

User Interface Engineering
242 Neck Road
Bradford, MA 01835
(978) 374-8300
(800) 588-9855
www.ue.com

User Experience White Paper

Enhancing User Interaction in Pet Market

By **Will Schroeder,**
Christine Perfetti, and
Jared M. Spool

*A usability comparison of blueprint
applications built with Macromedia MX,
Sun Java™, and Microsoft .NET*



UIE Reports and White Papers

For more than six years, User Interface Engineering has conducted innovative research on making web sites as usable as they can possibly be. Our research team has spent hundreds of hours observing real people using real sites, trying to understand why some sites perform better than others do.

For more information on our research, or to download other white papers, visit <http://www.uie.com>.

Enhancing User Interaction in Pet Market

By Will Schroeder, Christine Perfetti, Jared M. Spool

Published by User Interface Engineering
242 Neck Road
Bradford, MA 01835
(978) 374-8300
uie@uie.com

Copyright © 2002, User Interface Engineering

Copyright laws protect this white paper. Duplication is not permitted. Macromedia® Flash™, ColdFusion® Server, and JRun™ are trademarks or registered trademarks of Macromedia, Inc. Sun, Sun Microsystems, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. Microsoft and .NET are registered trademarks of Microsoft Corporation. Third party trademarks, trade names, product names and logos, contained in this reports may be the trademarks or registered trademarks of their respective owners.

Companies Mentioned in this Report

Macromedia, Inc.
www.macromedia.com

Sun Microsystems, Inc
www.sun.com

Microsoft Corporation
www.microsoft.com

10987654321

Enhancing User Interaction in Pet Market

A blueprint application demonstrates how Macromedia™ Flash MX will change the way developers think about building web-based applications that are effective and remarkably easy to use.

In May 2001, Sun Microsystems released Java Pet Store, their first Java™ Blueprint application. (<http://developer.java.sun.com/developer/releases/petstore>) At the time of the release, Sun didn't intend to enter into the growing online pet business. Rather, they built the application "to illustrate the best practices for developing and deploying applications using the J2EE platform."

The pet store application had all the necessary elements for a successful e-commerce site, including a shopping cart, inventory system, product descriptions, and credit card validation. Sun wanted developers to use the source code and documentation as a basis for building their own sophisticated Java applications.

In November 2001, Microsoft ported the Java Pet Store to its .NET architecture. The purpose of the .NET Pet Shop was "to take Sun's primary J2EE blueprint application, the Sun Java Pet Store, and implement the same functionality using Microsoft .NET. Using the .NET version of Sun's J2EE best-practice sample application, customers can directly compare Microsoft .NET to J2EE-based application servers across a variety of fronts."¹

The authors of these applications never intended them to act as real e-commerce sites. Instead, their intention was to get developers thinking. Much like an auto manufacturer's concept car or a real estate developer's model home, the applications gave developers a way to visualize how similar applications *could* be built using the J2EE and .NET platforms.

Because the developers intended to highlight the architecture and code of the applications, they created very simple user interfaces. These interfaces performed as many e-commerce sites do, based on the standard "click-for-new-page" model that is the hallmark of implementations which rely on loading a new HTML document for each page.

In June 2002, Macromedia, on the heels of the introduction of its MX family of products (including Macromedia Flash™ MX and ColdFusion MX), took the concept of an online pet store one step further by enhancing the front end with a rich client interface, taking full advantage of the capabilities of Macromedia Flash. Macromedia Pet Market application defines a vision of what Rich Internet Applications should look (and act) like.

¹ *Implementing Sun® Microsystem's Java™ Pet Store J2EETM Blueprint Application using Microsoft® .NET*, Microsoft, November 2001.

(<http://www.macromedia.com/desdev/mx/blueprint>) According to Macromedia, “Rich Internet Applications extend the web and HTML without replacing them, enabling the development of applications that offer significantly more intuitive, responsive, and effective user experiences.”

When we compared the user experiences of the Macromedia Flash-based Pet Market to Java Pet Store and the .NET Pet Shop, we found that comparing the three applications clearly demonstrated the inherent advantages of Rich Internet Applications over “page by page” designs for web applications. We also found that certain qualities associated with Rich Internet Applications dramatically enhanced the user experience and created a more intuitive web interaction.

All three applications demonstrate the strength and flexibility needed to become serious contenders in the web application space. However, *the Macromedia Flash-based Pet Market stands out. It demonstrates that building web-based applications in Flash is as easy as in the Sun and Microsoft environments. More importantly, it shows us that designers have the technology today to create effective, usable, smooth, and attractive web-based applications.*

In this white paper, we will examine the qualities of the three applications as well as the development effort involved in creating these types of applications in each environment.

Qualities of Usable Designs

Macromedia Flash-based Pet Market, Java Pet Store and .NET Pet Shop all include the basic functionality to complete a step-by-step shopping process. Pet Market demonstrates additional unique qualities that support smooth and error-free interaction.

The qualities we examine in this white paper (which are all characteristic of Rich Internet Applications) include:

Continuity

Rich Internet Applications allow smooth progress through the stages of a process while keeping the user’s focus. Every time customers are slowed or interrupted by a wait for a response, or a jump from one point in the site to another, they may lose concentration, data, or interest.

The “Big Picture”

If the design allows customers to see all aspects of the interactive experience at the same time, users can choose the path they want to follow and will know what content is available and what to expect.

Control Of Screen Real Estate

Maintaining the “big picture” is much easier if designers manage the screen area creatively. For example, an effective application might expand key design elements when they have the user’s focus and contract them when the user moves on.

Control Of Focus And Context

Designers must walk a fine line when they set out to guide users. Users should be able to go anywhere that makes sense without getting needlessly lost. Enforcing a user’s focus by modal control of dialogs accomplishes this “guidance,” preventing obvious errors without restricting the freedom of reasonable choice.

Responsiveness

There are steps in the shopping process (such as checking the cart) during which customers appreciate a quick answer, while for others (such as when switching interest from one type of pet to another) they are more willing to wait as long as their destination is clear and there is indication of progress. The “feel” of an application is improved by sub-second response times.

Self-explanation

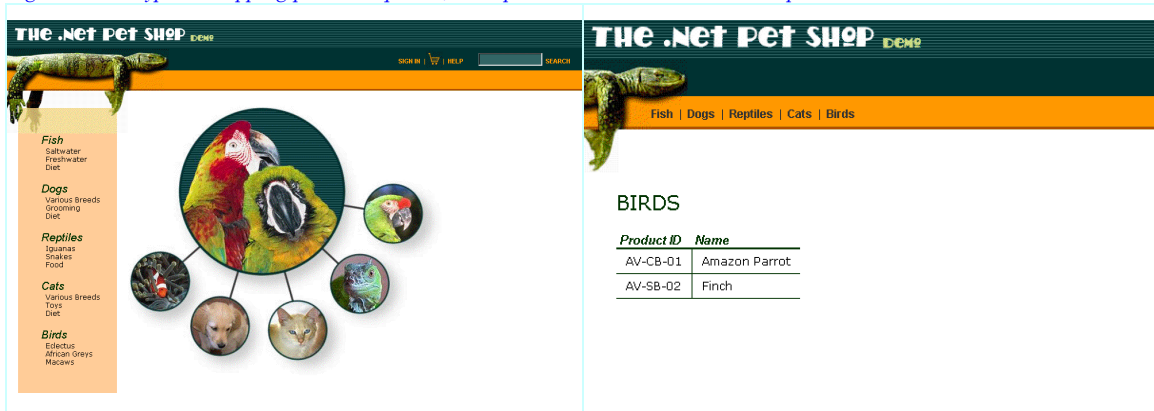
Whenever possible, the interface’s design should explain the purpose and result of actions and changes. For example, when the layout changes, customers should know where to find the design elements they had been interacting with.

Continuity and the “Big Picture”

Web applications involve users completing a process, or a series of steps, to accomplish a goal, whether that be the acquisition of information or the purchase of a specific item. Because of the history of HTML as a language designed for structured documents, the usual web site shopping process is a series of disconnected steps.

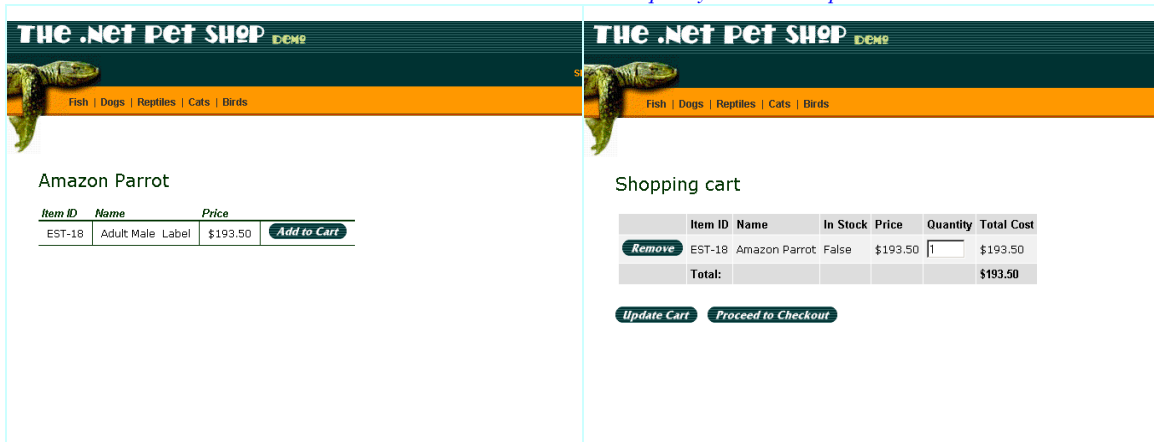
With standard HTML, customers must view multiple screens before completing the

Figure 1: The typical shopping process sequence, as implemented in the .NET Pet Shop.



Step 1: The user begins on the home page with a choice of pet “types” and chooses a “store”—in this case a category of pets.

Step 2: The user then reaches the “bird” store page. In a real site, this page might offer groups such as “Songbirds,” “Talking Birds,” etc. to manage a larger inventory, which would then require yet another step.



Step 3: The user learns more about a particular parrot, and has an opportunity to purchase.

Step 4: At this point, the user must select the shopping cart and load another page to complete the transaction.

shopping process. As they navigate through the product hierarchy, they view and make one choice per page, losing sight of the options at the previous level when passing on to the next. (Unfortunately, the page-by-page format, a throwback to days of “green screens,” has become the *de facto* standard for e-commerce.)

There are several characteristic problems and costs involved when continuity is broken. Discontinuity leads to user errors. Users lose concentration when the design forces them to jump from one part of a web page to another, or when an application displays a different version or layout of a page when content changes. These problems can disrupt a user’s thought process and can cause something to fall off the user’s mental “to do” list. These jarring designs disorient the user as they lose their understanding of the page’s spatial organization.

When the Java Pet Store application communicates with the server, users must wait and watch while a new page is loaded with the server’s answer. After each step in the process, the user must sit and wait while the browser retrieves a new page, because control remains on the server side, and the only way the server side communicates with the client side is by serving updated pages. It’s like shopping in a supermarket and adding items to a cart located at the registers at the front of the store; the customer must leave his or her shopping to look at the items in the cart. (Figure 1)

This disjointed view of the catalog and the shopping cart presents the developers with a challenge. Although users can reach each pet portion of the catalog with one link from each page, they can only reach the next level down (type of bird in this instance) from step 2—the bird “store” page, and this information cannot be viewed beyond this page. This can make browsing birds awkward. The catalog is like an iceberg: most of the products are invisible to the user at any given point. (Designers can sometimes solve this

Figure 2: The traditional “page by page” design, as demonstrated by Sun’s Java Pet Store.



Step 1: The user begins on the home page with a choice of pet “types.”

Step 2: The user reaches the “bird” store page. As in the .Net Pet Shop, a realistically sized inventory would require an additional page.



Step 3: The user learns more about a parrot, and has an opportunity to purchase.

Step 4: The user sees the cart and verifies that a correct purchase has been made.

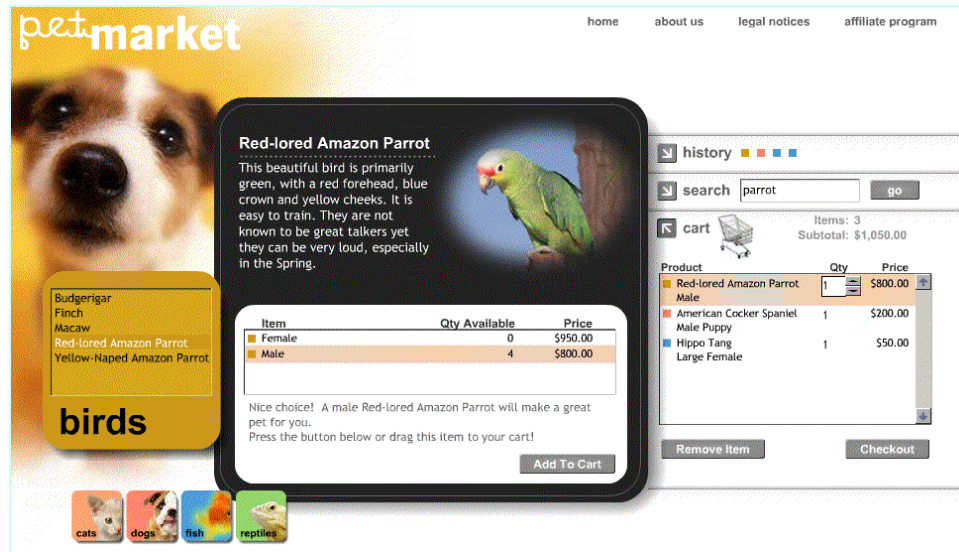


Figure 3: Macromedia Pet Market shows all elements of the purchase process on the same screen, enabling users to see the “big picture” of the application.

problem by adding links to the product page that connect it to other product pages, which can lead to a cluttered appearance.)

The Sun Java Pet Store architecture is almost identical to the Microsoft .NET Pet Shop, because it is operating under the same client-side constraints. (Figure 2)

Developers can circumvent this behavior by shipping relatively large amounts of data and pictures (covering all possible customer responses) within a single page. For an organization with a substantial number of products (SKUs), this translates into unacceptable download times, complicated development tasks, and heavy burdens for the servers.

Communication without serving new pages makes it easy for an application to use a “big picture” paradigm for the shopping process, with all levels of the product hierarchy and all steps of the decision process visible and functional within the same view.

The developers of Macromedia Pet Market have implemented a “big picture” display. Users can choose the order in which they would like to complete each step in the shopping process and lets users see all steps at the same time on one display. (Figure 3)

As users interact with the Pet Market application, there are no awkward pauses while one page disappears and another replaces it. A user can move from point to point in the process in a non-linear fashion. The elements may change size and aspect as the customer engages them, leading the customer to discover the store’s functionality.

In the Pet Market paradigm, the whole experience is always visible. In contrast, the Microsoft .NET Pet Shop and Java Pet Store paradigms rely on reading and understanding link descriptions to lead the customer to parts of the site that are otherwise invisible.

Maintaining client-side continuity while communicating with a large database requires the behind-the-scenes server communication that Pet Market demonstrates. The brief pause before the response is scarcely detectable.

A process without gaps is much more likely to be clear and comfortable for a user, whether the user is at an online store selling pets, a business application, or a customer self-service web site.

Managing Screen Real Estate

In the Java Pet Store and the .NET Pet Shop, the stages of the shopping process (i.e., store page, gallery page, product page, search results page, shopping cart page) appear on different screens. Pet Market's single "big picture" screen contains all state information that concerns the customer. Nothing is lost; nothing is out of sight. When the design requirements of a stage of the process change radically, (as, for example, when the customer moves from shopping to checkout,) Pet Market holds place and state by placing checkout in a modal overlay, superimposing the checkout module over the shopping layout and giving it focus. Customers can see where they have been without accidentally reverting to another page with an errant mouse click.

Users have become accustomed to using the back button as a tool for navigation. After drilling into a catalog, they will often "back" their way out of a hierarchy. Macromedia Flash sites have the default behavior of allowing the back button to take a user who is midway through a Flash experience to the beginning of the experience, or even to a previously visited page or site.

The designers of Macromedia Pet Market dealt with this problem by registering states of the application with the browser's history object. This allows Pet Market to act much like frame-based HTML as regards the forward and back buttons, but under the developer's control. Using this feature, the developer decides what states of the application to register as "pages." For the user, the back button is there when they need it and it behaves predictably.

The fact that customers can back out of a dead end does not excuse interfaces that lead them there in the first place. Pet Market addresses both sides of the issue, providing user access to all areas of the application all the time, while retaining navigational tools already familiar to them.

Pop-Ups vs. Overlays

Sometimes users want or need to make a small side trip from the principal process of a conventional e-commerce site to visit Help, check out a policy or an ad, or to see an enhanced view of a product. Designers often choose pop-ups—a second browser window containing the link target—to present this content to the user. The user interacts with the pop-up, closing it when they no longer need it.

Pop-ups are so useful to designers and developers limited by HTML that they appear everywhere despite the fact that they are annoying to customers and difficult for designers to control—a true usability nightmare. Not only are placement and appearance a challenge to control across platforms and browsers, but pop-ups are not modal. The developer cannot control whether the pop-up or the base page is active.

As a result, customers can lose track of their objective by attempting to pursue it in the pop-up window instead of returning to the page that spawned the pop-up. In addition, since the developer has no control over which window is on top, either the pop-up or the main window can become covered and consequently lost. An injudicious double-click can hide a pop-up window, leaving the customer without the provided information.

In Pet Market, the development team used an overlay to branch into the checkout process without taking the customer away from the main focus of the process. In Macromedia

Flash, graphic elements can be smoothly and precisely superimposed, and the response of the “underlying” design elements is completely under the designer’s control.

The checkout process for Pet Market uses a modal overlay, which guides the user to complete the checkout section. Although the “Fish” menu is visible, customers can’t click on it until they click on the “Shop More” button that causes the checkout overlay to disappear. (Figure 4)

Figure 4: The Pet Market checkout, an overlay, covers the remainder of the Pet Market functionality, since it is not applicable at this stage of the shopping process.

The Pet Market checkout does what all developers wish pop-ups would do—if they only could. It is placed precisely on the page, commands the customer’s attention, and remains an integral part of the design layout.

Because checkout and registration are modal overlays, the entire Pet Market purchase from beginning to end is effectively—for the customer—a single view. At no time do customers need to navigate to a place in the site that they cannot already see.

Responsiveness

Immediate feedback is one of the best ways to keep customers on the right track during a shopping process. It prevents or lessens the consequences of customer-side errors. Separation of action and reaction allows forgetfulness and confusion, makes fixing mistakes more difficult, and so forth. Every time there is a gap in the user’s process—either in space or time—there is a chance for the user to get lost or confused.

The quicker an application responds to users’ actions, the more responsive it feels. Responsiveness is one of the tradeoffs that web application designers have to make. Designers can reduce the amount of information displayed to minimize the response time, but there is always a practical minimum amount of information that the site must display. By comparing the demos, we see that if designers can change when the application ships this information over the wire, they can minimize the customer’s irritation caused by the inevitable wait.

The Java Pet Store and .NET Pet Shop both start quickly, but the waiting times between pages, as well as the trips to check out the shopping cart, soon begin to add up. Any advantage gained by an initial fast load eventually dissipates under the accumulated wait from one click to the next.

While Pet Market takes longer to download initially, this is a function of the application's design, not an inherent limitation of Macromedia Flash. To the user, it feels more like the launch of a desktop application than the initial load of a home page. (Often, designers can mask the initial load by implementing a "pre-loader" that provides instant interactivity or information. Instead of using a "pre-loader," Pet Market's designers chose a loading sequence that includes feedback about the various parts of the application.)

Once Pet Market's shopping process begins, the user moves quickly and responsively. Since the application is all self-contained in the "big picture", it downloads in a single pass. Pet Market caches subsequent information (such as the "History" window) on the client side, minimizing any delays during the user's interaction.

By carefully crafting when necessary information is retrieved from the server, the designers have made a smoother shopping process in Pet Market, choosing to have the site move faster when the user is actually shopping and making decisions.

Self-Explanation

The French use "l'architecture parlante" (talking architecture) to describe buildings and spaces whose purpose is clearly conveyed by the visual cues in the design. Ideally, a web site should also "explain," through its layout and dressing, what each piece is for and how it works. Pet Market cleverly uses animation in several places to reinforce and support the sequence of the shopping process. In addition, lists (such as Search, Cart, and History) smoothly open and close so that customers can register their appearance and disappearance. The Pet Market interaction designers used animation thoughtfully—not to dazzle and bewilder the customer, but to focus the customer's attention on moving the shopping process forward.

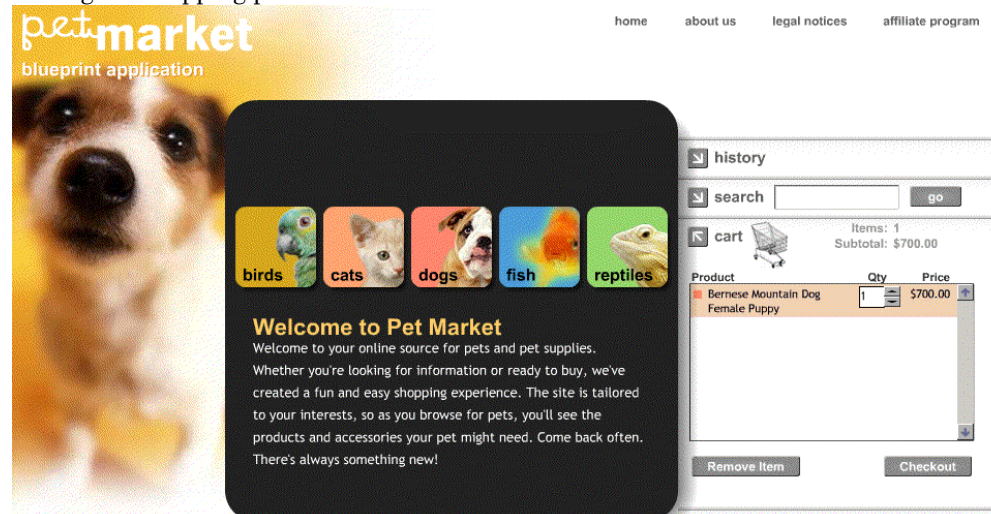


Figure 5, Step 1: The user has made one purchase and is in "Home Page" state, but the contents of the cart are still visible.

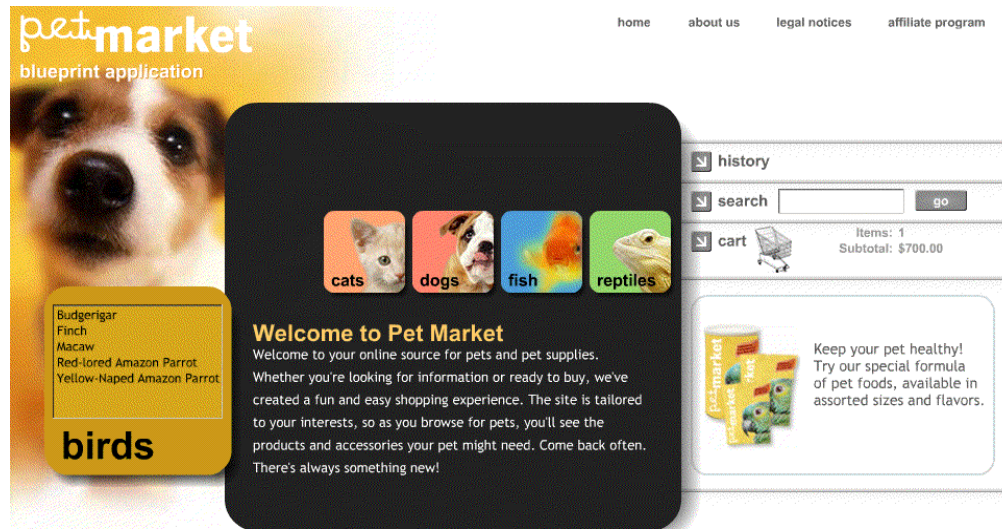


Figure 5, Step 2: The “Birds” tab moves to a new position and changes shape, appearance, and content. It is now a pick list. The user is now in the “Birds” store, ready to choose a bird. Note also that the cart has closed, replaced by a “bird-appropriate” ad.



Figure 5, Step 3: When the user selects a bird, the other species’ tabs move down the screen to an auxiliary position making room to display the specific product. The user is now learning about a parrot.

Figure 5 shows Pet Market’s 3-step, home page to store page to product page sequence. Changes in the screens involve change of both place and form for the species category tabs. When a user’s selection invokes changes, they take place continuously. The movement clearly connects the changed forms for the user, leaving no doubt about the identity of the “Birds” list or how it got there.

With the Pet Market application, objects change color and opacity as they change function, and elements move from place to place so that customers don’t lose track of them.

The Pet Market designers reinforce the transition from the “Home” configuration to one of the “species” configurations by animating the shapes (the tabs for the other categories glide to one side, still available, but out of the way) and opacity, as the central area fades before changing function.

In Figure 6, the user, looking at a budgie, clicks on the list to look at a parrot. During this process, which corresponds to the serving of a new HTML page, the user has ample indication of progress, and never loses focus on the product display area.

The transition from Step 2 to Step 3 in Figure 6 becomes even more apparent to the user by fading out the contents of the black rectangle before the content changes. This additional feedback warns the user that a change is about to happen, making confusion and dislocation even less likely.

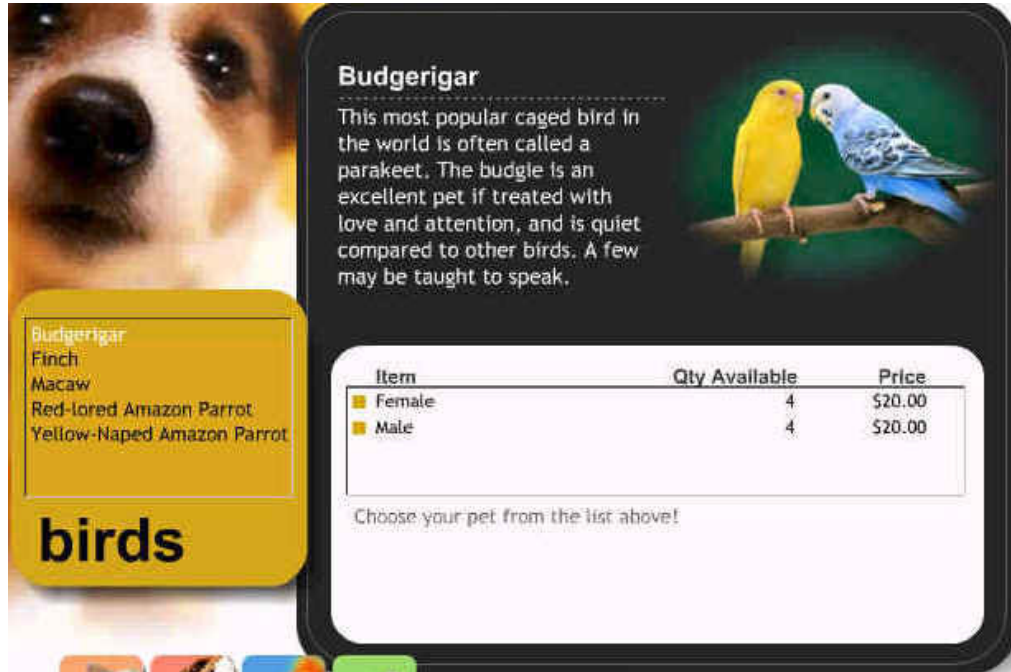


Figure 6, Step 1: The user, looking at a budgie, chooses a parrot.

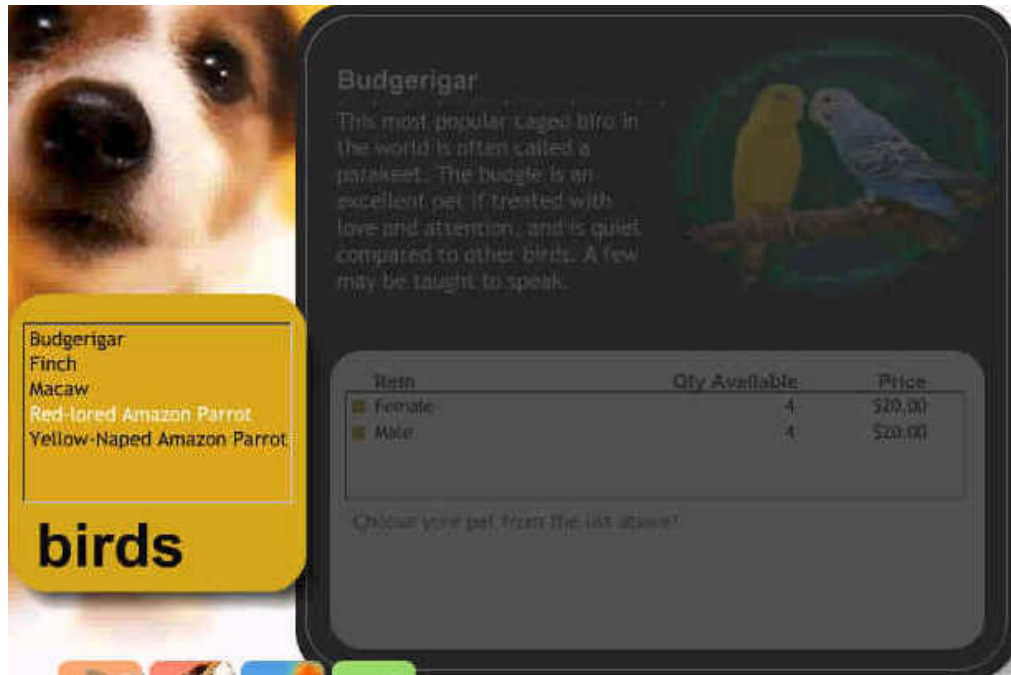


Figure 6, Step 2: The budgie fades, using a screen effect.

Pet Market illustrates what happens when animation is under the control of interaction designers whose primary goal is to expedite and clarify the purchasing process. They use these powerful tools to guide and orient customers, not to overwhelm them or steal their attention.

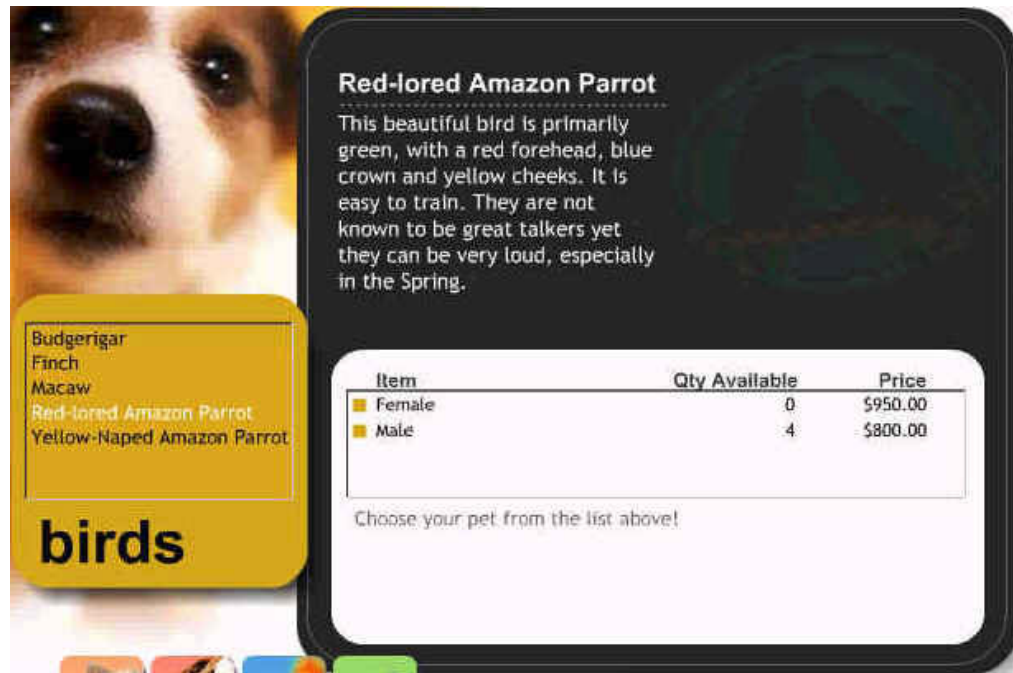


Figure 6, Step 3: The text for the parrot appears.

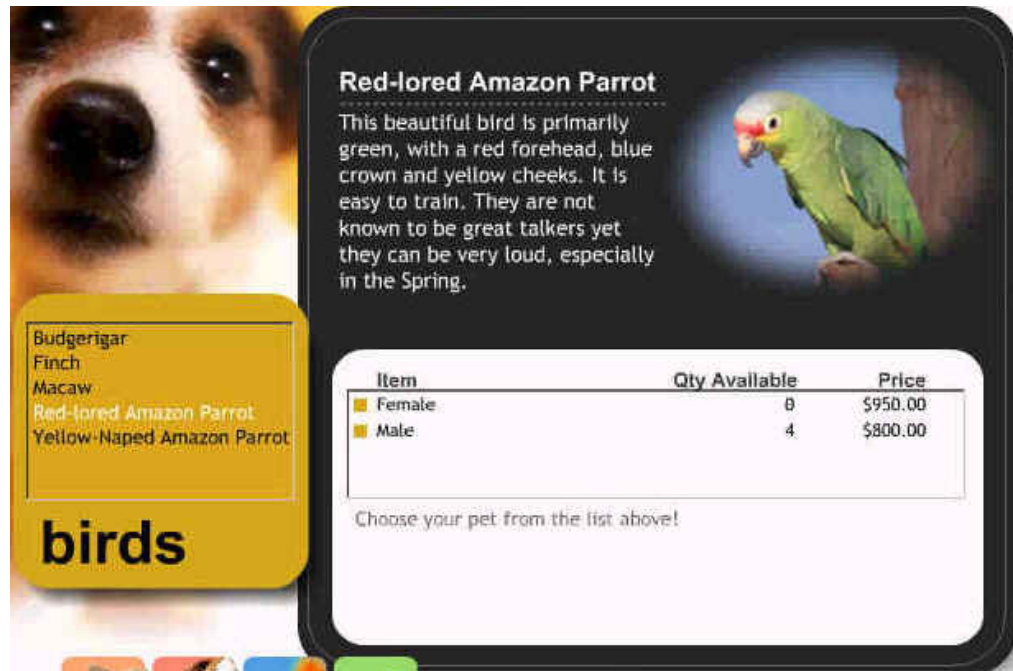


Figure 6, Step 4: The picture of the parrot arrives last.

What Are the Applications “Made Of”? – Key Ingredients

We have discussed a group of qualities that make sites more usable. To an extent, one might say that it is part of the designer or developer’s task to build these qualities into a site, thereby enhancing usability. So it is important to examine the extent to which the presence or absence of the qualities discussed in the previous section depends on what they were made of—what designers and developers used to build them.

The components of the Java Pet Store and Microsoft .NET Pet Shop that the user sees and interacts with were assembled by server-side scripts and rendered using the basic formatting capabilities of HTML. Pet Market uses Macromedia Flash on the client side, separating data, business logic, and presentation. Moreover, many details of the Pet Market interface—such as the ability to tab through the entire user interface controls relevant to the shopping process or the ability to absolutely control relative positioning across browsers and platforms—are simply not possible in HTML.

HTML is a limited toolset intended for the *display* of information; it has very limited tools for *exchanging* information.

A web site built with HTML has more in common with a correspondence by mail than with a face-to-face conversation. Each action by the customer sends a message to the server, which replies with a brand-new page. The only way a designer can respond to a customer with new information in HTML is by serving a new page. HTML cuts a process up into steps and stages separated by waits and changes. Inevitably, the process that the user is attempting to carry out is broken up, both in time (even with the best connections there is a wait) and in space. While the spatial disconnect is usually the more dramatic (abruptly returning the user to the top of a long page, for example) time gaps can also be disconcerting.

J2EE and .NET use the same model as HTML; HTML is the medium used to serve the pages that are first constructed on the server then delivered to a browser. With the exception of custom-written scripts and applets, HTML underlies what J2EE and .NET can do on the client side. The strengths of J2EE and .NET are the structure that they bring to the server side of the application. They give relatively little attention to tools designed to help designers build the client side—the part of the site with which the user directly interacts.

The qualities discussed in the previous section that enhance an application’s usability are characteristic of design and development tools oriented toward the client side.

Key Tools for Developers

While Pet Market and its Java and .NET cousins are interesting demonstrations of capabilities, the question arises: *Is Pet Market’s sophistication because Macromedia got the world’s most talented developers to build it or could it have been built by any competent application developer?* To answer this question, we took an in-depth look behind the scenes of the features and functionality of the tools the developers used to build these three applications.

Macromedia Flash’s functionality focuses on sophisticated *client-side* development. The functionality of .NET and J2EE focuses on the server side. For this reason, we would ex-

pect the Macromedia Flash-based Pet Market “concept car” to show the most “traits” that support more usable interfaces, as it does.

Development environments and tools don’t make applications usable. Usability is the responsibility of the designer of the interaction and the developer of the functionality. From the users’ point of view, the key differences lie in what can be achieved and what will benefit them in each environment. Time and money limit what can be accomplished—in terms of usability, functionality, or anything else. The right development tools can help to overcome these limitations. This section looks at usability for developers, because that is the first—and most important—step in gaining usability for users.

In this comparison, we include Swing with J2EE and Visual Studio.NET for development on the client side, because these platforms could include a client-side development effort. In this white paper, our primary interest are tools for the development of general, browser-based applications deployed across platforms and browsers, not large and highly specific applications targeting a single browser and platform combination.

First, for each development environment, we’ll take a detailed look at the following aspects:

- Components and Customization
- Communicating in the Background
- Modal Control
- Animation

Then, we’ll add it all up and see if one environment stands out from the rest.

[Editor’s Note: This section of the white paper deals with issues specific to developers and designers. Some of the following discussions reference products and techniques that might not be familiar to everyone—you’ll want to seek the opinion of an expert you trust.]

Components and Customization

Developers and designers need two things to produce rich and usable applications with time and resources available (which are usually far too scarce either to reinvent the wheel or create museum-quality design from scratch). Developers need an adequate set of effective widgets and components, so that they don’t have to build everything from scratch. Beyond that, they also need a good set of tools to customize these components.

In Pet Market, components change size, shape, and color, and move from place to place, but always for a purpose—to illustrate and convey to users what is taking place in the process, to manage the available screen real estate, or to guide users to their next steps. It is principally this *controlled customization* that distinguishes the “feel” of Pet Market from the feel of the Java Pet Store and the .NET Pet Shop.

For example, the designers placed the category list boxes in Pet Market within rectangles with rounded corners. They chose the colors of these rectangles to match the colors associated with the animals in the category.

While all three platforms have off-the-shelf components for common UI controls, Macromedia is the only one of these vendors that provides an easy way to maintain platform independence for the look and feel of the user interface. While not every application requires a customized look and feel to succeed, the décor of the user interface, like the décor of a fancy restaurant, provides an important aspect of branding for many applica-

tions. Moreover, customized user interface components can make use of other cues to make the application more usable.

Both Swing and .NET allow developers to skin their components for attributes, such as color, appearance, icon size, and text position, but it's not as easy as in Flash. As a result, Swing and .NET applets tend to be conservative in their look and feel, very similar to each other.

On the other hand, designers can make Java Swing components look like the client machine's operating system look and feel. Microsoft .NET components take on their look and feel from the browser. Making Flash applications vary based on the local platform or browser takes quite a bit of effort.

Interestingly, while there are those people who insist that usable application must conform to the local platform's look and feel, we have not seen any evidence in our observations of users that this is actually true. One doesn't have to look much farther than the very lucrative PC gaming world to see that novel interfaces, when designed well, can be very appealing. For the same reasons that people decorate and customize the front plates on their cell phones, we are seeing a trend towards very stylized front-ends to applications. Macromedia Flash is in the best position to take advantage of this trend.

Communicating in the Background

All three contenders can carry on background communication with the server. The differences lie in the time and effort it takes a developer to establish this communication.

Using a technology known as Macromedia Flash Remoting, connecting a Macromedia Flash user interface to application resources or web services is fast and straightforward. Flash Remoting uses an asynchronous RPC model that allows Flash to invoke methods directly on web services, EJBs, JavaBeans, JMX MBeans, POJOs, ColdFusion pages and components, ASP pages, and a number of other resources in the ColdFusion, J2EE, and .NET worlds. A simple set of common APIs connects to any of these resources and handles conversion of these types from their ActionScript (the ECMAScript syntax used in authoring Macromedia Flash) versions to native types. A developer can call web services with no server-side coding.

Java RMI (Remote Method Invocation) is in some respects more sophisticated than Flash Remoting, and is completely integrated into the core Java libraries. However, invoking methods on remote objects as if they were ordinary objects requires an extensive setup for each instance. For each class that the designer wants to access, they must define a remote interface and compile a remote proxy stub. What takes a minute to develop in Macromedia Flash MX takes ten times that long with the Java RMI.

In Microsoft .NET, client-side components can also be easily configured so that events fire back to the server and any .NET resource can be available. The server event handler then triggers other changes on the page. Instead of the typical page reload, the client-side component is refreshed. While the appearance of these events in a pure .NET implementation would be similar to what users would see in the Pet Market demo, connecting to resources outside of .NET requires additional effort. More importantly, data transformation and debugging is extremely challenging.

That being the case, .NET background communication is an ordinary form post responding with the entire HTML page. Therefore, while it does accomplish the preservation of

field values for form validation, it requires more bandwidth than Flash for the same data transfers.

Modal Control

“Modal control” refers to the layering and reuse of shared real estate without shipping new pages, as shown in Figure 4 above. In Pet Market, modal control is a key part of the experience—the customer’s interaction with various parts of the visible UI is under the developer’s control.

Swing also has modal behavior built in. The developer creates another frame, tells it to pop up and be modal, and it is. The user can’t click on the frame’s parent until they deal with the modal dialog.

Effects similar to those pictured in Figure 4, if developed in Microsoft .NET, would require coding from scratch. Since there is no built-in capability to have modal dialogues or overlays, developers need to make a large investment to get the same effect.

Animation

Several features unique to Macromedia Flash animation make possible continuous movement, property changes (such as brightness, tint, opacity, scale, and position), and the animation of any object. Flash’s use of vector objects makes movement and change very smooth. In addition, because Flash has a player common to all platforms, designers can simply animate using a broad range of properties dedicated to manipulating screen objects.

The Macromedia Flash development environment gives full control over all properties in all delivery platforms. To get the same sophisticated animation without Flash requires a third-party authoring environment to build JavaScript-based animation, which means that not all properties can be animated in every platform. The smooth and ubiquitous touches of animation which guide the customer from point to point in the purchase process in Pet Market would be much more difficult—if not impossible—if based solely on JavaScript.

Animation is also easy in Swing. One difference between Swing and Flash is that Flash replicates fine rendering details across platforms, maintaining a consistent look across platforms. Swing is set to render itself so as to blend in to each platform and look as much like other apps and widgets as possible. While Swing can be made to imitate Flash and maintain a distinct look, this requires additional effort on the developer’s part.

While Microsoft’s .NET has an entire package devoted to rich web UI, it offers meager resources for animation and no specific authoring tools.

Adding It All Up

Based on our analysis of the development environments, we’ve put together a direct comparison of J2EE + Swing, MS.NET + Web Forms, Macromedia Flash, and HTML. (Figure 7) *The aim of this table is not to say what can and can’t be done in each case, but rather what is easy and what is difficult or impossible.*

We measure each capability by ease of accomplishment. Based on our assessment, this is a crude scale of the inherent usability of .NET, J2EE, and Macromedia MX from the developer’s point of view.

Feature/Capability 0 = can't do it 1 = requires line-by-line coding 2 = existing widgets or methods provided 3 = GUI environment assistance	Flash	J2EE + Swing	MS.NET + Web Forms	HTML
Customization of components	3	2	1	0
Background client-server communication	3	1	1	1
Modal overlays	3	3	1	0
Animation	3	1	0	0
Client-side form/validation coding	2	2	3	1
Score	14	9	6	2

Figure 7: Design controls on the client side rated by ease of implementation across four development platforms—Macromedia Flash, J2EE+Swing, and MS .NET compared to plain, “garden-variety” HTML.

From our assessment, we see that Flash provides many of the capabilities built right into the development and runtime software. Macromedia claims that the Flash player is “the most distributed piece of software on the planet.” This means a consistent environment across platforms and browsers for application development. This is an advantage that should not be overlooked.

Over the years, we’ve conducted hundreds of usability tests on dozens of application development tools. In our opinion, the Macromedia Flash development environment is one of the most powerful tools we have seen for creating usable experiences on the Web.

Because of the power of the environment, building applications that have the sophistication and flair of Pet Market doesn’t require an extremely talented team of developers. Instead, the power of Rich Internet Applications is made accessible to a wide range of developers (from script writers to programmers with computer science degrees), reducing the overall cost of development to organizations.

In Defense of Visual Appeal

We have left one of the most important points for last: Visual Appeal.

While Pet Shop and Pet Store have the drab, mechanical, uninspired look and feel we have seen so much of on the web that we no longer even bother to complain about it, Pet Market is visually pleasing and laid out in an innovative and attractive new visual paradigm. Frankly, it looks great.

Is this important? We think so. Test users are more comfortable and enthusiastic about using Pet Market because the “software ergonomics” of the site feel better.

Donald A. Norman, author of *The Psychology of Everyday Things* and principal of the Nielsen Norman Group, has recently summarized this point better than we ever could:

“Usability? Yes, that matters, but beauty, pleasure, and fun—those are truly important. Yes, the product has to be balanced, yes, it should provide value, fulfill the needs of the users, and make good business sense. Sure, all of that. But if it is unattractive, if it doesn't feel right, who cares if it works?” (Communication on CHI-WEB, 2 May 2002)

A technology like Flash gives designers and developers so much control over so many aspects of the client side—the last mile to the user—that it has provoked some initial negative reactions. Instead of fighting to accomplish anything more than crude frame animation, designers suddenly find themselves in a position where major restraint is required not to go totally overboard with new ideas. Overcome with enthusiasm, a few momentarily forgot what Uncle Ben told Spider-Man: “With great power comes great responsibility.”

As a result, we hear much about gaudy, frivolous, and unfocused sites that are difficult and confusing to use—as if this was some kind of novelty on the web—and not enough about creativity serving the user, allowing for more intuitive and useful applications, which the Web desperately needs.

The Power of Rich Internet Applications

When compared to the Java Pet Store and .Net Pet Shop, we can see the power of Rich Internet Applications demonstrated by Macromedia Pet Market. In our examination, we pinpointed the major advantages Rich Internet Applications have over “page by page” designs for creating web applications. This new style of application offers developers several advantages for creating usable client-side code not possible in HTML: client-side continuity, the specific client-side development tools, and animation power.

In our analysis, Macromedia Pet Market application was the standout leader in the web application space. Flash MX makes building web applications as easy as Sun's and Microsoft's environments. More importantly, it delivers an extremely usable application with equivalent (or less) development effort. Developers using J2EE and .NET struggle to achieve what Macromedia Flash delivers easily. Consequently, Rich Internet Applications built with the Macromedia MX product family make it easier to deliver significantly more intuitive, responsive, and effective user experiences.

The concept cars tell us what the future will bring. Sometimes they are pure fantasy—the whims of designers with unlimited budgets and resources. However, sometimes they are practical investigations of a future that is near, affordable, and within reach.

Based on what we've seen in Pet Market, we believe that it will not be long at all before Rich Internet Applications begin to appear in corporate intranets and in-house applications as well as consumer-facing Web sites. Tools like Macromedia Flash MX are pushing the envelope for state-of-the-art development technology, making tremendous power extremely accessible to today's application developers.

New Report Available: Making the Best with Flash

A new report, written by Christine Perfetti and Matthew Klee, provides a detailed description of our latest research on developing Flash applications. This report discusses the five best practices for creating engaging content with Flash. Perfect for anyone currently developing in Flash or is considering it in the future.

For more information, visit <http://www.uie.com>.

About User Interface Engineering

About Us - User Interface Engineering is a leading research-driven company specializing in web-site and product usability. By providing usability information based on detailed observations rather than opinions, we empower development teams to create web sites, software applications, and other products that increase customer satisfaction and loyalty.

Best Practices Report Series - Based on years of watching real users on real web sites, this report series reveals design practices of the most-successful sites. We've packed each report with practical, hands-on advice plus plenty of examples from live, production web sites. You'll learn exactly what the best sites do to succeed and find out what mistakes to avoid. Contact us at uie@uie.com if you would like us to notify you as soon as we release our next report.

User Interface Engineering Web Site - Our web site, www.uie.com, has many resources for people involved in web and product development, including:

- Free white papers about our recent e-commerce research
- Articles on various aspects of web and product usability, including paper prototyping
- Descriptions and schedules for our public courses
- Upcoming conferences where we will be speaking

How to reach UIE - If you have questions or would like more information about our offerings, please send e-mail to uie@uie.com. You can reach us at (978) 374-8300, by fax at (978) 374-9175, or by mail at 242 Neck Road, Bradford, Massachusetts 01835, USA.

Author Information**Will Schroeder, Principal and Senior Researcher**

Will, the “Chief Science Officer” at User Interface Engineering, specializes in project management, systems analysis, data analysis, and psychophysics. Before joining User Interface Engineering, he worked for 17 years at Foster-Miller Inc., an engineering consulting firm. Will has lectured at the Harvard Graduate School of Education in visual studies and environmental design.

Will has a Masters in Business Administration from Babson College, a BET in Mechanical Engineering from Northeastern University, and a BA in English/Mathematics from Harvard College.

You can contact Will at wills@uie.com

**Christine Perfetti, Consultant and Senior Instructor**

When you talk with Christine, it doesn't take a long time to realize that you're talking to someone who knows a whole lot about designing usable web sites.

Christine is co-author author of *Making the Best of Flash*, a report about the best practice for creating engaging experiences in Macromedia Flash. She is also co-author of the white paper, *Macromedia Flash: A New Hope for Web Applications*. As one of the most requested instructors at User Interface Engineering, she has worked with dozens of companies on their toughest web design problems. In the last year alone, she's been a top-rated presenter at CHI 2002, the UIE Research Forums, Intranets 2001, and STC regional conferences. She has taught Human Factors at the prestigious Tufts University Gordon Institute for Engineering Management.

You can contact Christine at cperfetti@uie.com.

**Jared M. Spool, Founding Principal**

If you've ever seen Jared speak about usability, you know that he's probably the most effective, knowledgeable communicator on the subject today. What you probably don't know is that he has guided the research agenda and built User Interface Engineering into the largest research organization of its kind in the world.

Jared spends his time working with the research teams at the company, helping clients understand how to solve their design problems, and is a top-rated speaker at more than 20 conferences every year. He is also the conference chair and keynote speaker at the twice-annual User Interface Conference, is on the faculty of the Tufts University Gordon Institute, and manages to squeeze in a fair amount of writing time.

You can contact Jared at jspool@uie.com.