

How Adobe Flex can help to save on application development costs

Rich Internet Application (RIA) technologies in general and Adobe Flex in particular promise richness, usability and productivity exceeding those of today's web applications. As with most of relatively new technologies on the market, RIA also promises lower cost of development and maintenance.

You'll learn:

- What is RIA (Rich Internet Application)
- How to measure efficiency of application development
- Efficiency of Java/Flex versus Java/AJAX/HTML application developments (large projects)

You should know:

- Three-tier enterprise Java application architecture

However, little public information is available on measured results and savings achieved. This paper fills this much needed gap by providing results achieved by Verax Systems in real-life projects and is intended to provide real data rather than high-level marketing statements only. The purpose of this paper is to highlight the most important issues related to the efficiency of Adobe Flex application development for our customers and partners.

Introduction

In theory, the cost of Internet application development with Adobe Flex should be lower as compared to the classic HTML/AJAX application at each stage of development:

- **Design:** The GUI design process takes place directly in development tools in the fully WYSIWYG ("what you see is what you get") manner, resembling Visual Basic or Delphi application development. The key advantage of such an approach is that business users and designers can use the same Flex builder software as developers, which allows them to get involved in early stages of software development.
- **Coding:** Adobe Flex Action Script is an easy-to-use programming language with higher productivity for user interface than Java and popular extensions such as Apache Wicket. It is also less error prone to programmer's mistakes. An additional

benefit is a large number of commercial and open source GUI components. Adobe Flex components can be used similarly to Visual Basic OCX controls. Also, the separation of GUI from the back-end (or the server side) leads to clearer, more understandable code.

- **Deployment & testing:** One of the key reasons why Verax Systems decided to evaluate RIA technologies was the elimination of the "browser compatibility hell". With a high number of browsers on the market (Internet Explorer, Firefox, Chrome, Safari, Opera to name the key players) and compatibility problems across individual versions (e.g. IE 6, 7 and 8), achieving browser compatibility via CSS/HTML is a big and costly problem.
- **Maintenance:** Whereas the design to testing factors are relatively easy to measure and account for, the support and maintenance costs are often underestimated. At Verax Systems we measure those as number of bugs reported and efforts required to resolve them.

The subsequent sections describe Verax Systems' results on cost comparison. At this stage, it must be stressed that this paper discusses complex application development not a web site development. Furthermore, in the scenarios discussed, introduction of Adobe Flex on the front-end did not eliminate server-side Java code.

RIA and Flex technology overview

RIA is a term coined by Adobe/Macromedia for web applications (i.e. running within a web browser) that have characteristics of traditional desktop programs. Such applications are stateful, have local data on the client side and communicate with the server mostly for data interchange.

Traditional web applications are built using client-server and thin client architectures which require server involvement for user-interaction. RIA shifts processing for user-interaction to client side, resembling traditional desktop approach. RIA retains most benefits of classic web application such as centralized upgrades, use of browser cache, etc.

Leading RIA technologies are Adobe Flex, Microsoft Silverlight and Sun JavaFX. Some authors also include AJAX, Curl, Apache Wicket as RIA platforms, however at Verax Systems we deem and refer to these as “classic web” development technologies.

It is not our intention to describe RIA in detail in this paper as there are many resources available on the Internet.

At this stage however, it is worthwhile to list the most frequently mentioned benefits of RIA:

- **End-user benefits:** enhanced user productivity, higher customer experience index, shorter UI response times, multimedia and Web 2.0 integration, user guidance.
- **Technical benefits:** web browser and operating system independence, central upgrade and distribution, pervasive runtime (in case of Adobe Flash), lower hardware requirements for the server side (GUI processing/rendering takes place on the client machines), lower network bandwidth required (only data is transferred without formatting such as images, CSS or HTML page layouts).

Again, RIA is not a silver bullet and is not a good choice for all applications. At Verax Systems we believe that RIA is good for complex applications with many GUI fe-

Table 1. APINI and NMS technology comparison

Used technology comparison	
APINI	NMS
• Front and components: DOJO and internally developed controls	• Front and components: Flex with charts
• MVC: FreeMarker and Struts	• MVC: Cairngorm
• Client Server Communications: AJAX, HTTP	• Client Server Communications: Adobe LCDS or Blaze DS
• Server side: Java, JSP	• Server side: Java, Spring
• Database Persistence: Hibernate and HQL	• Database Persistence: Hibernate and HQL
• Database support: Oracle and MySQL	• Database support: Oracle and MySQL

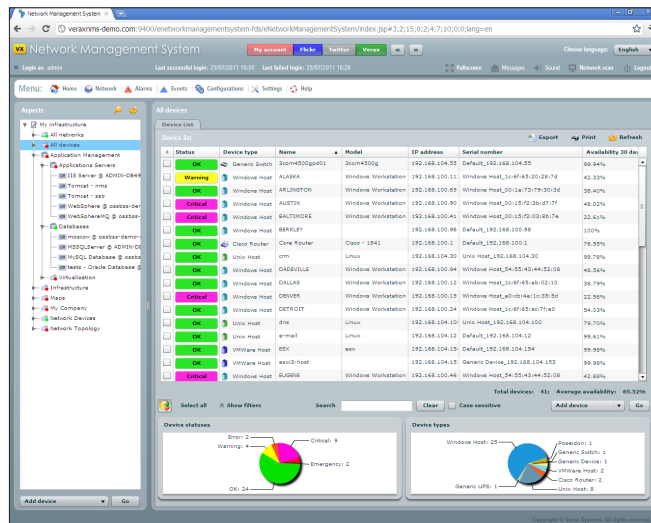


Figure 1. Verax NMS - Aspects view (Java/Flex)

atures where user experience is to be high, such as electronic banking, electronic reporting, management KPI dashboards or knowledge bases, to name a few.

Case analysis

Compared applications overview

The analysis was performed on two applications continuously developed by Verax Systems, within our Research and Development unit. The purpose was to measure efficiency of Java/Flex versus Java/AJAX/HTML application developments. The measurements were taken over a six months period, both projects were similar in team sizes. The following applications were compared:

- **APINI** (<http://www.veraxsystems.com/en/products/apinikb>) is a Web 2.0 portal system for gathering and exchanging information in knowledge driven enterprises. It also contains task, project and resource management features for management of project-oriented organizations.
- **Verax NMS** (<http://www.veraxsystems.com/en/products/nms>) is a highly scalable, integrated network, data center and application management and monitoring system featuring full FCAPS (fault, configuration, accounting, performance, security) functionality.

Table 2. KPI values for compared applications

KPI	NMS (RIA)	APINI (classic web)	% difference	Comment
Number of new views developed	21	15	40%	Good, more functionality achieved
Number of issues	475	1087	-56%	Good, fewer problems
Karma produced (average per developer per month)	29372	21531	36%	Good, more lines of code written - developers focused on development rather than other issues (e.g. achieving browser compatibility)

Both applications share a similar architecture (three-tier enterprise Java running on the Tomcat application server) and differ mostly in how the front-end (GUI) is implemented.

Detailed lists of the technologies used in each product are provided in the Table 1.

Comparison methodology

Key assumptions

- The measurements were taken over a six (6) months timeframe.
- Each product team consisted of 4-5 developers and one tester (the effort of man-hours used for comparison differed by 5%).
- Both system sizes (counted as lines of code) were similar (about 20% difference, the development times were two and three years).
- Both projects used identical methodology for project development.
- Both projects used identical tools for development (Maven, Eclipse, CVS, JProbe and others). There were minor discrepancies in Eclipse plugins used and use of Flex Builder for the NMS.
- Both projects followed identical, standard Verax GUI design guidelines.

KPIs used for measurements

- Elements of functionality developed counted as number of views compliant to the Verax GUI design guidelines including the back-end functionality (e.g. object persistence, filters, etc.). This measures pure developer productivity in terms of functionality i.e. what is the efficiency of building new views.
- Number of GUI related issues reported by testers (these included not only bugs, but also items like poor performance, poor user-experience, non-intuitive use, etc.). This measure reports the quality of the software delivered.
- Source code Karma factor (as calculated by CVS monitor) meaning developer productivity in terms of creation of code. Karma is a better KPI than number of lines of code (please refer to <http://sourceforge.net/projects/cvsmonitor/> for details).

The general approach taken was to favor classic web application development: use optimistic values for HTML and pessimistic ones for RIA.

Results obtained

The table below contains a summary of results for KPIs described above in both projects (Table 2.).

The above states clearly that the engineer productivity increases about 40%, even for a conservative comparison. Moreover, the views developed for the NMS are far richer: for instance they contain mini dashboards with charts and trend graphs.

This is definitely an advantage, however analysis of additional costs such as recruitment, initial training, learning curve and additional license costs has to be provided. All of these factors can be treated as an investment, for which ROI has to be demonstrated. At Verax Systems we have analyzed the following categories:

Table 3. Cost per seat of Adobe Flex introduction

Investment	RIA/Flex	Classic web
License costs	799 USD	0
Training	5 man days	0
Learning curve	6 weeks	0

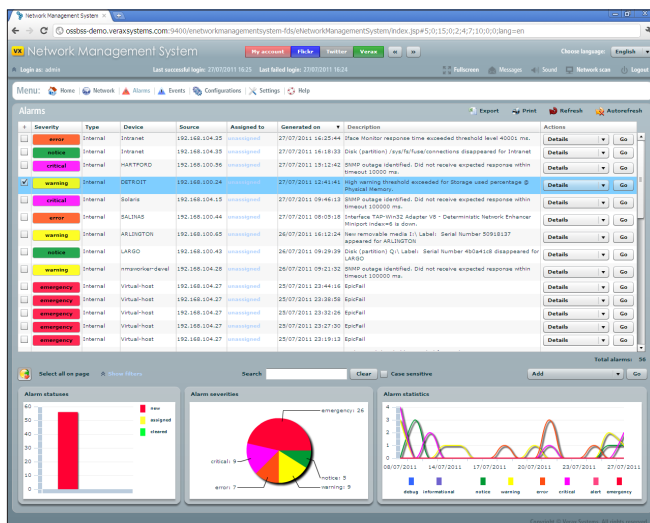


Figure 2. Verax NMS - Alarms view (Java/Flex)

- **License costs.** Adobe Flex requires a per-seat development license whereas HTML development usually does not require any commercial licenses and relies mostly on open source packages such as Eclipse. Verax Systems uses Adobe Flex Builder with charting which is more expensive than the standard edition.
- **Training costs.** Only the Flex Builder tool training was taken into consideration, rather than particular system development training (which is usually required anyway). For the purpose of comparison, it has been assumed that classic web development tools are known by developers due to their popularity (which may not always be the case).
- **Learning curve.** This factor measures how long does it takes for a developer with no prior RIA experience to become a fully productive team member. We estimate the learning curve to be approximately six weeks – this value is based purely on Verax Systems' experience in our development environment and applications.
- **Skill levels required.** This measure (not listed in the table below) measures developer skills required to use the technology. Based on Verax Systems' experience there are no particular skill level requirements for Flex. In fact, Flex allows lower skilled developers to be very productive.

The investments related to RIA introduction are listed in the Table 3.

Given the above, at Verax Systems we estimate about 3 months ROI on the RIA/Flex introduction. Once the ROI has been achieved, the 40% more effective software development factor holds true. We expect the ROI to be



Figure 3. APINI - Home view (Java/AJAX/HTML)



Figure 4. APINI - Project statistics view (Java/AJAX/HTML)

shorter in the future as RIA becomes more popular. Currently, the trained developer pool is limited and training has to be provided internally.

Conclusions

Verax Systems has chosen the Adobe Flex as its primary platform for GUI development. The initial key driver for the adoption of RIA technology was efficiency of software development (understood as higher developer productivity and lower cost of issue resolution). Additional benefits, that the company took advantage of after a while lied in: user experience, user productivity and other factors normally pitched by RIA advocates.

NICK ZMIERCZAK

Nick is a CTO of Verax Systems. He has over 15 years of experience in both IT and telecommunications, mostly in the area of design & development of large scale enterprise applications.

nick-zmierczak@veraxsystems.com