by Using Automation Testing	Virtual Users vs. Manual Users	Automated Test Efficiency Per Test, Per Resource	Manual Test Hours Used over Automated Test Hours Used
1	13	24 vs.14	50% more efficient	27
2	13	226 vs. 14	50% more efficient	27
3	28	517 vs. 29	50% more efficient	57
4	53	554 vs. 54	50% more efficient	107
5	53	4,542 vs. 54	50% more efficient	107
6	53	770 vs. 54	50% more efficient	107
7	53	710 vs. 54	50% more efficient	107
8	13	10 vs. 14	50% more efficient	27
9	13	10 vs. 14	50% more efficient	27
10	13	10 vs. 14	50% more efficient	27
11	53	732 vs. 54	50% more efficient	107
	32.55 (average)	**650.72 (avg. difference)**		**66.09 (average)**

Table IV demonstrates the difference in efficiency between the estimated manual test scenario for Product B and the actual load and performance test scenario for Product B. An explanation of each column follows below.

A. Resources Saved by Automated Testing

The automated load and performance test required only one resource at all times during execution, while the manual test would require an estimated high of 54 people (during tests four through seven). So, the maximum number of resource savings by using automation is "53." That is, 53 additional people would be needed to manually execute a load and performance test (a total of 54 people) than if it were executed using an automation tool. The same principle applies to the other rows in this column. The average resource savings for automation was recorded at 32.55.

B. Virtual Users vs. Manual Users

The more users that can execute a test, the better. For the manual test scenario, only a maximum of 54 users would be able to execute a test (during tests four through seven). With automation testing, the largest number of virtual users recorded was 4,542, which occurred during test number five. This represents a tremendous difference of 4,488 users. The sample principle applies to the other rows in this column. The average difference between virtual users and manual users was 650.72.

C. Automated Test Efficiency Per Test, Per Resource

The manual test scenario would have used at least 8 hours, per resource, to complete one test. The automated test only used 4 hours per resource to complete any one test. This resulted in the automated test saving 50% more time per test, per resource, than the manual test. In effect, the automated test was 50% more efficient than the manual test, per test, per resource.

D. Automated Test Hours Used vs. Manual Test Hours Used

At its most rigorous point, the manual test required 54 resources to use 432 hours to finish one test (during tests four through seven). The automated test scenario used only four hours to complete any one test, even with 4,542 virtual users (during test five). Thus, the manual test used 107 times more hours than the manual test at its most rigorous point. The same principle applies to the other rows in this column. The manual test used an average of 66.09 times more hours than the automated test.

CONCLUSION

As this research shows, there has been an apparent and growing need in recent years for Adobe Flex developers, and software testers for Adobe Flex and RIA in general. More companies are seeing the value of testing their RIA prior to going live, in order to maintain fluidity and compatibility across their software offerings and the unpredictable Internet. Some companies use manual testing to accomplish this, which is not very efficient in that it requires hundreds of hours of time and live users to execute the scenario from start to finish. Load and performance testing, however, allows for thousands more virtual users than would be possible with live users, and it can execute the tests among those virtual users simultaneously, which saves hundreds of hours of time. Testing under a large number of users prior to going live with an RIA is absolutely vital toward gaining a clear understanding of how the application will perform in various circumstances [7].

"Product B" fit Celerity Innovation Center's original criteria seamlessly, but the tool also helped the company meet its needs for monitoring and controlling, and without gimmick pricing schemes. As with Product B, any load and performance testing tool should not waste time and effort in creating and executing load and performance test scripts. Nor should it waste time and resources in rendering the application to properly pick up and analyze its data parameters and environment correctly. This is ultimately what a load and performance testing tool should do: increase efficiency while reducing risk, increase quality and ensure that the software meets its requirements to the best of its ability.

Product B emerged on the top tier of load and performance testing tools for Celerity Innovation Center, due to its compatibility of testing Adobe Flex architectures. But, there are still many RIA testing tool products on today's market that companies should evaluate and choose from for this purpose. The criteria outlined in this paper should become a standard guideline during the evaluation process of RIA tools.

REFERENCES

1. Adobe Systems Incorporated. (2011). Retrieved from http://www.adobe.com/products/flex.html
2. Fraternali, P., Rossi, G., & Sanchez-Figueroa, F. (2010). Rich Internet Applications. IEEE Internet Computing, 14(3), 9-12.
3. Stat Owl. (2011). [Bar graph illustration of RIA support over five months.] Rich Internet Application Market Share: RIA Market Penetration and Global Use. Retrieved from http://www.statowl.com/custom_ria_market_penetration.php.
4. Goldberg, A. (2011). Smartphones exploding thanks to Android. McClatchy - Tribune Business Information Services. Retrieved from ABI/INFORM Dateline.
5. Cerf, V. (2011). Open Source, Smart Grid, and Mobile Apps. IEEE Internet Computing, 15(1), 96. Retrieved from ProQuest Computing.
6. Bielski, L. (2008). 2008: Year of Rich Internet Apps? ABA Banking Journal, 100(4), 35-36. Retrieved from ABI/INFORM Global.
7. Iyer, A., Chatterjee, A., & Kishnani, J. (2010). Compiler Transformations to Enable Synchronous Execution in an RIA Runtime. IEEE Internet Computing, 14(3), 13-23. Retrieved from ProQuest Computing.
8. Blumenstyk, M. & Decker, R. (2001). Performance testing: Insurance for Web engagements. Consulting to Management, 12(4), 57-60. Retrieved from ABI/INFORM Global.
9. McManus, S.E. (2008). Influence of the CSI Effect on Education and Mass Media. Retrieved from anthropology.cos.ucf.edu/content/graduate/thesis.html
10. S. Tiwari. (2008, Mar. 14). Adobe Flex Gaining Ground in India; Java IU – Watch Out! Retrieved from http://www.oreillynet.com/onjava/blog/2008/03/adobe_flex_gaining_ground_in_i.html

11. Gan, X. (2006, Sep. 9). Software Performance Testing. Retrieved from http://www.cs.helsinki.fi/u/paakki/gan.pdf.
12. [Linaje, M., Preciado, J., & Sanchez-Figueroa, F. (2007). Engineering Rich Internet Application User Interfaces Over Legacy Web Models. IEEE Internet Computing, 11(6), 53-59. Retrieved from ProQuest Computing.
13. Celerity Innovation Center Chooses Neotys' NeoLoad to Performance Test State-of-the-Art Adobe Flex Web Applications. (2011, Oct 3). Retrieved from http://eon.businesswire.com/news/eon/20111003005331/en/web-application-development/load-testing/performance-testing.
14. Chang, W.C., & Chuang, M.H. (2004). Performance monitoring of remote Web sites using mobile agents. Software Quality Journal, 12(2), 159-176. Retrieved from ABI/INFORM Global.