

## **Planning for RIA success**

*Tad Staley, Adobe Consulting*

[www.adobe.com](http://www.adobe.com)

**Table of Contents**

Planning for RIA success ..... 1

*Tad Staley, Adobe Consulting* [www.adobe.com](http://www.adobe.com)..... 1

1. Introducing rich Internet applications ..... 4

    Structure of this document ..... 4

    RIA background ..... 5

    The value of RIAs..... 5

    Additional resources ..... 6

2. Determining the appropriate RIA scenario..... 7

    Advantages of RIAs over HTML-based sites ..... 7

    Section summary ..... 14

3. Organizational focus: Skills required for RIAs ..... 15

    User experience design ..... 15

    Engineering..... 17

    Project management ..... 19

    Section summary ..... 20

4. Prototyping ..... 21

    Vision prototype ..... 22

    Technical proof of concept ..... 24

    Section summary ..... 26

5. Piloting the solution ..... 28

    Set objectives and measure the results ..... 29

Choosing the pilot application ..... 30

Section summary ..... 31

6. Assessing the impact of rich Internet applications ..... 33

## 1. Introducing rich Internet applications

Rich Internet applications are fast becoming an important differentiator in the area of web presence. RIAs combine the reach of the Internet with a rich and compelling user interface that provides greater level of satisfaction and success in a user's web-based interactions.

RIAs are also becoming increasingly popular and accepted. According to Gartner Group (<http://www.gartner.com/>), mainstream adoption and critical mass among IT and commercial software projects will occur by 2008, and at least 60 percent of new application development projects will include RIA technology by 2010.

Planning for success in rich Internet applications goes beyond writing code. At Adobe, we have found through consulting work that successful implementations require prior considerations related to the ultimate application users' technical and usage context, effective design principles, skills and infrastructure of the organization, and ongoing support considerations.

This paper focuses on factors to weigh when considering how, when, and where to get started in the design, development, and deployment of RIAs in your organization. We address considerations in the areas of application suitability, organizational readiness, impact on technical infrastructure, mitigating the risk and measuring rewards of an RIA.

### Structure of this document

This document addresses the various perspectives to be considered when undertaking the development of a rich Internet application; the sections are detailed below. This paper begins with some background on rich Internet applications, with pointers for additional information, and then lays out a step-by-step plan for RIA success.

1. **Introducing rich Internet applications:** This section gives background information on rich Internet applications and their business value.
2. **Determining the appropriate RIA scenario:** The first question for application planners is to determine of the type of application to build and under which conditions is it appropriate to build an RIA.

3. **Evaluating the organizational skills required for RIAs:** Adobe customers often want to know how the adoption of RIAs fits into their current organization and skills; and, to which extent they need to go outside the organization for resources, and where they can find these resources or training.
4. **Prototyping:** Adobe Consulting group has found that in many cases an effective way to get started with an RIA is to develop a “vision prototype.” The objective is to discover the value that can be realized, calibrate expectations on what is possible, and build interest (and funding) within the organization.
5. **Piloting the application:** Though some organizations plunge directly into building a large RIA project, Adobe Consulting offers a more measured approach that allows an organization to measure the value without the risk.
6. **Assessing the impact of Rich Internet Applications:** The project objectives should be associated with specific, quantifiable metrics, such as average time required to complete a task, or percentage of users who complete a specific task.

## **RIA background**

At Adobe, we coined the term “rich Internet application” in 2002 when we introduced a major upgrade to Adobe Flash Platform. Since then, we have enabled more RIA sites than any other organization through our products, consulting group and partner network. In 2006, with the introduction of Adobe Flex 2, a new generation of RIAs will be brought to market that are faster, more compelling and easier to build than ever.

RIAs do not replace websites. They work within the framework of an organization’s existing infrastructure to deliver better user experiences and functionality than sites that are solely HTML-based.

## **The value of RIAs**

Creating and delivering a superior user experience is not merely about making an interface that looks pretty. The look and feel of an application is important and helps retain the user’s focus, but the essence of great user experiences is the ease and efficiency with which the

user's goals are met. In this context, it is important to ensure that RIAs deliver measurable results.

- **Increased revenue:** more online transactions due to speed, task completion rate, and return. For example, when the Broadmoor resort ([www.broadmoor.com/](http://www.broadmoor.com/)) introduced its RIA reservation system, it doubled the conversion rate from browser to buyer.
- **Increased differentiation:** more intuitive, customized, and compelling access to services.
- **Engaging experience:** the narrative is beyond what simple HTML can provide. For example, Intelligent Finance ([www.if.com/](http://www.if.com/)) achieved a 15 percent increase in direct mortgage sales when they delivered an RIA calculator into the customer journey. The RIA explains the benefits of offset mortgaging to customers better than static HTML text.
- **Increased understanding:** from complex business information to understanding trends in statistics, RIAs help a user understand content in a visual and interactive way.
- **Reduced support costs:** more usable + fewer errors = fewer support calls. For example, when Yankee Candle ([www.yankeecandle.com/](http://www.yankeecandle.com/)) deployed its RIA for creating and ordering custom candles, they realized a 70 percent decrease in support calls to their call center.

### **Additional resources**

For overview information on rich Internet applications, including white papers and demos, see the Adobe rich Internet applications topic center at [www.adobe.com/devnet/ria/](http://www.adobe.com/devnet/ria/)

For more information on Adobe products and the Flex and Flash product lines, visit [www.adobe.com/software/](http://www.adobe.com/software/).

## **2. Determining the appropriate RIA scenario**

Rich Internet applications can provide a significantly improved user experience for many different types of user tasks. However, they may not be the ideal approach for every kind of Internet-related application. This section focuses on when to use RIAs, when to use HTML, and when a combination of the two fits the project best.

It is difficult to improve on browser-delivered HTML when the purpose of that page is consistent with the original intention of the web: text-centric information, flowed around static content (such as graphics), containing links to other pages of similarly structured information. In other words, for text-based, page-oriented information, standard HTML-related development and delivery environments are likely to be the appropriate choice.

There are even usability advantages to HTML in these cases, since millions of users are conditioned to interact with text-based content as rendered by the browser. These advantages include short-cuts such as Ctrl+F for finding text within the page, or highlighting text on the page for copying and later pasting elsewhere, or bookmarking the page based on the URL (web address) that appears in the browser. While all these experiences can be replicated in the Adobe Engagement Platform, they are built into the standard operating behavior of HTML-based websites.

### **Advantages of RIAs over HTML-based sites**

Despite the strengths of HTML, it's clear that not all information is best conveyed in text form. Looking at standard office automation suites such as Microsoft Office, we see that people interact with their information in a variety of ways, ranging from text to numbers to graphics and more.

In fact, it's often the case that while text-based content is essential to providing substance to a site, text is generally an ineffective way to provide a navigation framework, or data visualization and analysis capability. Furthermore, standard text-oriented HTML is static and does little to engage the user and make the experience exciting and more relevant.

RIAs provide particular user experience advantages for several application characteristics, however:

***RIA advantage 1: Context continuity***

One of the challenges of an HTML-based application is that it is page-based. When the user wants to view other information, or even look at the same information in a different way, standard HTML requires a new page, or a new version of the same page, to be loaded. No matter how fast the Internet connection is, or how efficiently the page is rendered, the fact is that there is a period of time—usually between 2 and 5 seconds—when the new page is loading and not fully available. This pause allows the user's attention to stray, and when the new page arrives, some re-orientation is required, however minor. If it were possible to graph the user's level of engagement with the application, the curve would be very jagged, spiking when the page loads and dropping as each new page is requested.

The advantage of an RIA is that these gaps in the user's engagement can be minimized or eliminated. Instead of loading new pages, an RIA can *reveal* content that has already been loaded—sometimes in the background and unnoticed by the user. The effect on the user is that their attention can remain consistently engaged, with many benefits:

- **Responsiveness:** When the application responds directly to a user's command, or when the user can directly manipulate elements on the screen, it engenders a feeling of connectedness and responsiveness to the application. The user trusts the application, which feels more like a solid-state machine than a harried librarian shuttling off to fetch the next volume at a user's request.
- **Productivity:** Not having to subtly re-orient themselves with each new page, users can optimize their focus and stay engaged with the tasks at hand. The feeling of a solid-state machine also empowers the user to explore more, without fear of losing the page-oriented thread or having to reload previous data-filled pages upon return.
- **User persistence:** The feeling of connectedness to the application, without page transitions that cause attention gaps, keeps users more committed throughout the

course of an application task. Each attention gap provides an opening for a user to shift his or her attention and move to a different task, application or site.

Regarding user persistence, there can be significant return on investment in developing an RIA simply by realizing an increased rate of task completion. The benefit should be measurable: We know of many companies, ranging from online employment services to insurance companies, who have measured the drop-off rate on an existing HTML site. The loss of revenue due to employers not getting through the cumbersome job posting process, and the increased cost of customer service when claims aren't successfully filed electronically, represent lost business value that an RIA can directly address.

### ***RIA advantage 2: Data-intensive web applications***

A web application that consumes large amounts of data is an excellent candidate for an RIA. A traditional HTML site that involves data must essentially deliver the data as static "snapshots" of the actual data that resides on the server. If the user wants to filter the data, or request a different data set, the application must go back to the server and create another snapshot. This process violates the principles of responsiveness, productivity, and user persistence mentioned earlier.

A rich Internet application provides a different experience. The original data remains on the server, of course, but the RIA can request data *in the background*, while the user focuses on the application itself. Moreover, with more business logic resident on the client, many operations can actually be done on the client, resulting in more responsive user interfaces.

### **Case Study: The perception of local data**

An online guide that creates television programs and schedules developed an RIA version of its channel guide. Their application required displaying 15 days of television programs, in 30-minute increments, across 600 channels. The content, stored on the server was more than 10 megabytes of data.

The original HTML application delivered *snapshots* of the data in two-hour increments across 600 channels (each downloaded page was about 180k). If a user were interested in the next

night's line-up, the application returned to the server to fetch a new page. The application was well engineered, so each subsequent page generally loaded in less than three seconds.

The RIA version of the application took a different approach, however. The application only immediately downloaded the data that would be visible to the user initially. This amounted to two hours of data across 14 channels (the user interface (UI) only displayed 14 channels simultaneously).

Once that screen had been rendered, and while the user oriented him- or herself to the data, the application—in the background and unbeknownst to the user—went back to the server to fetch the next most likely screen of data (surprisingly, the most typically requested next block was the *previous* night's line-up). From then, the application used idle processing time to fetch and cache the expansive program listing, so that when the user navigated to a new time slot, the response would seem immediate to the user.

### ***RIA advantage 3: Embedded rich media***

You have probably experienced the excitement of discovering that a favorite movie clip or song is available to be played directly on the web. This enthusiasm is quickly dampened by the need to answer several questions about bandwidth and media player version. After getting through these questions, the desired media usually appears in a separate window. This makes the entire experience awkward and less immersive (and therefore much easier to abandon).

You often don't notice it when you encounter a site with embedded media, however. It feels natural to have audio or video part of an integrated site experience. An example of this is Google video ([video.google.com](http://video.google.com)), where a user can play a video directly on the page.

The value of embedded rich media goes beyond entertainment. Many online businesses have discovered the improved customer experience of using rich media to provide in-context guidance, educational content, or simple orienting introductions. Consider the advantage of having audio or video tutorials *on the same page* as the loan application you are trying to fill out.

With the Adobe RIA platform, video content can actually interact with the application. We developed an application in which a video is available to explain the difference between setting up one-time and regular payments. The application was developed so that it highlights the areas of screen referred to in the video and *allows the user to interact with the page* while the video plays.

While the previously mentioned advantages of rich Internet applications are available with a variety of deployment environments, the embedded rich media capability is unique to the Adobe Engagement Platform ([www.adobe.com/devnet/platform/](http://www.adobe.com/devnet/platform/)). The platform has excellent rich media streaming capability, but the uniqueness lies in the Flash Player, which can either support a small application embedded in a portion of an HTML page, or serve as the delivery platform for a dedicated Flex-developed application.

The Adobe Engagement Platform is also a completely cross-platform video environment that goes beyond mere playback, like other solutions, to offer rich programmatic functionality to video objects and related functionality.

#### **Case Study: Improving customer service**

The Adobe Engagement Platform also enables *two-way* audio, video, or text communication. Recently a large investment company determined that its key online differentiator would be its ability to provide better, more responsive service to its customers. After considering a number of options, this company made the decision to evaluate the Adobe Engagement Platform. In fact, Adobe Consulting guided the company through the process outlined in this paper.

In this case, the company developed a proof-of-concept and prototype application that demonstrated the improved customer experience. Along with other important areas of RIA functionality, the prototype included real-time and in-context text messaging, as well as on-demand in-context video chat with a customer service representative. The application also included a form of screen sharing, so the customer service representative could see the customer's screen and provide better directions.

The prototype process was so successful that the company is in the process of defining the pilot phase of their RIA adoption. The rest of this paper will outline how to define and deliver these stages of RIA planning.

#### ***RIA advantage 4: Transitions and user guidance***

While it can no doubt be distracting to have too much motion in a website, our experience with designing and delivering usable and engaging applications indicates that discreet and purposeful motion is incredibly useful in providing context and guidance for the user.

Earlier in this paper, we discussed the concept of an RIA as a solid-state application, or the application as a predictable and whole *machine*. Rich Internet applications are oriented less to *pages* and more to *states*. When a user moves from one task to another, a well-designed RIA directs the user's focus appropriately, as if guiding him or her into another room in a house.

Imagine that you are using an RIA to study the financial performance of a company. Looking at a graph of the data, you notice that there was a news event associated with a rise in the company's stock price. In a traditional HTML application, you would click the news link and go to another page to read the story. An RIA, however, would likely provide access to the story in a way that would engender *context continuity* by discreetly restructuring the page to allow room for the news story to appear. Perhaps a reading pane would "grow" onto the page from the bottom border. When the user finished the story, he or she could close that reading pane, which would "shrink" out of sight.

The gradual transition to the new state, where the reading pane is displayed, allows the user to retain context. As described below, we refer to the design of these transitions as an application's "choreography." These transitions are far from gratuitous animation! Well-defined choreography is like walking the user into another room, while affording the user the certain knowledge on how to return to the room.

**Case Study: The value of an improved multi-step process**

One of the original RIA applications was a series of online hotel reservation systems produced by iHotelier ([www.travelclick.net/solutions/IHotelier.cfm](http://www.travelclick.net/solutions/IHotelier.cfm)). This application, called OneScreen, took what is traditionally a multi-step, five-page reservation process and reduced it to a single screen. In the earlier HTML version of the reservation system, each new page required the user to wait and lose the context and content of the previous page. These gaps in continuity significantly increased the chance that a user will abandon the process altogether.

By contrast, the OneScreen application presents hotel information in three sections on a single screen that does not force the user to follow a uncompromising sequence. The three screens include:

- An interactive calendar that displays room availability and rate information. The user can choose and change dates directly on the calendar to set their arrival and departure dates.
- A room information panel that displays information about each room type, including rates, a picture, and a description of the selected room type. Availability of the room type selected in this panel displays in the calendar panel.
- Finally, a billing panel displays an interactive summary of the total cost of the accommodations, including taxes, as well as a credit card payment form that the user fills out and submits to complete the transaction

The results of this dramatically improved multi-step process are impressive. Bluegreen Vacation Rentals adopted the OneScreen application and discovered:

- A 20 percent increase in the number of reservations
- A 50 percent increase in revenue booked

Many businesses, from insurance companies to travel sites, require users to go through a multi-step process to complete a transaction. Increasing the users' success rate through the use of an intuitive and effective RIA can result in significant business value.

## Section summary

Adobe Consulting has worked with companies on a wide variety of rich Internet application types, ranging from small “widgets” that are embedded in a larger HTML page (such as a stock quote lookup) to comprehensive ERP front ends. Our experience indicates that there are several application criteria that indicate you should consider an RIA implementation.

HTML is still a very viable web deployment medium, and is particularly useful for the following usage types:

- For quick access to large quantities of text and image content.
- For simple, uniform user interaction.
- When navigational integration with the web is critical.

Meanwhile, RIAs are useful in the following situations:

- When the user’s goals require more than a page metaphor. RIA delivers:
  - Single screen
  - Direct manipulation
  - Immediate rich feedback
- When there is time and commitment to iterate on the interaction design.
- When customer engagement, loyalty, and satisfaction are at stake.
- When user interface design is a key differentiator.

### 3. Organizational focus: Skills required for RIAs

An important factor in the decision of where and when to pursue an RIA strategy lies in the human capital of the organization and beyond: Will resources be available to design, implement, and maintain the solution effectively? This section identifies the skills typically required in supporting a rich Internet application.

The good news is that the design and development of rich Internet applications generally can leverage existing skills in a mature IT organization. The skills required to develop traditional client-server or HTML-based applications can be still be useful, though an additional dimension is often required.

Like most full software development projects, standard disciplines are applied in an RIA project: business analysis, design, engineering, and project management.

#### User experience design

The heart of a Rich Internet Application is its design. In one way or another, the advantages of an RIA discussed earlier all center around an improved user experience, which will drive improved user performance, productivity and satisfaction. These benefits obviously don't happen automatically, but result from well-supported design practice.

This means that design needs to be given enough attention and funding on a project. Many organizations rush through the design, or begin development before design is complete, which can undermine effective user experience. Our experience in Adobe Consulting is that good design generally comprises at least 25% of the overall project budget; *great* design requires 40% or greater.

Furthermore, designing great user experiences is not about making an application attractive. In the words of the Adobe Experience Design (XD) team, user experience design should be done *in service of surfacing capability, improving task completion, and making the application enjoyable.*

There are typically three or four roles used in the design of an RIA. To best understand these roles, it would be instructive to consider the process of RIA design as similar to making a movie. In fact, good design tells a story, just like a movie, and often guides a user through a set of tasks as if s/he were following a narrative.

The following roles are common in RIA design process:

- User experience lead: Like a movie director, this person ties together the structure and behavior of an application into a well-orchestrated experience. This person focuses on consistent use of visual cues, a coordinated palette, appropriate pacing in the transitions and other behaviors, and a rational navigation structure. The overall focus is to develop a sense of connectedness to and trust in the application.
- Information architect (IA): This is a traditional role in web design, and remains a cornerstone of the RIA design process. The IA is responsible for mapping business requirements to application *schematics*. However, the metaphor is different—no longer oriented around static pages, but instead states of a single, unified application. This adds an extra dimension to the structure of an application, since RIA content is often revealed within an existing screen rather than requiring loading a new page.
- Visual designer: This person is responsible for the look and feel of the application, just as in a typical HTML site. However, an RIA adds a new dimension which can be called *behavior*—that is, the transitions within a single screen application of the UI components in response to user gestures. In addition to producing pixel-level design specifications, the visual designer produces choreography specifications that define the application's behavior.
- Design technologist: The role of *design technologist* can be played by either the information architect or the visual designer. This role carries on the design work into the implementation phase of a project. The design technologist is responsible for ensuring that the development team adheres to the specifications, and then actually facilitates that process by being part of the production process in developing the

code for skinning and styling, as well as the creation of graphical assets and some visual components.

### ***Web design vs. RIA design workflow***

Many organizations have developed a well-defined process through which web designers work with web developers. This typically involves having the designers develop static HTML versions of the application, with all the associated graphic assets, styles and colors. The developers then take these pages and develop the code required to for them to be generated and interact with the web server.

The process may not be so different when implementing an RIA. RIA designers in many Flex development organizations actually deliver their specifications and layout in code, often using Flex Builder. This requires familiarity with the development environment—MXML and ActionScript in particular—but this is no more difficult than learning HTML and JavaScript, which most web designers have mastered.

Other organizations simply deliver pixel-level specifications (usually as Photoshop or some other graphical file format) and rely on the developers to translate those specs into working code. In this case, the design technologist can play an important role in the translation.

### **Engineering**

In terms of role definitions, the engineering assignments on an RIA look very similar to traditional web or client-server development efforts. There is still the need for system architect(s), as well as developers focused on the database, the middle-tier (or business logic) and the presentation tier.

On a Flex project, the database and middle-tier development assignments may look very close to the tasks required in web development. The database still needs to be designed, and access methods such as stored procedures will generally still be required. The middle-tier may consist of Enterprise Java Beans (EJBs), plain old Java objects (POJOs) or some other means for servicing the client interactions (Cold Fusion, for instance). So much of an existing organization's technical skills can be leveraged.

Skills required for the client-side development depend on the chosen RIA environment. Typical roles in the implementation of Flex applications include application architect as well as module and specialized component developers. There can be significant client-side business logic development in a Flex application, which will be familiar to classic software engineers, especially those comfortable with object-oriented programming. In our experience, Java developers are very comfortable leveraging their development skills in RIA development.

***What background and skills are required to learn Flex development?***

Flex was designed for professional application developers, and developers familiar with Java, C# and other OO languages typically learn Flex quickly. The Flex development model, in fact, encourages object-oriented software design, and the separation of applications into an industry-standard MVC (Model-View-Controller) architecture. Developers who are familiar with other application development technologies, such as MFC, Motif, Swing, and AWT, very naturally take to the event-driven and component-development model of Flex.

Though a Flex application uses Flash Player as a runtime environment, experience with the Flash authoring environment is not at all required. In fact, many techniques in Flash authoring, such as management of the application *timeline*, are no longer relevant in Flex.

At Adobe Consulting, when we hire new Flex developers, our standard job description includes a variation of the following requirements:

- *N* years of software development experience designing and implementing complex Internet applications.
- Solid experience in web programming environments, such as Java, or C#.
- A strong understanding of building object-oriented, reusable solutions.
- Solid understanding of Java server technologies and Internet technologies, including J2EE and web services.
- Experience with a declarative programming model, such as JSP or ASP.NET, is a plus.

## Project management

Although this paper will not focus on project management principles, it is useful to acknowledge that some project management techniques are more conducive to successful RIAs. In particular, because RIAs are generally creating new user experiences, it is more difficult to build complete predictability into the plans. Or, stated another way, if clear definition of milestones and deliverables are required, it's possible that some innovation in the design will be compromised.

Adobe Consulting has standardized on a delivery methodology steeped in the development and project management practices of Agile development. Not only do agile methods advocate a tight coupling in the relationship between the implementation team and the customer, but we have been able to extend our agile delivery methodology to embrace the value of user-experience design throughout the stages of realization of a rich Internet application. Agile approaches embrace change throughout the delivery cycle, offering a great deal of flexibility in the delivery of the user experience—whether that is through changing customer requirements as the vision unfolds, incorporating the findings of usability testing, or reprioritizing of features—while ensuring that a product is delivered within scope and budget that is maximally useful, usable, and desirable to the user.

We have observed projects in which, without sufficient communication between design and development, the scope of the effort increases needlessly. For example, Adobe Consulting was asked to support the implementation of automotive configurator. The designers had created an elegant vision for the project, but had done so without consideration for the cost. An accordion control, for example, was designed to have more than one pane open simultaneously; while this is feasible, it was unnecessary for the user experience and the custom component added a few days to timeline of the project.

One reason the *Agile* process is effective is that project teams often sit in one large room, and there are short daily checkpoints from each project member, thus ensuring that no one gets too far out of alignment. It is also worth mentioning that the *Agile* approach of pair-

programming is an excellent means of ensuring quality, as well as an excellent medium for mentoring and performing knowledge transfer.

### **Section summary**

The skills required for RIA development do not differ radically from the skills generally found in professional IT organizations. Adobe Flex was explicitly conceived and developed to leverage existing industry expertise. We have observed experienced web and application developers move very quickly up the Flex learning curve.

The design skills are also consistent with web development, with two important distinctions. Beyond the structure and appearance of an application, the third dimension introduced by RIAs is *behavior*. Familiarity with this added dimension is essential to creating great RIA experiences.

Perhaps the most important added element in RIA design is a change in mindset. Web designers are accustomed to think in terms of pages, but an effective RIA has a different usage model that resembles not the loosely coupled pages in a book, but the well-coordinated functionality of a solid-state machine. Like the new application model, the project manager must orchestrate the interactions between team members so that all are parts of a well-coordinated effort.

## 4. Prototyping

Organizations generally seek validation of the RIA approach prior to making a larger commitment to new initiatives. There are typically two aspects to this validation that can be addressed together or separately.

Essentially, each form of validation is directed to a distinct aspect of the organization, business or technical:

- To ensure the validity of an RIA for the business and gain support from management, organizations often create what can be called a *vision prototype*. This phase precedes the rigorous design discipline of a fully defined project; instead, the vision prototype is used to demonstrate the potential of an RIA within the context of an organization's business priorities.
- To satisfy the development group so that the RIA can interoperate well within the established infrastructure and standards, many organizations commit to doing a *technical proof of concept*. This often takes the form of one or more simple applications, without any design applied, that demonstrate the actual deployment of the RIA within the target environment.

Each of these points will be discussed below, with attention to conditions that would indicate when each is appropriate and examples of how they've been used in the past.

One important note: the two types of validation are often, in fact, joined in a single project. That is, the *vision prototype* is designed in such a way that it also accesses "real" data, and is deployed in the actual environment, or a close facsimile.

Though it sounds like a logical approach to bring together both forms of validation in a single project, we often discourage this approach. The reason is that the objectives of each portion are distinct, and the process of meeting more than one objective with a single stream of work adds to the complexity of the project and dilutes focus.

The other reason we tend to avoid this approach where possible is that the delivered application can be very close to operating like a real application. However, as we will elaborate below, the resulting application will have been neither thoroughly designed nor fully architected, so it is unlikely to be a sustainable and viable foundation.

## **Vision prototype**

There are many reasons to consider prototyping a solution before committing to a larger effort, but a key consideration is that a prototype can help establish expectations on design, functionality, and the resulting requirements of an application. This is particularly relevant to RIAs, because in many cases, the team will be venturing into unfamiliar terrain.

Because of this uncertainty, traditional software development project techniques can be difficult to apply. A certain amount of *discovery* is required in the RIA process. Put another way, user experience design is of particular importance in an RIA, and it is often difficult to predict the outcome of an iterative design process.

## **Case study: Prototyping at Alpha Securities**

In early 2005, the decision was made by senior executives at Alpha Securities (fictional name), a large investment firm to radically improve its presence on the web by improving its customers' user experience.

The Adobe Engagement platform was considered by Alpha Securities to be one of the most promising vehicles for this new level of customer engagement. However, to that point Alpha Securities did not have significant experience in designing and developing using the Adobe platform; there were some Flash designers in the organization, but they had little experience with a more comprehensive design framework.

Alpha Securities engaged Adobe Consulting to work closely with them to create a vision of what was possible. A few key factors enabled this: time, money and resources.

- Alpha Securities took the investigation seriously enough to provide adequate funding. The cost was not prohibitive, but by allocating funds to investigate the possibilities,

Alpha Securities ensured that the appropriate attention and expertise would be applied to the effort.

- Alpha Securities assigned a senior executive to oversee the process. This person was very familiar with the organization and its priorities, and also was a savvy student of trends in Internet designs and application deployment.
- Finally, Alpha Securities understood that the evaluation process would take some time. In fact, it took only about four months before the platform and approach were validated, but it's important to acknowledge that Alpha Securities was willing to invest the time, to go through the full evaluation process, before making its informed decision.

The first deliverable in this phase—a pure *vision prototype*—took about six weeks to complete. With a more linear and directed process, the design and prototype development could have been done more quickly, but the project was purposefully granted freedom to experiment. Following the directives of the visionary executive, the project worked through a variety of possibilities, but was always measured by a set of business objectives and design principles.

The resulting application included a variety of functionality and content, and certainly not something Alpha Securities would implement as delivered. However, the layout was clean and elegant; the transitions (or “choreography”) were compelling and demonstrated that this was not a typical web application. The content and functionality was chosen to address various aspects of the company's strategic direction. This was all wrapped up and delivered in a portal framework that itself suggested a longer-term vision for how modules could coexist and each deliver a compelling experience.

The *vision prototype* was a tremendous success. It was demonstrated throughout the organization, all the way up to the chairman, who himself was the original catalyst for the new web initiative.

The design and functionality changed peoples' thinking within the organization. We often heard people comment, after seeing the prototype, "wouldn't it be great if we could take this and add..." In other words, the prototype succeeded in tangibly informing various stakeholders in the organization about the potential of the platform, so much so that they could visualize extensions that applied to their own functional needs.

### **Technical proof of concept**

Of course, it would be irresponsible to get your organization riled about the possibilities inherent in an RIA platform without quietly taking steps to assure the technical feasibility of such a platform. Cautious organizations will do this step prior to committing to any vision effort; many create a parallel technical stream so that each can inform the other—without, as mentioned above, constraining or altering the project's objectives.

The technical proof of concept (TPOC) is often a smaller effort than the vision prototype, though we've seen companies take months to prove to themselves the viability of an RIA platform. The TPOC is best performed when, before starting, a distinct set of validation points are established. Then sample applications are developed to prove those specific points. Here are some examples we've seen of technical conditions to be tested:

- The ability of the platform to access third-party content through SOAP web services
- The performance of downloading an image file to be used in a product configurator
- Time required to scroll down a TV channel listing of 600 cable stations
- Ability to display 150 updates per second from a real-time data feed
- Displaying and selecting from a palette of over 1,000 colors

In agile terms, these TPOCs are sometimes referred to as "spikes." The metaphor is derived by the concept of driving a spike from the front to the back of the problem to prove that it can be solved. Solving a thin slice of the problem demonstrates that the larger context is achievable also.

The key in this process is to make the tests themselves as small as possible, and measurable as well.

Sometimes, the TPOC is delivered not by developing a sample application, but by delivering a written description of the solution. This process involves the RIA architect spending time learning the particular infrastructure and standards of an organization, and then writing and presenting an architectural white paper that establishes the interoperability.

Alpha Securities performed their TPOC efforts in parallel, so that when there was general agreement on the platform from a design perspective, the development organization could simultaneously affirm the feasibility.

Often the TPOC track is simply there to keep the visioning process grounded in reality. This is sort of like a technical chaperone at the gala vision dance.

#### **Case study: Technical proof of concept**

A large international marketing services firm was looking to develop and deploy a distributed front-end to its end-to-end ERP system, with top-level modules including estimating, ordering, scheduling, project tracking, costing, shipping and billing. The initial phase of the project would be internally facing, allowing their back office employees a single, unified user interface for accessing the seven modules of the backend system. Subsequent phases would allow customers to access the system to manage their accounts, projects and artifacts.

The application was initially conceived as being developed and deployed on .NET, however, once Microsoft announced its intention to discontinue support for Internet Explorer on the Macintosh platform, the company became concerned about the long-term accessibility of the system. As a marketing services firm that relies on creation and publishing graphical artifacts, many of its customers and partners used Macs.

The company's implementation team identified several issues and limitations of the .NET environment for applications to be delivered to a cross-platform user base. Before

committing to the Flash-based RIA platform, the technical team performed a series of tests to ensure that the key requirements could be met.

The technical proofs of concept were developed to understand the technical environment and therefore were not elegant; no design effort was taken to lay out these test applications. The tests fell into two parts:

- **Front-end controls and memory management:** Because this would be the user interface to a comprehensive enterprise system, thousands of UI elements across over 200 screens would be defined across the seven application modules. This portion of the TPOC process was to evaluate the platform's ability to defer the instantiation of these objects until they were needed. Also evaluated was the platform's ability to unload objects from memory when no longer in use.
- **Back-end integration:** The initial design called for data to be stored in SQL Server. The database design had been completed with a sophisticated array of stored procedures developed to extract and insert the application data. This portion of the TPOC process focused on the ability of the Flash-based platform to leverage the existing database design and provide adequate performance.

In this case, the proof-of-concept evaluation went well. In the process, however, the team learned a great deal about effective alternatives for optimizing the programmatic control of the user and data environment.

### **Section summary**

We strongly recommend considering both a technical and design evaluation of an RIA platform before committing to a large implementation effort. Think of it as working out on a practice field before a game: The process will help shape the final application and also provide insight into effective implementation approaches.

At Adobe Consulting, we have seen cases where this kind of preliminary exercise led to a different approach. As noted in the first section, RIA may not be right for every project. To ensure success with an RIA, there must be demonstrable return on the investment, a

person to champion the vision (generally a business owner or user representative), and a technical team confident and enthusiastic to support the application.

The preliminary efforts of creating vision prototypes and technical proofs of concept go a long way to securing this level of support within an organization.

## 5. Piloting the solution

Adobe Consulting generally suggests planning on an initial engagement that allows an organization to better understand the process and opportunities involved in RIA initiatives. The primary metric in choosing such a pilot project is to keep the value-to-risk ratio as high as possible.

Pilot applications are often chosen so that the value-to-risk ratio is high. This can be done by focusing on maximizing application value or choosing low-risk projects, or both.

One low-risk approach we've employed is to deliver a pilot application wherein the RIA version is redundant to existing functionality that will continue to be available. This approach is particularly compelling if an application-monitoring process is established, because it allows an opportunity to explicitly measure the added difference in value between the RIA and a traditional solution.

For example, a large insurance company developed a pilot RIA application to allow glass damage claims to be submitted through the web. The pilot application was launched in only two states. This allowed the company the opportunity to measure the success rates of the RIA submissions. Their tests showed a dramatically higher success rate using the RIA application, and the result helped drive additional product design and development priorities.

Another way to pilot an RIA application that keeps risk low yet still derives benefit is to design and develop an application that resides within a complete HTML page. We sometimes refer to these small applications as "Embedded RIAs": components with a limited but specific functionality that can be deployed within an existing HTML site in one or more places. Examples of this could include a stock quote lookup tool, a mortgage calculator, or a graphical interest rate lookup.

For example, a Wall Street financial services firm last year developed a simple interactive graph of the market, one that allowed users to select and compare specific sectors. The embedded application was relatively small, so was easily included in several areas of their

site. The pilot process was useful for them to better understand the development and deployment process. It was also a relatively low risk, since, if there had been any problems, the application could have been pulled from the site and replaced with the older, static graphic.

## **Set objectives and measure the results**

When we deliver pilot applications, we identify the various objectives for the application, and find metrics to measure the results. These metrics may be as qualitative as usability results, but as concrete as the percent of users who complete a task. The objectives generally fall into three areas (Note: this is not specific to RIA applications, though the examples provided are from RIAs).

1. **Business goals:** The ultimate value of an RIA should be measured in business terms. Examples of some business metrics that we have seen include:
  - o Increase awareness and access of specific functionality. Simply measuring user clicks in an RIA is sometimes sufficient. Be aware that measuring RIA activity may require different tools: for one thing, the RIA will likely be a single screen application or widget, in which case a typical page count will not be an accurate measure.
  - o Increase conversion rates.
  - o Increase effectiveness of cross-sells or average number of product holdings.
  - o Decrease number of call-center calls or average call handling time.
  - o Improve the qualitative user experience of visitors. This is generally measured in focus groups or some other form of follow-up interview or survey.
2. **End-user goals:** The intended benefits for end users include:
  - o The interface will be straightforward, intuitive, and easy to use. Typical usability techniques are used to measure effectiveness.

- Task efficiency: time and effort to complete tasks should be decreased in an RIA.
  - User preference and satisfaction: following simple A/B testing, users choose which interface they are more comfortable using.
3. **Technical goals:** A pilot often has several technical goals, including:
- Test support for more sophisticated client-side validation rules.
  - Evaluate integration with the existing Java J2EE application server platform.
  - Reduce server processing and network bandwidth utilization.
  - Application performance is often a primary focus. RIAs often perform complex tasks, or consume significant amounts of data, so application performance requires evaluation.

On this last point, note that simple A/B testing is generally not sufficient when comparing HTML sites to RIAs. On the TV program listing application mentioned above, the 150K HTML page was generated and loaded fairly quickly, actually faster than the 200K Flash-based application. However, each subsequent HTML click loaded a complete new page, while the Flash-based application incrementally loaded new data without replacing UI artifacts. Across a user's typical session, the Flash application was more responsive.

### **Choosing the pilot application**

Obviously, the appropriate pilot application depends on many factors and the objectives defined in the previous section. However, other generic factors are worth considering as well.

Here are some of the criteria we've observed as organizations choose pilot applications for their first RIA project:

- Most organizations look for an application that is relatively small in scope and relatively straightforward, so that it can realistically be designed and developed in a six to ten week period.

- It's useful to consider a pilot application that is an alternative for a hard-to-use static HTML interface, especially one that is having a negative impact on the business. This approach would allow the benefit of the pilot application to be assessed by comparison, and then, potentially used as the basis to replace the existing static HTML application. This point leads to the next consideration:
- Where possible, choose an application area that could provide a long-term business win. This could be in the form of simply replacing an existing underperforming application. However, the pilot RIA application could be the foundation from which other application areas could be evolved and improved.
- Make sure your pilot application allows for real user interface innovation. Replacing a text-centric HTML application with a Flex application will not deliver significant ROI. However, complex applications with multiple view states and/or lots of data would be ideal candidates. The user interface innovations here will not only inform your organization about what is possible with RIA, but could set standards for your next generation of user interfaces.

One other point to consider: As mentioned above, it's useful to consider a pilot application that, if successful, could provide a clear lead-in to a larger project. However, it's also imperative to learn from the pilot application and make improvements or optimizations when moving to the production application. Usability and performance testing are critical to ensure the success of an RIA, and a pilot application is an excellent opportunity to test and further evolve the user experience.

### **Section summary**

Rich Internet applications are not a passing fad. They represent the heart and soul of the next wave of Internet applications and presence. Astute organizations should look to evaluate how RIAs can add business value to their web presence.

The evaluation process should be rigorous and well-planned. It includes funding, adequate resources, iterative refinement and patience. Establishing a method for piloting applications is an important part of understanding this new environment and ensuring its success.

## **6. Assessing the impact of rich Internet applications**

As stated earlier, the key to success in planning for RIA success is defining your objectives and measuring the results. This should not be informal but a process that is rigorously followed to ensure value is derived and understood.

To avoid ambiguity and the need for interpretation, the project objectives should be associated with specific, quantifiable metrics, such as average time required to complete a task, or percentage of users who complete a specific task.

One customer held an all-day off-site session to assess a three-month pilot application and make recommendations on ways to improve the process. What was particularly useful was the honesty with which they assessed the process and their willingness to confront the inefficiencies. For example, they realized there was not a sufficiently well-defined decision process, especially with respect to design. This led to seemingly endless debates over trivial design details and ultimately slowed down the project.

Similarly, the technical team performed an in-depth analysis of the capabilities of Flex as compared to alternative RIA approaches. This analysis was done over a range of areas, including performance, integration, development environment, ease of debugging, and code integration. In each case, dealing with specific metrics rather than opinions allowed the technical team to assess their experience.

The experience of designing, developing, and deploying Flex-based RIAs should not be undertaken lightly. Adobe Flex is a rich and sophisticated RIA environment and should be evaluated seriously. However, as hundreds of customers have discovered, the results for the business, the user, and the technical community can be astounding.