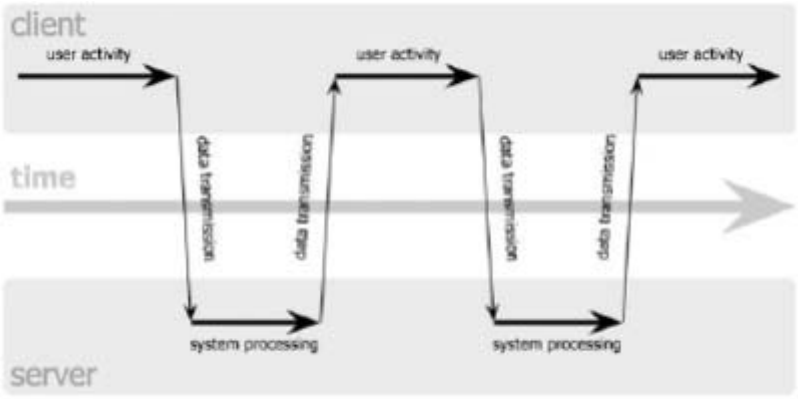
Rich Internet Applications (RIA)

- Krithika Venkatesh

The Web was originally intended to help researchers share documents as static pages of linked text formatted in HTML. From there, Web pages quickly evolved to include complex structures of text and graphics, with plug-in programs to play audio and video files or to stream multimedia content. Web developers supplement the basic browser function of rendering HTML by invoking code *(scripts)* on the user's computer *(the client)*. These scripts can execute UI methods, for example, to validate a user's input in an HTML form. Today users can book travel, check email, buy and sell stocks, submit tax forms or listen to music from any networked computer.

But these do not change the fundamental model in which application logic runs on the server and executes between Web pages after the user clicks. This behavior is said to be *synchronous*, that is,after each click the user waits while the server handles the input and the browser downloads a response page.



To overcome the above difficulty, a new technology is introduced to free the web page from the page model by enabling updates to specific parts of the page instead of a distracting page reload. This new technology is called ***Rich Internet Application (RIA)***.

Rich Internet applications (RIA) are **Web applications** that have the features and functionality of **traditional desktop applications**. RIAs typically transfer the processing necessary for the **user interface to the Web client** but keep the **bulk of the data** (i.e., maintaining the state of the program, the data etc) back on the **application server**.

The ability to update part of a page enables the creation of a richer set of controls, allowing users to interact directly with page elements through drag and drop, direct text editing, resizing objects, moving sliders and other techniques.

For example, most map sites load maps around the area currently being viewed. This allows users to click and drag the map to instantly view more of the map with no delay, giving a sense of

direct interaction and immediate responsiveness. It is this type of natural interaction and feedback that make RIAs so appealing and engaging.

The ***RIA Behavior Model*** summarizes three factors that together determine RIA performance:

1. The application's design and usage environment, or context
2. The user's expectations and behavior
3. The application's behavior when used

A user's experience of response time depends on the combined behaviors of the client and server components of the application, which in turn depend on the application design, the underlying server infrastructure design, and, of course, the user's Internet connection speed. The most effective RIA will be one whose creators took into account these factors at each stage of its development life cycle, and who created the necessary management processes to ensure its success when in production.

Today's rich Internet applications are almost all built on either one of three technological platforms: **AJAX**, **Macromedia Flash**, or **Java**. A comparison chart can be seen below.

Table 1. Platform Comparisons (Source: UWEBC) — Good / Average / Poor

	AJAX	Macromedia Flash	Java
Graphical Richness	Average (Same as HTML)	Very Rich	Rich
Container/Engine Footprint	Very Light (browser built-in)	Light	Heavy
Application Download	Fast	Slow	Slow
Audio/Video Support	Poor (unless use ActiveX)	Excellent	OK
Consistency on Different Computing Environments	Varies	Very consistent	Relatively consistent
Server Requirements	None or very minimal (TIBCO General Interface)	Yes (Flex or Open Laszlo)	Yes or No (Nexaweb, Java Web Start)
Plug-in/Runtime Requirement on Client	No	Flash (Player)	Java Runtime (JRE)
Development Challenge	Very complex without tools such as TIBCO, and high skills required (JavaScript, CSS, XML, XSLT, DOM, ActiveX...)	Relatively easy with tools such as Flex or Open Laszlo (XML, DOM, JavaScript, Flash, ActionScript)	Relatively easy with tools such as Nexaweb (XML, JavaScript, Java)
Security Concerns	JavaScript codes are open to public. Everybody can see source codes if desire	Flash files (compressed binary) are created. Flash Player becomes a sandbox	Class/Jar compressed binary files are created. JVM (Java Runtime) becomes a sandbox
Cost	Custom Build - Free TIBCO - Unknown	Open Laszlo - Free Flex - $15,000 per CPU	Java Web Start - Free Nexaweb - Unknown

Lets take a look at some **guidelines** for using RIAs to improve the user experience.

1. Determine where richness is appropriate

RIAs can be entire applications, certain sections of a site, or just rich elements added to traditional Web pages. Give careful thought to where richness adds value and where traditional

pages make more sense. For example, traditional html pages are more than adequate for displaying content such as news stories. Rich interface elements could be used to navigate to those stories or to interact with them.

2. Know your users

Knowledge of your users and the tasks they are trying to accomplish is important with any design but even more so with Rich Internet Applications. Guidelines, proven conventions, and prior experience can guide much of your design decisions with traditional Web sites and Web applications.

3. Borrow from familiar conventions and maintain consistency

Where appropriate, borrow from the conventions of HTML and desktop applications so that users can quickly learn how to use the application.

4. Controls should communicate their purpose and function

Controls must make their presence and purpose visible. A control's appearance should communicate what can be done with it and how to use it. For example, the traditional 3D appearance of buttons and the way they seem to press down when clicked, simulate real world buttons and communicate their purpose. Whichever type of controls and interactions you use, they should operate consistently within your application and consistently with other similar rich sites.

5. Ensure that the back button and bookmarking work

Consider where people will be likely to click the back button and either use separate pages or use methods to enable the back button. Use methods that allow bookmarking and sharing links where appropriate.

6. Communicate change

With partial page updates, it's very important to make sure that people notice those changes. In order to be noticeable, changes should occur close to the area where people are looking, and only one update to a page should be made at a time. Be careful not to overdo page updates so that users aren't overwhelmed with multiple changes that break their concentration on their task.

7. Provide feedback when changes are not immediate

Although RIAs are known for quick feedback, sometimes an action does not produce an immediate response. In a traditional Web site, waiting for a page to load is obvious feedback that something is happening. Without a page refresh in an RIA, unless there is an immediate response to an action, it can appear that the action has had no effect. In this type of situation, some kind of feedback is necessary to show that something is happening. A common technique is to bring up a small animation in the area of the page where the update will take place.

8. Consider solutions for people who can't use RIAs

Unfortunately RIAs are difficult if not impossible for people with certain disabilities to access. Other devices, such as mobile phones and PDAs, also may not be able to access rich sites. As with any design, these audiences' needs should be considered, and accessible or alternative versions may be necessary.

Rich Internet Applications are a great improvement over the limitations of traditional Web sites. With good knowledge of your users, careful design, and usability testing, rich internet functionality can greatly improve the user experience of your site.