

# Assessing the Value of Rich Internet

...identifying the user experience value of rich interaction models

A **USER EXPERIENCE SERIES** whitepaper

By **DAN WILCOMB** | Axis Information Architect

Despite all the hot press around Rich Internet technologies, user experience professionals are often at a loss to identify opportunities or assess the real value of rich interactions.

## Introduction

Building Web applications has always been akin to fitting a square peg in a round hole - the design effort inherently attempting to artificially guide user behaviour into linear processes. To its credit, the design community has developed techniques and usage patterns that accommodate most interaction types, but even these efforts are still restricted by the basic nature of Web technologies: server calls and responses, page loads and refreshes...linear, sequential interactions. Rich Internet Application technologies could change all that, but what's stopping them, and how do we in the design community avoid falling into old habits when utilizing these new tools?

## About Rich Internet

Rich Internet Applications (RIA) and the technology frameworks that support them are a key component of the much-hyped Web 2.0. These technologies allow for more desktop-like behaviours in Web applications than was previously possible – or at least practical – using traditional Web technologies. Using these technologies, it is possible to develop modularized applications with inter-communicative components, non-linear user processes, and application-initiated interactions. There are a wide variety of open and proprietary RIA platforms emerging in the industry. At the moment, the most prevalent are:

- **AJAX** – Asynchronous Javascript And XML. An open standard and set of coding techniques for developing Rich Internet Applications using a combination of client-side scripts referencing XML data.
- **Flex** – Developed by Macromedia and acquired by Adobe, Flex is an MPL open-source rich internet solution utilizing server software, an Eclipse-based IDE, and XML-like code that renders through a standard Flash browser plugin.
- **SilverLight** – A Microsoft cross-platform framework for delivering rich media and interactions, advertising vector graphics rendering, overlays, and integration points into existing standards such as AJAX.
- **Ruby on Rails** – a full-stack (database-to-interface) open-source solution for rapid coding and release of rich Web interfaces using the Ruby language.

Each of these platforms has its benefits and drawbacks, which we will not discuss in detail here. What is important to note is that these applications do some things not previously possible on the Web, and many more things that were possible, but constraints around bandwidth, difficulty of development and maintenance, and minimal reusability were so severe as to make them impractical. The newest generation of technologies promise to deliver rich experiences with a combination of rapid development through familiar IDEs with customizable libraries, ease of maintenance and reuse through reliable standards, and higher utility through more dynamic and non-linear interaction models.

## Benefit 1: Instant Feedback

Traditionally on the Web, any update to information being viewed on the screen required a server call, server response, and a refresh of the page, meaning that a user had to commit to an often time-consuming load process in order to get results or feedback on even simple activities. While client-side Java, Flash components, etc. have tried to surmount this hurdle, the solutions have always been difficult to develop and maintain, and have existed mostly in isolation on a page, rather than integrated into the user experience. Rich Internet technologies permit a certain level of client-side processing, enabling instant feedback to be provided to the user in even more complex activities such as charting and spreadsheet data manipulation.

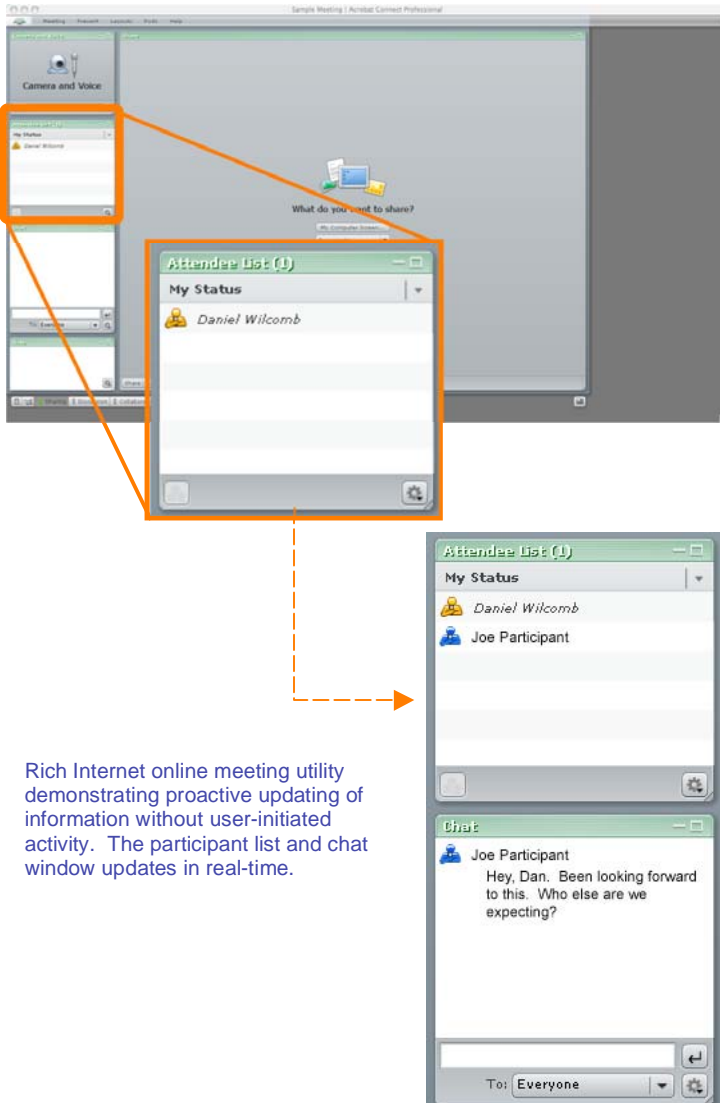


Demonstration of instant feedback – changing one piece of information (Time Period) updates all relevant graphs on the page instantly, and without a screen refresh.



## Benefit 2: Proactive System Interactions

Another drawback to the request-response model of the Web is the need for user initiation of a system activity. (e.g., you need to click “update cart” to calculate a new total, or click “refresh” to see the latest news headlines.) This is particularly detrimental to workflow-driven processes, where an action by the system or another entity of the system needs to prompt user activity – such as when a request requires approval, or when the status of an item of interest changes.



Rich Internet online meeting utility demonstrating proactive updating of information without user-initiated activity. The participant list and chat window updates in real-time.

## Benefit 3: Modularity

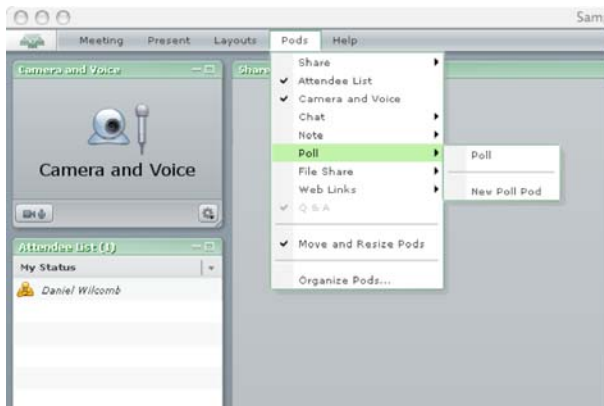
Dynamic page-level interactions have always been problematic or impossible for traditional Web applications. Any inter-page communication is typically simulated through a database post and a whole-page refresh maintaining the state of all the entities contained on the page to provide the illusion of communication. In addition to being relatively slow, these techniques also lead to some convoluted code, a need for a dedicated application database to handle UI state information, and high effort in maintaining and updating functionality. Rich Internet platforms allow true communication between page elements and background calls to the database without any need for a whole page refresh, and all the commensurate smoke-and-mirrors involved in making it appear that a refresh didn't really happen. Updates of individual modules instead of the whole page dramatically reduces load times and also makes true page-level interactivity possible.



Component management utility demonstrating how updating information in one module (Style Controls) automatically communicates the update to another module (Sandbox)

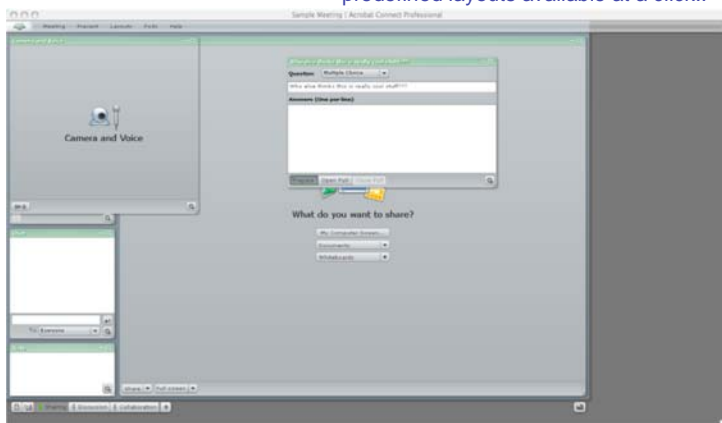
## Benefit 4: Customizability

The modularity and statefulness of a rich interface also lends itself to high user customizability. Portal servers have long allowed users to “build” pages to suit their needs, utilizing pre-canned tiles laid out on a set grid. While good in concept, the need to manage this layout in a separate mode and with clunky Web interactions led most portal administrators to turn off the feature, or most users to simply ignore its availability. Modularized rich internet applications have the potential to provide in-mode customizability in a much more dynamic manner than possible even in the best portal environments.



Above: Online meeting utility demonstrating the ability to add or remove modules (“pods”) in-mode.

Below: Same utility demonstrating ability to customize page content and layout, with predefined layouts available at a click..



## Challenge: Perception of the Technology

The tools are there, but despite their growing prevalence, technology sponsors in established industries are often reluctant to consider them for application development initiatives. In the minds of many evaluators, these technologies have been relegated to the Web animation stockpile – just another moving banner for a marcom site.

Adding to the misconception, many of us in the user experience industry have been unable to successfully demonstrate tangible value in the technologies, or even to make effective use of them in designing interactions. In some cases, the industry simply hasn't explored the appropriate use of these technologies. In other cases, we haven't taken the time to reassess how we describe and work with the interactions made possible by rich models. We've designed and built for an HTML environment for so long, we've been indoctrinated into the mindset of traditional Web interactions.

The solution to these problems is evolutionary – the industry needs to build an understanding of the value these technologies provide, a defensible position on assessing opportunities, and a body of work showing demonstrable success in providing enhanced user experience.

## Challenge: Successfully designing for and documenting dynamic interactions

- More attention must be paid to combinations of view and state. The complexity can get out of hand quickly.
- Development advantages start to wane if designers deviate significantly from the provided libraries.
- Design changes can have broader implications now that interactivity between multiple modules must be accounted for. Changes that appear localized may have wider impacts than anticipated.
- Sitemapping and page flow exercises become less relevant. A storyboarding mechanism is needed to describe intended flows. Deliverables and design vocabulary will have to evolve.
- Customizability simplifies page grids in some respects, but complicates visual design in others. (i.e., bitmap graphics in scalable modules. Unreliable size constraints, z-order considerations, etc.)

## In Closing

RIA is not synonymous with Web 2.0, but they are complementary. Web 2.0 projects are typically driven by user contribution and community-interaction, and they tend to make heavy use of Rich Internet technologies and techniques. When assessing RIA, designers and business sponsors alike must be careful to judiciously select situations that are appropriate to the use of a dynamic, often non-linear interaction model. Designers must adhere to traditional principles of technology selection, and consider RIA platforms to be simply powerful new tools in their toolbox – to be used when an appropriate situation arises, and then with care to architect the solution in a way that satisfies the needs of the user, not that caters to the abilities of the technology.