UIE REPORTS | FUNDAMENTALS SERIES

THE DESIGNER'S GUIDE TO WEB APPLICATIONS | PART II:

Web Apps Tour 2007:

LEARNING from SUCCESSFUL DESIGNS

By Hagan Rivers and David Rivers, Two Rivers Consulting

User Interface Engineering

User Interface Engineering | 510 Turnpike Street, Suite 102 | North Andover, MA 01845 | (978) 327-5561 or (800) 588-9855 | www.uie.com

UIE FUNDAMENTALS REPORTS

There are concepts every web designer should know. These are the concepts that determine how people find and digest information. In other words, how they use the web.

In the UIE Fundamentals Series, User Interface Engineering brings such core concepts to the surface. The series presents proven, time-tested ideas that drive today's most successful designs.

In this report, *The Designer's Guide to Web Applications, Part 2: Web Apps Tour 2007,* Hagan Rivers and David Rivers, pioneer web application designers, examine seven unique web applications and highlight the most interesting design elements. We believe their insights will change the way you think about designing web applications.

WEB APPLICATION STRUCTURE

Written By: Hagan Rivers and David Rivers, Two Rivers Consulting
www.tworivers.com

Published By: User Interface Engineering
510 Turnpike Street, Suite 102
North Andover, MA 01845
(978) 327-5561
www.uie.com

Cover Design: Laura Spindler

10 9 8 7 6 5 4 3 2 1

A LETTER FROM JARED M. SPOOL, PUBLISHER OF THE UIE REPORTS

THE MAKING OF A WEB APP TOUR

Art students study great works of art. Music students study great works of music. Architecture students study great works of architecture.

If you're trying to learn how to develop a web-based application, what do you study? Great web-based applications, of course. As a student of web-based applications, you'll want to go through their designs and see how their designers solved difficult problems—challenges you might be facing yourself.

When you're knee-deep (or deeper) in a design project, it's hard to set aside the time to look at the designs of other applications. That's why we asked Hagan and David Rivers to put this report together. We figured, since it'll be hard for you to go see these applications yourself, we'd bring them to you.

CHOOSING A TOUR GUIDE

I am not an artist, by any stretch of the imagination. (In fact, the only thing I can draw is blood.) As a trained software engineer, I can appreciate elegance in an algorithm or code segment, but when I look at a painting or a sculpture, I have no sense as to what makes it great or interesting.

That said, one of my most favorite activities is to go to art museums with my good friend, Nina. Nina is a trained artist and she understands what we're looking at on a level that I've never experienced. In addition to being a talented artist, she's also excellent at providing insights into the subtleties of what makes the art we're seeing interesting and unique. Traveling to the museum with her opens up ideas I never have when I go on my own.

I first met Hagan and David ten years ago at a conference where there were presenting a full-day seminar, touring dozens of web sites and viewing their designs. At the time, they were working for Netscape, helping them develop some of the Internet's first web-based applications.

From the moment I first talked with Hagan and David, I was impressed with their insight into the subtleties of what makes web-based applications work. Over the years, we've had several opportunities to work together, and each time, I always feel smarter after interacting with them. In my mind, Hagan and David are now the premier experts on web-based application design and they are the first people I'd want with me on our tour.

CHOOSING THE SITES

When we sat down with Hagan and David to write this report, we had to decide which applications we would include. We settled on two criteria: First, the application had to carry the company. In other words, the success of the company needed to depend on the success of the application. The sites you find in this report all meet this criteria.

Second, the application had to solve hard problems in interesting ways. As designers, we can debate and discuss whether a given solution is the best solution. However, the variety of approaches gives us the insights necessary to tackle the challenges we face in creative ways. As you read this report, you'll see Hagan and David do an excellent job of pointing out where the designers have done an excellent job and where there's still room for improvement.

Let's look at the history of each site we'll be touring:

SALESFORCE.COM

Founded in 1999, Salesforce.com changed the face of the enterprise software world by making a major suite of sales force automation tools available through the browser. They now have more than 550,000 subscribers at 27,100 companies worldwide, using their web-based application solutions. In their fiscal 2006 year, their application generated more than $280 million dollars in revenue.

The primary users of their application are salespeople. While some may have significant computer experience, many are often using Salesforce.com as one of their first applications. In addition, it's an application they "live in" most of the time. So, the design has to be really easy to learn and quick to work with.

The success of Salesforce.com's business is a testament to the effectiveness of their application's design. Every detail, from the way they distribute the functionality, to how they present the individual pieces of data, has been well tuned for their audience.

WEBOFFICE

Originally called Intranets.com, they changed their name to WebOffice when the web conferencing company, WebEx, acquired them earlier this year. WebOffice is trying to tackle a difficult problem of helping remote workers easily collaborate, to give them a virtual office.

What makes WebOffice interesting is their approach to supplying a rich functionality while keeping the design straightforward. It's a nice comparison to Saleforce.com's design.

SERENATA FLOWERS

SerenataFlowers.com has stepped away from the mainstream, with a very interactive approach to displaying their products. By utilizing AJAX technology, they've created an innovative method to quickly winnow down the selection of products by price and color.
This is an important tour stop for anyone who needs to deliver to their users a fast method for selecting an item out of a large list of possibilities.

BACKPACK

Backpack is one of a series of interesting products from the folks at 37 Signals. What makes Backpack interesting to us is how they've managed to create a minimalist approach to design, by only revealing functionality when it's necessary to see it. This technique could be very useful to anyone needing to build an application requiring sophisticated data manipulation.

APPLE CONFIGURATORS

Apple.com wants to sell you a computer. Yet, their computers have a complicated array of options. How does Apple ensure you get the computer you want?

They've created a configuration tool that lets the customer specify the options they desire. Keeping the customer informed on how changes to the configuration affects the price and updating them on any conflicts is a complex process. Apple.com's designers have done a nifty job of solving this challenge.

SURVEYMONKEY

SurveyMonkey is one of the oldest applications on the web, providing a tool to create, manage, and analyze online surveys. It provides an interesting editing capability and some very effective project management solutions. SurveyMonkey has been around the block a few times and their experience shows in their design approach.

WRITELY (NOW GOOGLE DOCS)

Writely turned the web-based application world on its head when it produced a browser-based word processor that mimicked much of the functionality and interface available in traditional desktop word processors, such as Microsoft Word. Of all the applications on our tour, this is the one that will be most interesting to those designers who are moving their applications from a desktop environment into a browser delivery platform.

Since we started the production of this report, Google acquired Writely and they've re-released it under the name

Google Docs. It demonstrates the power of the web and how, with a little ingenuity and a lot of hard work, you can accomplish almost anything.

More to Come

Of course, in deciding on these applications, we couldn't include many other excellent applications. That's why we've decided to make our tour a regular publication. We plan to tour other applications in future editions of this report series, looking at the best solutions to the hard challenges application developers are facing today.

Jared M. Spool
Publisher, UIE Reports

Table of Contents

Web Apps Tour 2007: LEARNING from SUCCESSFUL DESIGNS

Each year, our town holds a Kitchen Tour as a fundraiser for the local schools. When people purchase a ticket for the tour, they get a gourmet picnic lunch and a map of several houses around town. Each of those houses has a beautiful, very clean, and usually recently renovated kitchen. While on the tour, visitors have an opportunity to see what a well-designed kitchen looks like and to steal ideas from it for their own homes. It's a win for everyone: the home owners show off their very expensive renovations, the builders get to advertise, the schools raise money, and local residents go home full of ideas.

As we walk you through the Web Apps Tour, your experience will be very much like our town Kitchen Tour, but instead of kitchens, we'll look at web applications. And just like the kitchen tour, we want you to look at other designs and learn from them.

In this report, we've included profiles of seven unique web applications. For each application, we'll describe its purpose and target users. Then, we'll highlight some of the interesting interface elements and discuss how each application tackles its specific design issues. At a minimum, you'll see some rather slick applications. More likely, you'll find solutions for many of your design problems.

Let's begin our tour.

1 SALESFORCE

WWW.SALESFORCE.COM

Salesforce is for salespeople to track existing customers and potential new leads. Users track their organization's products and pricing, manage customer contracts, develop and track new opportunities, forecast future sales, and view tons of reports. The web application's primary users are sales staff employees at medium to large companies. In addition, sales managers and company executives use Salesforce to track and predict sales.

You can sign up for a free trial demo of Salesforce online at www.salesforce.com.

Let's begin with Salesforce's home page. When users sign in, they immediately arrive on the home page. (Figure 1-1)

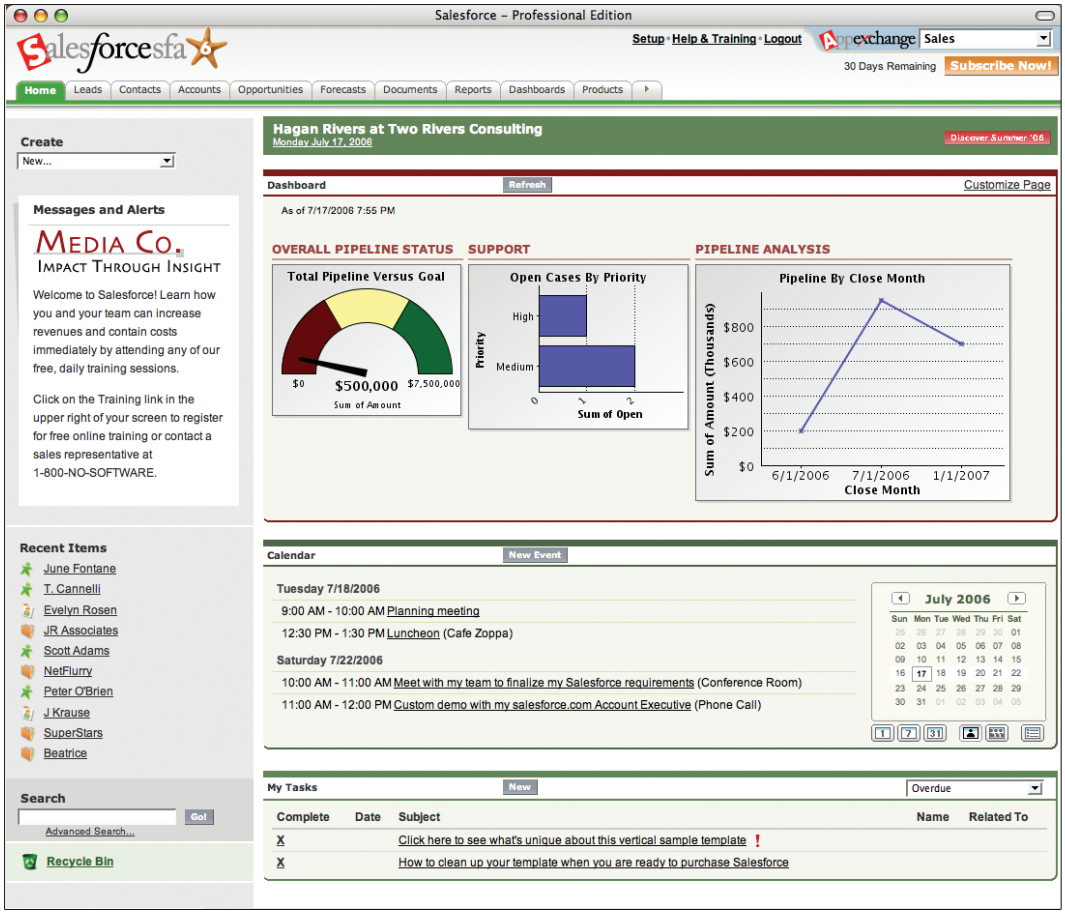

Figure 1-1: Salesforce's home page

Salesforce's home page is what we refer to as a *dashboard*. Salesforce's dashboard doesn't provide too much content but the information that is presented is timely and useful.

When designing a dashboard, here are some of the key factors to consider:

- With complex applications, a dashboard gives users an executive summary of the information. Salesforce's dashboard includes graphs of the **Overall Pipeline Status**, **Support**, and **Pipeline Analysis**, as well as **Calendar** and **Tasks** information. When a dashboard acts as an executive summary, it's critical to display the most appropriate information for users in the summary.

- When an application has many different users, a dashboard can help satisfy the needs of each user group. With Salesforce, the designers could provide salespeople with a summary like the one shown in Figure 1-1, whereas sales managers could get a completely different dashboard with information unique to their needs. To create a dashboard for each user type, it's essential for designers to understand the users' needs and their primary tasks. As a result, a dashboard is a key test of a quality design. If you're unsure of what should be on the dashboard, then you don't know your users well enough.

  Designers also have the option to give users power to customize and create their own dashboard. While most users don't change application defaults, some users do make this effort.

- A dashboard can give users an easy way to jump into key areas within the application. For example, the Salesforce **Calendar** has links to today's events so users can view the event details.

When designed well, dashboards are promising design elements that help users dive into large, complex applications. However, because dashboard design is still in its infancy, it's essential to conduct up-front research with your users as well as usability testing.

1.1 ONE TAB, TWO TABS, THREE TABS, MORE

All web applications have a *navigation system,* where users can move from screen to screen within the application. The navigation system is usually a collection of tabs, menus, and lists of links. Salesforce's navigation system consists of tabs. It's a very simple, straightforward navigation system that works effectively despite the web application's complexity.

Saleforce's first tab is **Home,** the dashboard we just reviewed**.** (Figure 1-2) Users respond positively when an application includes a home page. Users access Home after they've finished a task or they've decided what to do next.

Salesforce has 11 tabs: **Home**, **Leads**, **Contacts**, **Accounts**, **Opportunities**, **Forecasts**, **Documents**, **Reports**, **Dashboards**, **Products**, and a little **arrow**. We recommend striving

for no more than seven tabs. Salesforce's design definitely pushes the limits since users often get overwhelmed by too many tab options.

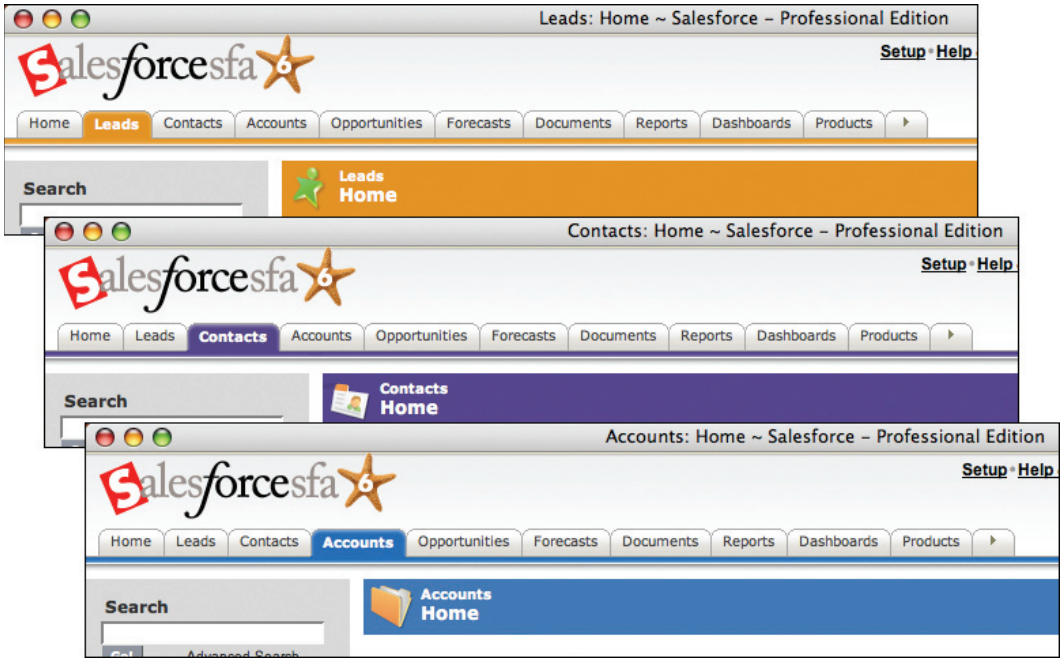

Figure 1-2: Salesforce navigation is tabs

The first ten tabs correspond to each of Salesforce's major areas. We refer to each of these areas as a hub. (A hub is a type of web application structure where the functionality is centralized and each function branches off of the center, like the hubs and spokes of a wagon wheel. We talk about hubs and other types of structures in more depth in the report, *The Designer's Guide to Web Applications, Part 1: Structure and Flows.*[1])

When users click on a tab, they go to the home page for that hub. Each home page consists of a list of items. For example, the home page for **Leads** is a list of sales leads and the home page for **Contacts** is a list of contacts. In reality, Salesforce has 15 hubs. But where are the other hubs? And what about the last tab, the little arrow?

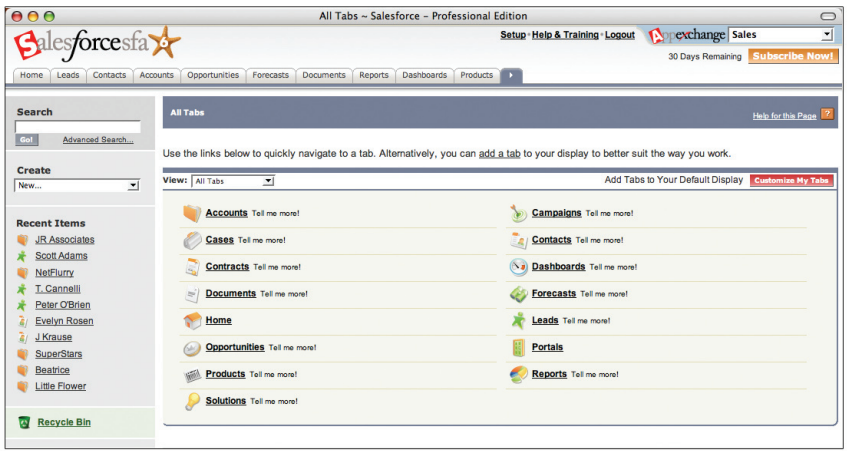

Figure 1-3: A list of all the other main application areas (additional tabs)

1 You can buy *The Designer's Guide to Web Applications, Part 1: Structure and Flows* at **www.uie.com**.

When users click on the arrow tab, they visit a complete list of Saleforce's available hubs. (Figure 1-3) From here, they can visit all the hubs that appear above as tabs and the other five hubs that were not included in the tabs.

Salesforce's designers picked which hubs to show as tabs by custom configuring the product for each customer and user type. As a result, if customers don't use the **Opportunities** area, that tab doesn't appear. On the other hand, if customers make frequent use of **Campaigns**, the designers can add that area to the tabs. Salesforce's designers can also rearrange the tabs depending on the specific needs of their customers.

Designers can learn a great deal about successfully customizing applications by carefully examining Saleforce. By choosing to customize the product, Salesforce's designers have masterfully reduced the application's complexity by hiding unused features.

1.2 ARE THERE MORE PAGES?

As we mentioned earlier, Salesforce's navigation system takes advantage of tabs. The application has 15 hubs and displays 11 tabs, but the product has far more than 15 screens. When we dig deeper, we see that Salesforce has many *child pages* for each tab. Hierarchically speaking, a child page is a page that belongs under another page. For example, the **Contacts** page includes a list of contacts. But there is also a **Contact Details** page (see Figure 1-4) for looking at the information related to just one contact. There's also an **Add a Contact** page. **Contact Details** and **Add a Contact** are both child pages.

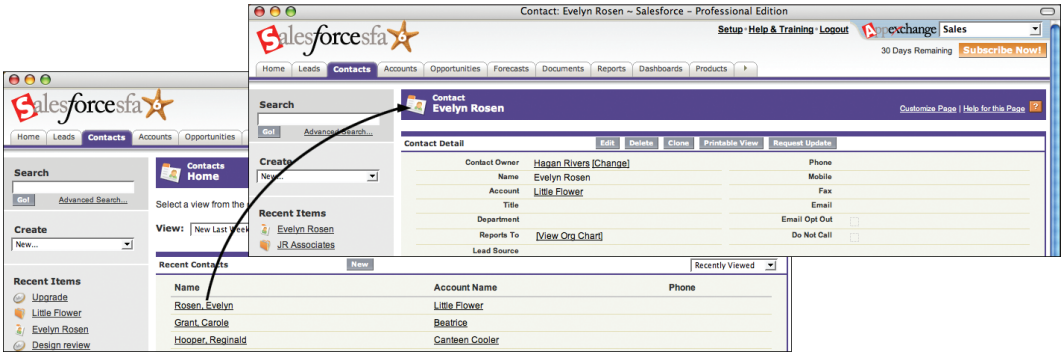

Figure 1-4: Sub navigation in Salesforce

Since the tabs don't include **Contact Details** or **Edit a Contact**, it's interesting to examine how users can navigate to these child pages.

In the bottom left of Figure 1-4 we find the **Contacts** tab. The screen includes a *list of all contacts.* **Rosen, Evelyn** is the first name on the list. On this page, we see the Contacts tab is selected and the page's title is **Contacts Home**, with an address book card icon beside it. Each hub has a unique icon and unique color associated with it. On this page, the color is purple.

If users click on **Rosen, Evelyn**, they view the *details for one contact* page on the upper right in Figure 1-4. Salesforce doesn't change the tabs at all when the user visits a child page—the **Contacts** tab is still selected.

Salesforce also has a very lean *orientation system*. While designers create a navigation system to help users jump around to various application pages, the orientation system helps users figure out where they are. Think of signs at an airport. When travelers step off an airplane, they see signs for baggage claim, transportation, and bathrooms. These signs are navigation systems—they show people which way to get somewhere. But travelers also encounter signs for the various airport terminals, which are orientation systems—they indicate where travelers are within the airport.

Users can see they are on the **Contacts** page because the tab is selected. In this case, navigation and orientation are mixed together in one control. In addition, the screen includes a lot of purple, and the page's title is **Contacts Home**. Users can tell they are on the **Edit a Contact** page because the page title is **Contact Evelyn Rosen**. From an orientation standpoint, the only difference between the two pages is the title. To get back to the list of Contacts, users can either press the **Back** button or click on the Contacts tab again.

Is the orientation system well designed? We concluded that Salesforce might benefit from orientation cues for each page as a part of their navigation system, perhaps with sub tabs. However, the designers can't provide a sub-tab for **Details** until they know *which Contact* to view the details for. Breadcrumbs are another possible solution, showing users the parent pages as well as how the pages relate to one another. If we were doing a usability study of Salesforce, we'd look to see if the existing design is satisfactory for users or if a change is warranted.

It's important to note, when we use Salesforce, we *never feel lost*. We never wonder which screen we are on or how that screen fits in with all of the product's other screens. We always understand that when we finish a task, we can click on the tab for the next hub we want to visit. If our experience is similar to other users' experiences, then we believe the cross-linking inside the application is playing a key role in creating this sense of ease.

1.3 FOLLOW THE LINKED ROAD

One of Saleforce's best features is the cross-linking between different types of items, such as contacts, accounts, and opportunities. The cross-linking is so well designed that it's like having a second navigation system, beyond the tabs. It ties everything together and makes it easy for users to move through the application.

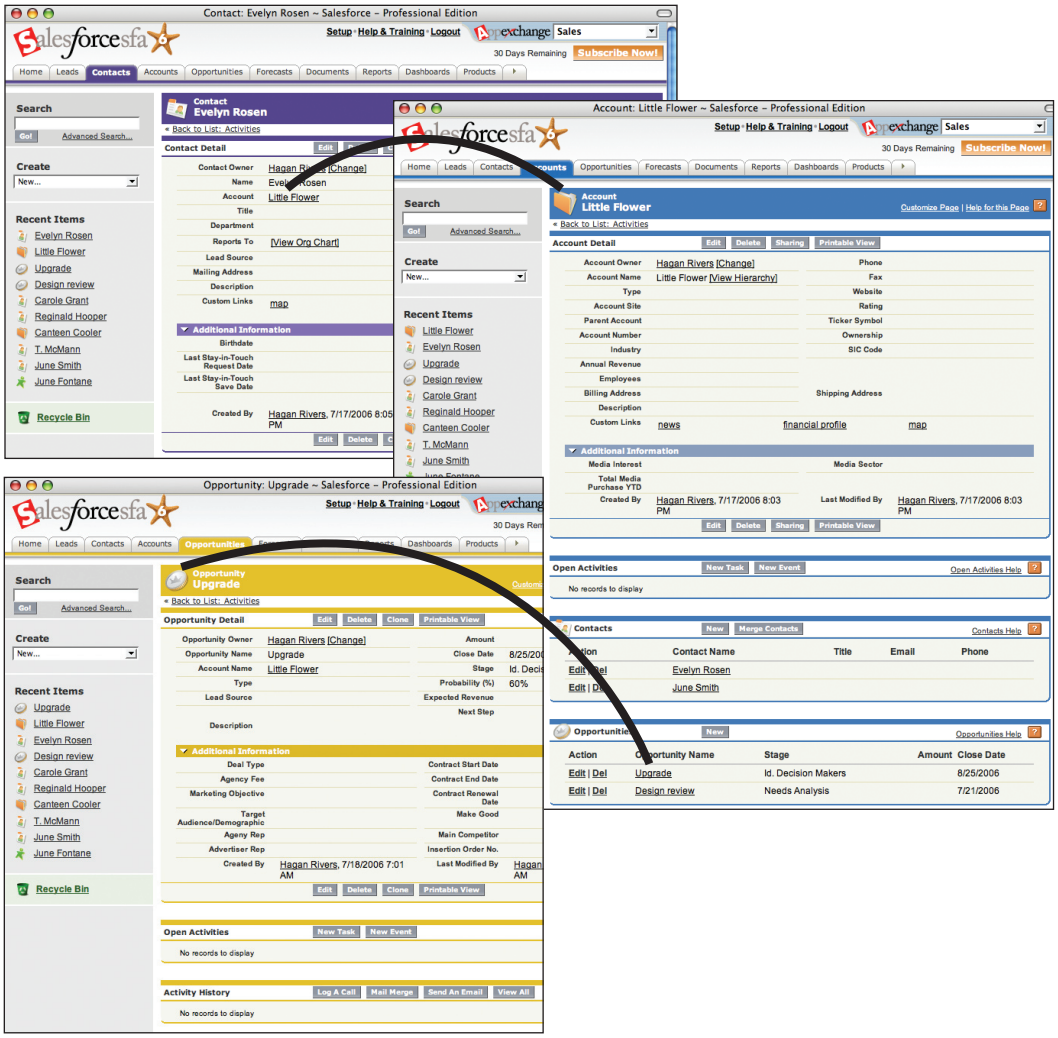

Figure 1-5: Following links through the data

Look at the Contact Details on the upper left in Figure 1-5. At the top, we learn that the Account associated with the contact, **Evelyn,** is **Little Flower**. When users click on the **Little Flower** link, they access the **Accounts** tab (right, Figure 1-5), with Details on **Little Flower**. Looking down the page, we see a list of **Contacts**, including Evelyn. The data is interconnected—users can follow the connections by clicking on the strategically placed links.

Further down the **Little Flower** screen, we find a list of **Opportunities**, including a link called **Upgrade** that brings us to the **Opportunities** tab. There, we see the details as well as a list of **Contacts** that includes **Evelyn** and the **Account, Little Flower**. This is hypertext at its best and it's astonishingly powerful and useful.

Going back to the example of signs at an airport, when travelers fly into a new airport for the first time, they have very specific goals in mind. They'll depart the plane and look for a bathroom by following the signs. Then, they'll walk to baggage claim by following signs. Finally, they'll look for the rental cars by following signs. At no point does a traveler need to look at an airport map or know which terminal they are in. They wander through the airport using nothing more than local navigation links. This type of cross-linking is clearly a good match for Saleforce's users.

Let's stop for a moment and consider Saleforce's users. Salespeople working with an existing customer must draw upon a wealth of information. When *Joe Parker from Little Flower* calls, the salesperson can open the **Little Flower Account**. They can see that Joe Parker is not a current contact at Little Flower, but Evelyn is. They can ask Joe if Evelyn still works there because they can see information about her. Or, they can examine the list of **Activities** and **Opportunities** to follow the links for sales opportunities yet to be explored with Little Flower. The cross-linking offers salespeople immense flexibility.

We are impressed by the effectiveness of Salesforce's navigation and orientation systems. They are simple and get the job done.

1.4 EXTRA ASSISTANCE FOR SALESPEOPLE

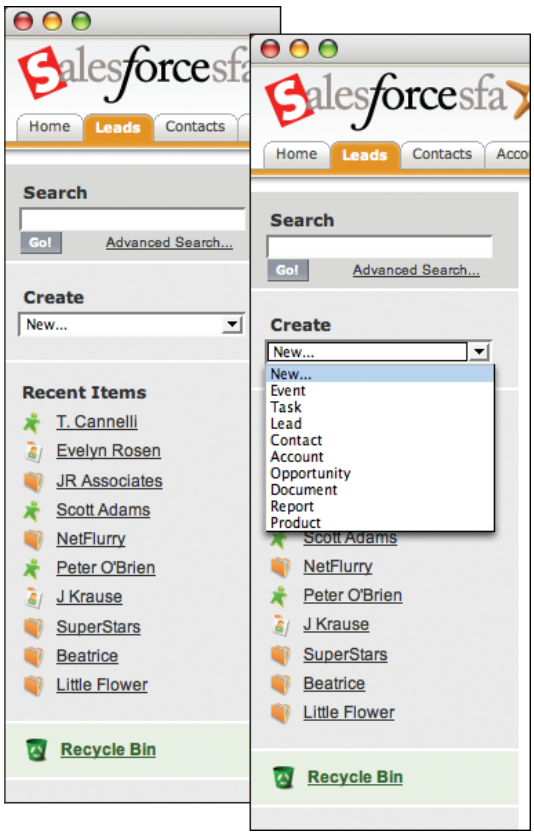

Figure 1-6: Left side controls in Salesforce

Each of Salesforce's screens has an identical set of controls on the left side. (Figure 1-6)

With the **Search** control, users search for any kind of item in the system, such as leads, contacts, opportunities, and documents. If users know what an item is called but can't remember where it is, or don't have the patience to click several times to get to it, they can always just type in the name and click **Go!**

Below the Search, we find the **Create** menu. No matter where users reside in Salesforce, they can always create new types of items. For example, users create an **Event**, **Task**, **Lead** or **Contact** no matter where they are.

Next, we see a list of **Recent Items**, which contains the ten most recent items users clicked on. The first list item is the Lead, **T. Cannelli**, then the Contact, **Evelyn Rosen**, and then the Account, **JR Associates**. Each recent item is associated with an icon. The lead, **Evelyn Rosen**, uses the address book card that we saw earlier in the page title in Figure 1-4.

Finally, there's a **Recycle Bin** with anything users have recently deleted.

Every Salesforce screen has this set of controls available. Salesforce's designers must have determined that these are commonly used commands. Also, despite the product's many hubs, Salesforce accounts for users who need to perform some tasks no matter which hub they are in.

Let's return to Saleforce's users. Salespeople must handle many contacts in a given day. Sometimes they contact customers by phone or email, and sometimes the customers will call them. It's a job with many partially completed tasks and an enormous number of interruptions.

The salespeople may be in the middle of a task when a customer calls. Since a potential sales lead is incredibly important, they stop whatever they're doing and begin recording information about the new customer. Saleforce's designers chose to put the **New** menu on the left where salespeople can always find it.

By placing Recent Items in the left-side control, the designers also make it easy for users to return to interrupted tasks, and to answer the question, "What was I doing?" Clearly, the designers know their users.

Not all applications will benefit from these particular controls on the left side. However, it's important to determine whether your application has any commands that users will always need available to them.

1.5 A STAGE TO STAND ON

Let's start digging deeper and look in more detail at the *stage*. The stage is the work area, the place on the screen where there are *no navigation and orientation controls*. When users click on the **Contact** tab, they see **Contacts Home** and the application dims out all of the navigation and orientation controls, leaving only the stage. (Figure 1-7)

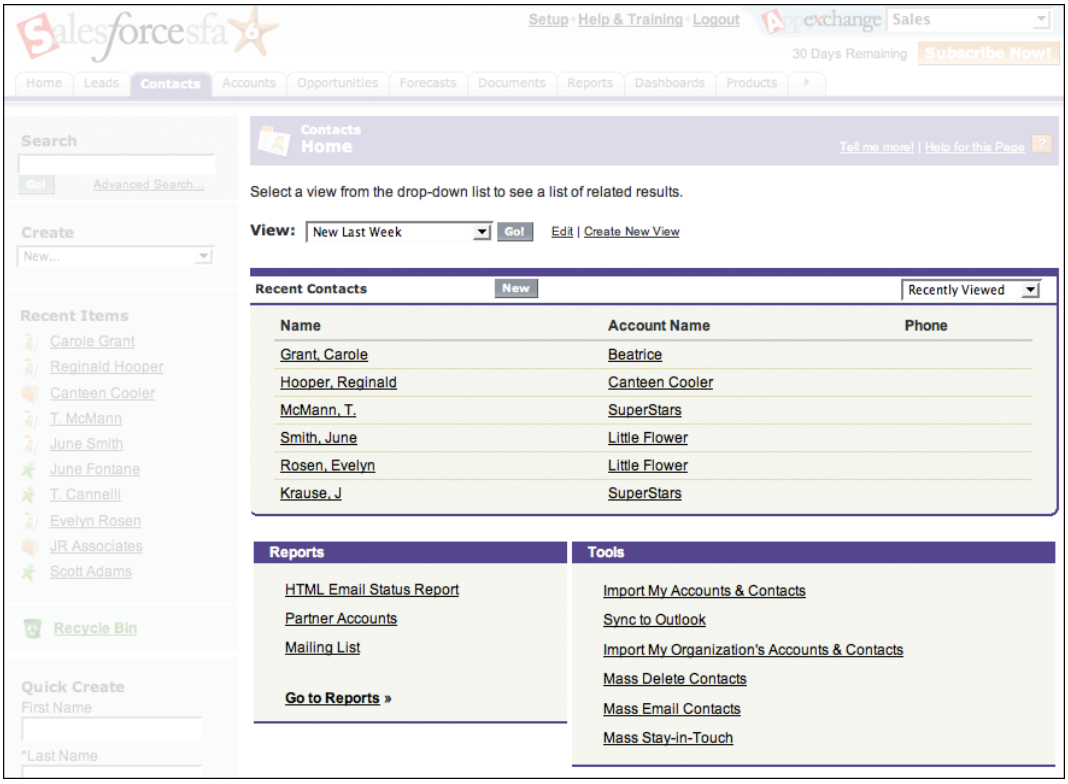

Figure 1-7: The stage where users can get work done

Users first encounter a **View**, where they can control what they see, and then a table listing **Recent Contacts**. At the bottom, we see two sets of links that take users to the other related product screens, **Reports** and **Tools**. This screen's visual design is somewhat weak. Nothing stands out on the screen because the table of **Contacts** looks too similar to the list of **Reports** and **Tools**.

> What about the stuff on the left? Isn't that part of the stage? It's a set of commands and certainly not orientation information. Both **Create** and **Recent Items** are arguably navigation controls. But since none of this information is related to the user's particular task, we don't think of it as part of the stage.

All of these elements make up Salesforce's stage. Because the stage is the users' primary work area, it's important that designers make it as large as possible. When designing a web application, it's inevitable that navigation and orientation controls will compete for space with the stage. While navigation and orientation are important, they serve only one purpose: to get users to the right stage for getting their work done.

In other words, users are primarily focused on the stage. Everything else on the screen is a necessary evil. An effective stage is large and the focus of attention on every application screen. The screen in Figure 1-7 has just 60% of the pixels devoted to the stage. (That's not too great, but we've seen some applications with as little as 30% of the pixels left for the users' actual work area.) For our designs, we try to allocate 75-80% of the screen's pixels for the stage.

1.6 A WHOLE NEW VIEW

Earlier in the report, we talked about Salesforce's excellent tab customization where the designers can create different tabs for different types of users. Salesforce's designers customized in another area, *Views*. Figure 1-8 is a **Create New View** screen.

In Salesforce, a *View* is the combination of a specific search and a collection of filters (to find and narrow down a group of items) and the instructions for how to display the results (which columns to show and in what order.) Users decide which results to display and how to display them. Some examples of custom views might include contacts a user created in the last week or any contact at the company, Little Flower.

In addition to creating custom views, users can share these views with other users in their Salesforce system. For example, members of one sales team can share the same views.

We've seen many web apps with customizable views. Designers are attracted to the flexibility and power they give users. Marketing and sales professionals also want the capability so they can tell a potential customer, "You can get the feature you want by just adding a view."

Unfortunately, we've found that very few users create and edit their own views. For users, the whole notion of a view is abstract and difficult to understand. Also, to create a view, users have to spend time tweaking the software. Creating a view distracts users from accomplishing their tasks, meeting goals, and doing anything even remotely fun. Users' only pleasure from making a view is that the software will be easier to use going forward and they can share their knowledge with other users. Very few people get a thrill out of this kind of work, except perhaps developers who love views.

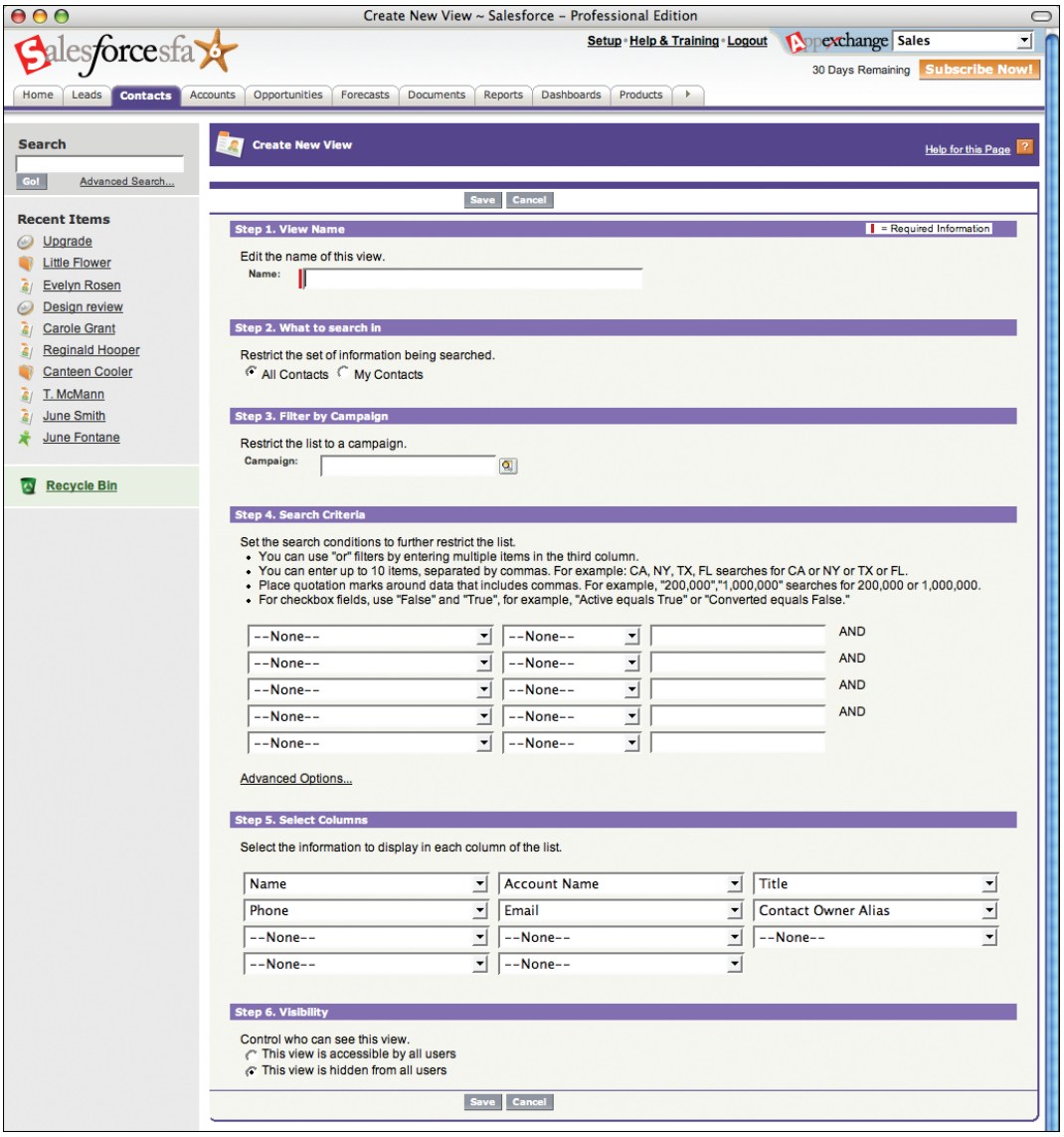

Figure 1-8: Creating a new view

Salesforce's designers tackle this problem by shipping the product with custom views for their users. Assuming the designers did a good job of creating the views, no user will ever need to fiddle with creating custom views.

If your application has different views, it's essential to determine the one, two, or even twelve views that closely address your users' tasks. We recommend you assume your users won't create any views and you should deploy with just the ones you consider important to their work. If you do include custom views, you shouldn't expect most users to use them.

1.7 THE POWER (AND WEAKNESS) OF TEMPLATES

Finally, let's discuss Salesforce's use of templates. As we have seen already, the application uses the same general screen layout repeatedly (such as in Figure 1-5). Yet, it also uses templates within a screen. Let's take a look at the **Lead Details** screen in Figure 1-9.

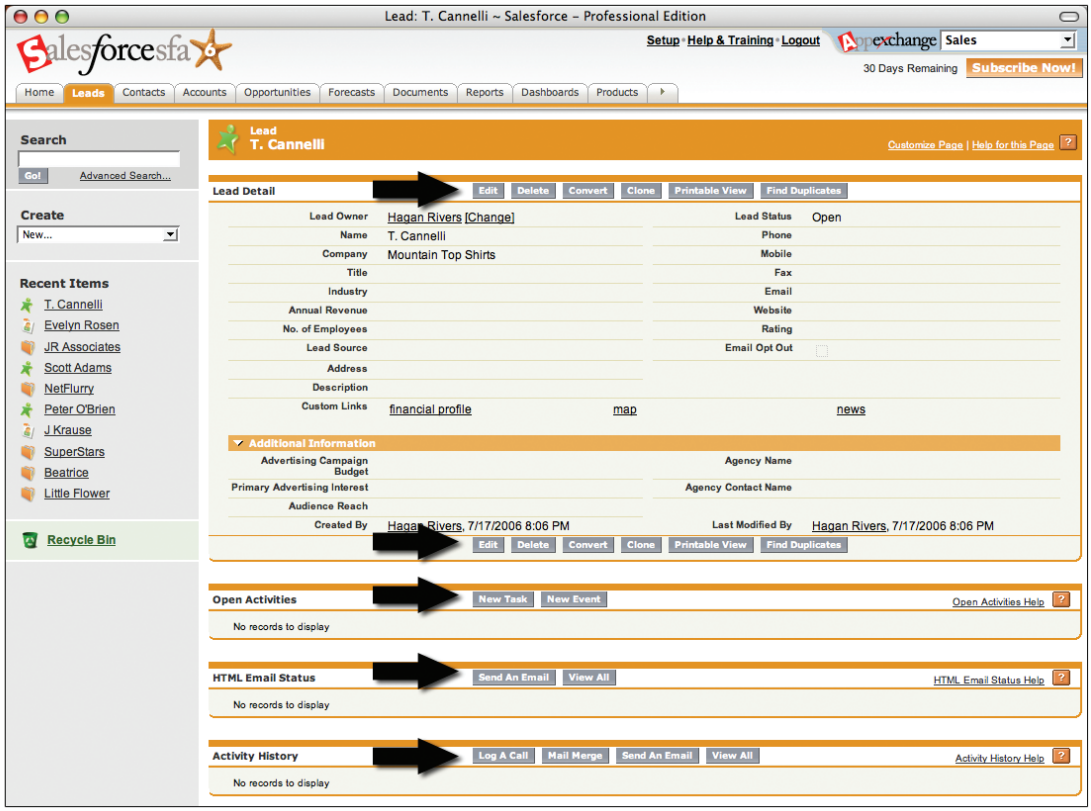

Figure 1-9: Toolbars on the Leads page

Along the top of the **Lead Details, we see** a series of buttons we refer to as a command toolbar. The toolbar includes the buttons: **Edit**, **Delete**, **Clone, Printable view**, and **Find Duplicates**. The toolbar commands operate on the contents of the **Lead Detail** section.

When we scan to the end of the section, we see the toolbar also displayed at the bottom. This is a good strategy if the contents of the section scroll off the screen. You'll notice that the section is surrounded by a box and has a title.

This is the template. It's repeated for each of the subsequent sections of the page. There's a toolbar for **Open Activities**, **HTML Email Status**, and **Activity History**. Templates are extremely powerful. When designers create a template, they need to consider all of the possible different types of information they may have to display and build one standard, consistent way to show that information.

From an engineering standpoint, templates are a blessing. Engineers can reuse portions of their code regardless of how many screens they add to the product. From a usability standpoint, a template can be a win too. Once users figure out how to work with one instance of the template, they can take what they learn and utilize it anywhere the product uses that template. Templates are very helpful for everyone.

Unfortunately, templates do have a downside: everything starts to look *repetitive*. One way that Salesforce's designers dealt with the repetition is to code each tab with a unique color. The colors help to break up the monotony.

> Saleforce's designers may have hoped users would connect a color with each tab in the product. For example, users might remember purple is Contacts. In reality, users don't typically make these kinds of connections. Instead, we suspect the designers used color to introduce some variety in the product. Because each tab contains a list of stuff, all of the tabs start to look alike. Color changes can help to introduce variety without a lot of extra coding.

Color change alone may not be enough to give the product visual design punch. Visually speaking, the product is very orderly, but potentially dull. This is one of the great dangers in creating and using templates. When all items in the interface look alike, how can users find things? How will users know which things are more important? They will have to rely on screen position and text labels to make these distinctions. If we conducted a usability study of Salesforce, we'd look to ensure users could find things easily and knew which things were important, otherwise we'd consider other approaches to changing the design.

Overall, Salesforce has an outstanding design. The application has many more screens than we've explored in this report. We highly recommend that you take a detailed look at Salesforce—it contains many design solutions that make it well worth the study time.

2 WEBOFFICE

WWW.WEBOFFICE.COM

WebOffice helps businesses share project-related information. Users can share files, calendars, databases, tasks, expense reports, contacts, and discussions, to name just a few. WebOffice has a wide audience, since its target users are members of workgroups from any type of business.

The product has a free trial available on www.weboffice.com.

WebOffice provides users with some of the same interface design elements as Salesforce: dashboards, templates, and tabbed navigation. We'll concentrate our review on areas where WebOffice and Salesforce are different.

2.1 SHOW ME THE OFFICE

Figure 2-1: The WebOffice home page

Figure 2-1 is the WebOffice **Home** page. Like Salesforce, the home page is a dashboard that aggregates key information (**Announcements**, today's **Calendar** items, **Tasks** due soon) in

boxes on the stage. Each of these boxes includes an icon, title bar, and controls to minimize or collapse the box. Users can control which boxes appear and where they appear through drag and drop.

WebOffice's designers have clearly worked hard to make the dashboard visually appealing, using drop shadows, colors, and icons.

One of a dashboard's great benefits is it has the potential to add sex appeal to any application. This is not merely eye candy. A well-designed dashboard makes a product appear weighty, powerful, useful, and usable. On the other hand, a poorly designed dashboard makes a product appear confusing and complex.

2.2 LEFT-SIDE NAVIGATION

Where is WebOffice's navigation system? We see a few links at the top of the screen (**WebEx Meetings**, **Web Folders**, and **WebOffice Advisor**) but they link to related products. The navigation system doesn't appear as tabs along the top of the screen, but rather along the left side. There we find commands such as **Home, Shortcuts**, and **My Favorites**.

Although they don't look like tabs, the left side commands act just like tabs. As a result, we refer to them as *stacked tabs* or *vertical tabs*.

Each command on the left side navigation has a little triangle to the right. Users may think that a menu will open up when they click on it, but they would be wrong. Although our complaint is minor, it's potentially confusing for WebOffice to include a triangle icon, a design element usually reserved for menus or expand/collapse controls.

Are left tabs or top tabs more effective? We've seen navigation systems in all of these locations work well. Both have their strengths and weaknesses. Let's take a closer look.

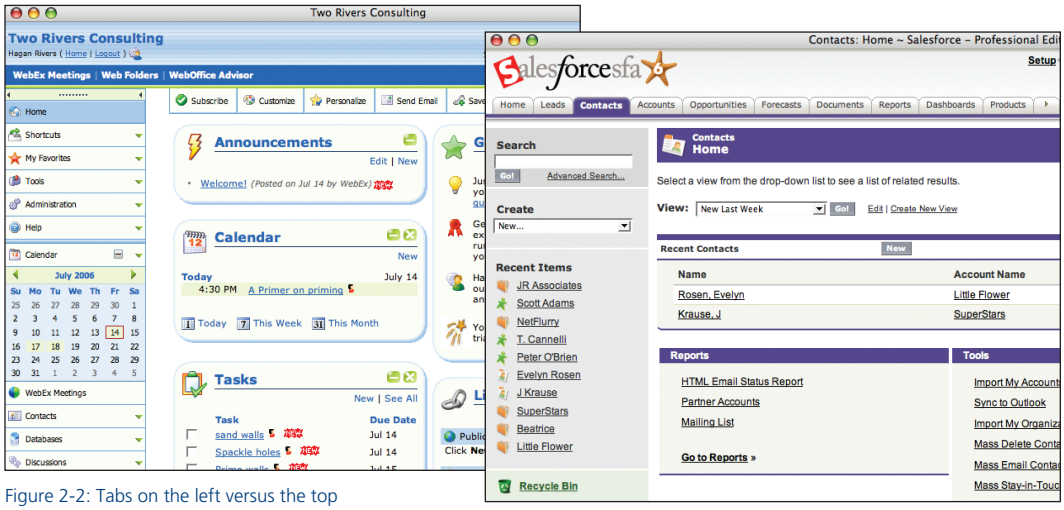

Figure 2-2: Tabs on the left versus the top

Left and top tabs differ in how they compete with the stage for space. Left side navigation takes away pixels from width, while top navigation consumes pixels from height. Which is better? That depends on the information designers choose to include in the stage and where they can afford to lose pixels. Left navigation typically consumes more real estate because it has more unused pixels or "filler."

Top tabs force the stage further down on the page, which increases the odds that important information may be hidden below the visible region on the screen. However, left tabs are often unworkable in an application with wide tables because it's possible they'll force users into horizontal scrolling, which typically isn't desirable.

Another difference: WebOffice's left side navigation elements are all the same width. In fact, the WebOffice designers chose to make their left navigation the width of their calendar item. If they had chosen not to include the calendar, the whole left navigation would be much smaller.

The designers sized Salesforce's tabs according to the text label inside it, which means they saved some space. However, because their tabs are stacked horizontally, they can fit fewer of them on a typical screen.

The top tabs lend themselves to sub-navigation much more easily—the sub-tabs appear in a second row below the primary tabs. Left tabs don't easily accommodate sub-navigation.

When you choose between top and left tabs, it's essential to consider the stage, the content that will appear on the screen, and whether you'll include sub-navigation.

2.3 LOOK AT THOSE TABS!

Let's take a closer look at WebOffice's tabs.

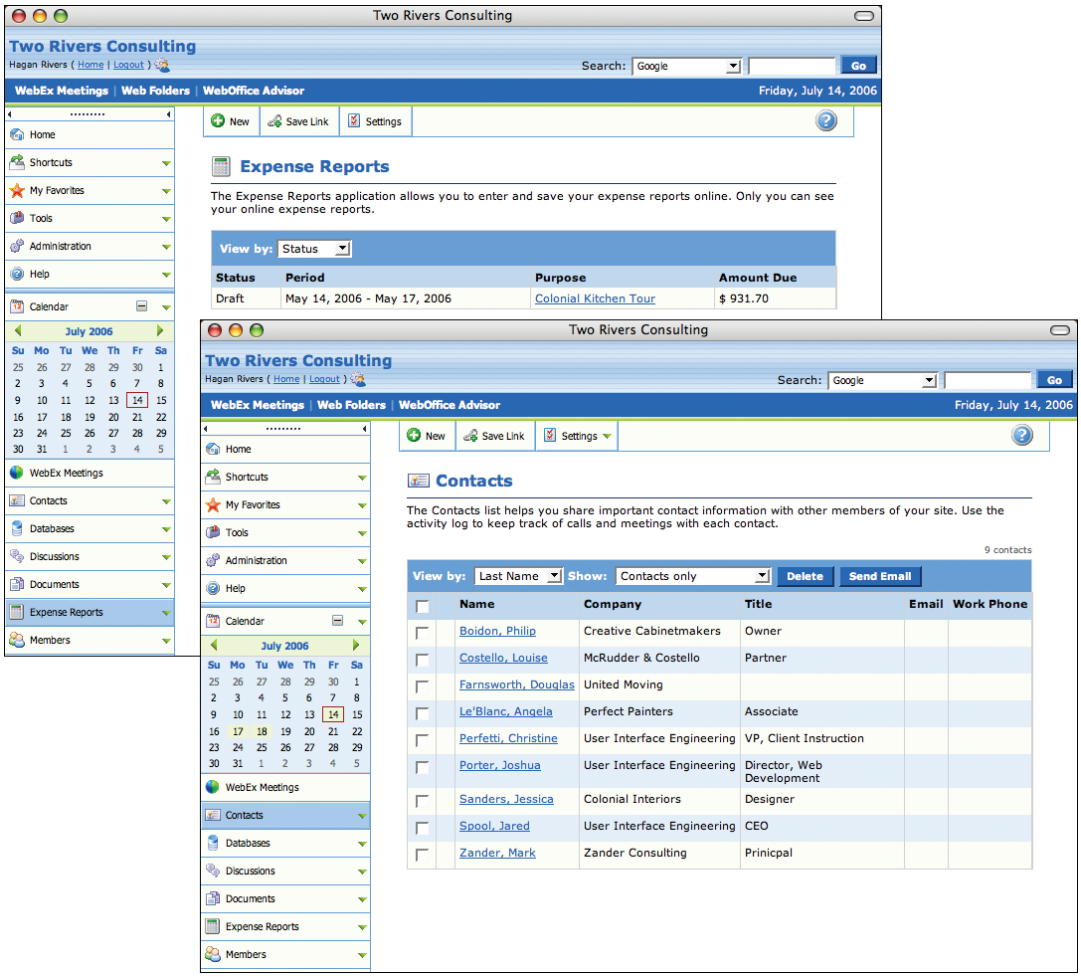

Figure 2-3: Expense reports and contacts

In the top left of Figure 2-3, we see the highlighted **Expense Reports** tab. In the bottom right of Figure 2-3, we see the highlighted **Contacts** tab. These are two of WebOffice's typical screens.

Each of the left tabs corresponds with one of WebOffice's 16 hubs. With 16 hubs, it's no surprise that the designers chose to work with stacked tabs. It's unlikely they could fit all of them across the top of the screen.

The WebOffice's team chose to visually break them into two groups. The first group of tabs is: **Home, Shortcuts, My Favorites, Tools, Administration,** and **Help.** Then we have the **Calendar** and then the next group of tabs: **WebEx Meetings, Contacts, Databases, Discussions, Documents, Expense Reports, Members, Polls**, and **Tasks**.

WebOffice effectively divides up the monotony of a long list of labels and helps users find the tab they need through *location*. The Calendar becomes a landmark on the screen—users remember that some commands are above it and some are below it.

We noticed one odd characteristic about WebOffice's tab design. The designers chose to display all of the tabs related to configuration and administration at the top, while they display all of the tabs users need for their work at the bottom. It's a questionable design choice because users rarely configure an application yet the tabs to access these controls are in the most valuable real estate on the screen, the upper left corner. If we were designing this application, we'd put the tabs for the most commonly used hubs in this area.

2.4 NOW YOU SEE IT, NOW YOU DON'T

At the very top of the left navigation, we find a thin button with a series of dots in it. (Figure 2-4) When users click on this button, the left navigation collapses horizontally, eliminating the tab labels and leaving only the icons.

If users find and click on this button, they gain back about 125 pixels of screen width for the stage. Earlier we mentioned that the stage and the navigation are often fighting for screen real estate. Web Office's strategy clearly reduces the area used by navigation controls.

However, the strategy only works if users realize they need more space, recognize the button, and decide to do something about it.

If users collapse the left navigation, they must rely upon position, icons, and hover-effects to remember each command. Some of these icons are fairly obvious, like the calendar. But it's not so clear if users will remember what tab to click for expense reports.

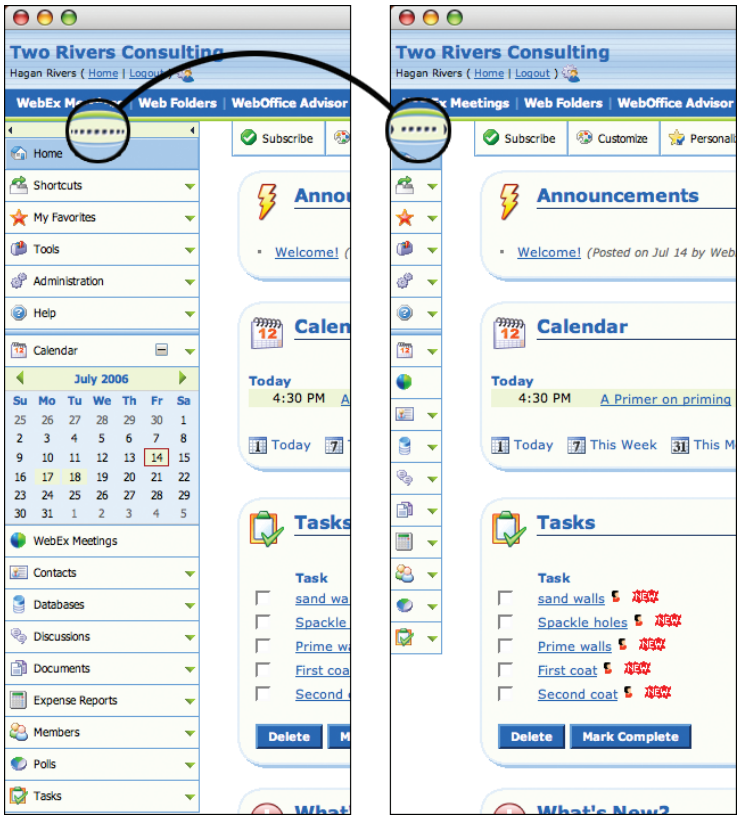

Figure 2-4: Collapsing the left navigation

In our design work, we've generally stayed away from these kinds of controls. We haven't seen evidence that users take advantage of them. However, we also haven't tested this application with WebOffice's users, so we don't know whether or not this feature is useful. It's certainly worth further study.

2.5 ASSISTING WITH DATA ENTRY

In Figure 2-5, we've selected the Tasks tab on the left side of the Tasks screen. This screen contains a list of tasks (which is presently empty.) Let's click on the **New** button at the top of the screen and open the New Task screen (on the right in Figure 2-5.)

> Now that we are viewing a child screen, let's look at how the orientation cues work. The **Tasks** tab on the left is still selected, although now we are on a sub-page within the Tasks hub. The Tasks icon is used in the page title, and the page title is **New Task**. Like Salesforce's approach to the same problem, the orientation cues are limited to changes in the page title.

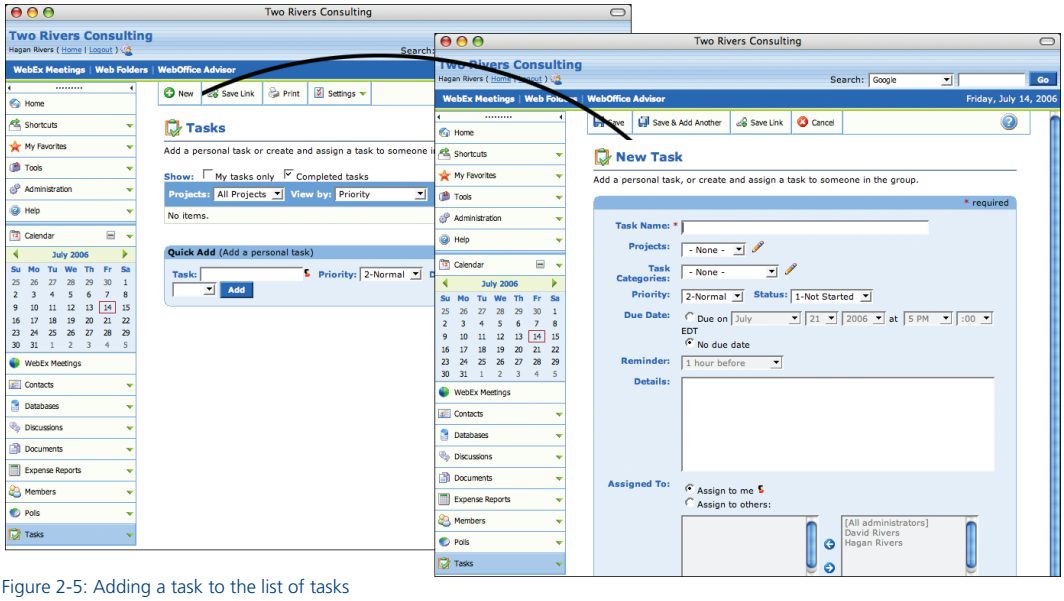

Figure 2-5: Adding a task to the list of tasks

What is a *task*? On the **New Task** screen in Figure 2-5, we can see all of the properties of a Task in WebOffice: **Task Name**, associated **Project** and **Category**, **Priority**, **Due Date**, **Reminders**, **Details** and **Assignments**.

When designers require users to perform data entry tasks, they need to make that process as easy and painless as possible. For example, one of Web Office's assets is it has good defaults for users. For instance, the **Priority** defaults to **Normal** and the **Due Date** is grayed out.

However, the due date is automatically set to be a few days from today, so when users interact with it, they have a good default. Setting good defaults saves users a substantial amount of typing and irritation.

WebOffice's designers also clearly planned out the data entry process. If users type in a task, they are likely to type in another. As a result, the designers added a command that saves users time. Whenever users add a new Task, they can simply **Save** (which will add the new Task and then return to the Task list), or they can **Save & Add Another** (which will add the new Task and then create a blank New Task page, ready to fill out.) Users appreciate anything that eases data entry.

2.6 A VIEW ON TASKS

Once we've added a few tasks, the task list starts to look a little more interesting (left, Figure 2-6). It's clear that the designers carefully considered the presentation of tasks. For example, new tasks are marked with a **New** icon, and the completed tasks have a strike-through effect, which is very natural and easy to understand. Overdue tasks are red. There are columns for due date and status, and users can assign tasks to other people.

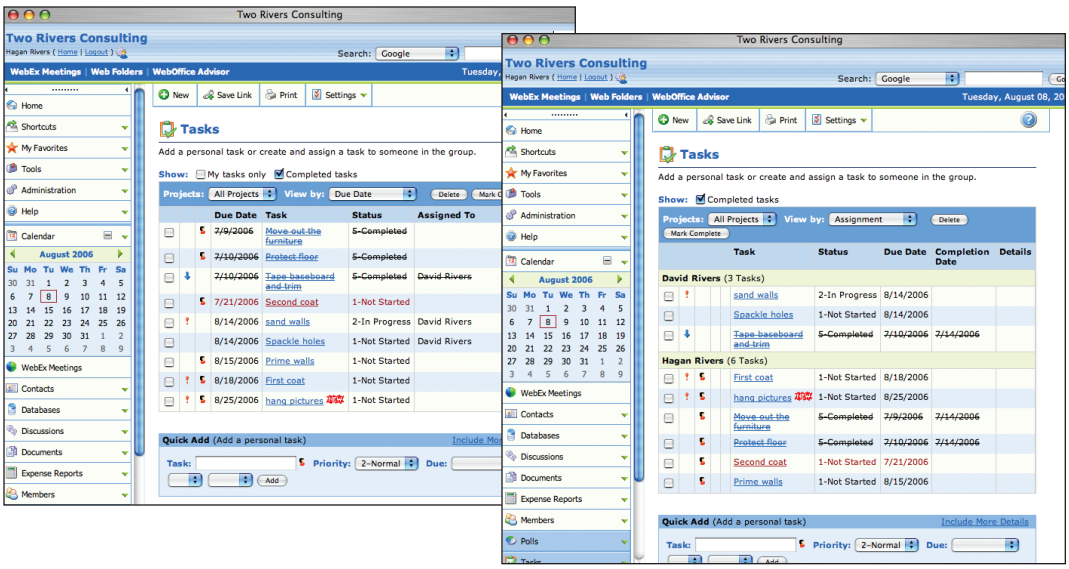

Figure 2-6: View tasks by priority and view by status

Remember earlier when we looked at the powerful (but potentially seldom used) View Builder in Salesforce? With it, users build custom views and share them with other users. WebOffice has a slightly different approach. They've made several pre-defined views of tasks, and they've crafted each to do its job well.

Let's begin by viewing tasks by **Priority** (left, Figure 2-6.) We see the highest priority tasks listed first. High priority tasks show a little red exclamation point, while low priority tasks show a blue down arrow. Normal priority tasks have no icon at all.

If we view by **Status** instead (right, Figure 2-6), the tasks are displayed in status order. The first group is **Not Started**, then **In Progress**, and finally **Completed**. Now that we are grouping by status, we no longer need the status column and it disappears.

The designers have also rearranged all the columns. The **Assigned to** column now appears immediately after the **Task** column, and within each group, they've sorted the tasks by **Priority**.

Why all the changes? Well, why would users ask to View by Status? Maybe they want to see the tasks they haven't started yet. If that's the case, they probably want the high priority tasks first. And it will be important to know which tasks belong to each user. Rather than just changing how they sort the table, the designers re-thought everything about the view and came up with the best view to help the users achieve their goals.

Clumping information in the list by categories (as we have **Not Started** tasks together here) provides more than just a change in view—it adds a different way of looking at the information. The designers created a unique, custom view tailored to assist users with specific goals of handling tasks. Users cannot add their own views, but as we discussed earlier, many users would never do this anyway. This approach is an excellent way to use views in a product and to make them useful.

2.7 ICONS

WebOffice makes use of many icons. In some ways, the icons help the design. But, in many places, we think they are overused.

Figure 2-7: Icons throughout WebOffice

As you can see in Figure 2-7, just about everything in WebOffice has an icon. We see icons in the left navigation, on each dashboard, and on the toolbar along the top of the home page. On the right in Figure 2-7, we see snippets from other screens in the application where the designers used icons in the toolbars and the page titles.

Designers use icons to help with orientation and to help users recognize the application's objects or commands. For example, the icons identify common things, such as a **Task** in Figure 2-7. When a Task appears in the left navigation, on a dashboard, or as a page title, the design always displays a Task icon. Icons are also part of commands, such as the **Save Link** button in Figure 2-7.

Unfortunately, users generally pay far less attention to icons than designers would like. Users don't memorize icons very readily and they don't spend much time studying them. As a result, you should consider using icons as visual reinforcement instead of as a stand-in for more meaningful text labels. Most of WebOffice's icons appear with text that provides the true meaning.

Even with the pitfalls associated with icons, they can still add value. Icons can help make a product more visually interesting, approachable, and usable. In certain instances, icons can completely replace a word in the interface. For example, users clearly understand the meaning of a **Calendar** icon. We could see it replacing the "Calendar" text label in WebOffice's interface.

However, we've encountered places where WebOffice's designers overused icons. Let's look at the Site Administration tab. (Figure 2-8)

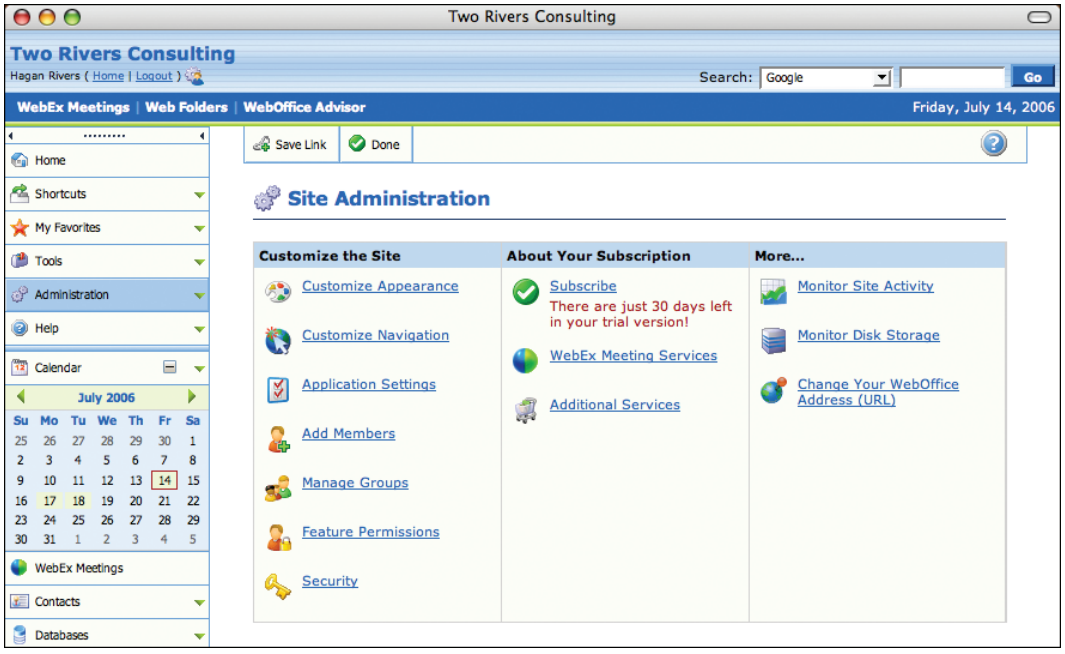

Figure 2-8: Icons on the Administration tab

WebOffice's Administration tab includes 13 different options for customization, arranged into three groups: **Customize this Site**, **About Your Subscription** and **More**. Each of these 13 options has its own icon associated with it, including **Customize Appearance** and **Customize Navigation**.

Does each link under Site Administration need a unique icon? In this instance, we believe the designers could have displayed the customization options as a neat, well-organized list of commands.

It's important to consider whether it benefits users to memorize these icons. Users only see these icons within Site Administration. Even if the user memorizes the icon for **Additional Services**, they won't encounter it anywhere else in the application. So, why are they there? The icons probably exist solely to stay consistent with the template. All of the other screens use them, so this screen does as well. Designers should be careful not to become slaves to their templates. Icons, when overused, only add clutter and confusion.

Using icons can be helpful and attractive in applications. However, you should take advantage of icons only when they add both beauty *and meaning*.

2.8 SHOULD I SAVE OR SHOULD I CANCEL?

Finally, let's talk about WebOffice's challenging problem of how to save user changes. In Figure 2-9, we see the **Save** and **Cancel** buttons in the toolbar on the **Customize Appearance** screen.

WebOffice uses an *explicit save model* for changes to the application's information. Users press the **Save** button to save changes or they press the **Cancel** button to leave the screen without changes. If users change the site name from **Two Rivers Consulting** to **Banana Split,** they must then click **Save** for the application to reflect the change.

Instead, if our user clicks on the **Home** button on the left, users lose the site name change when they go to the Home screen.

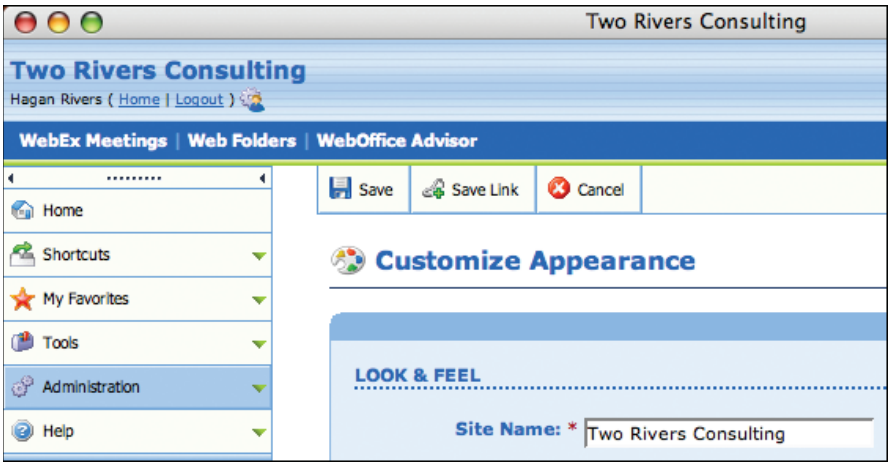

Figure 2-9: The Save/Cancel buttons in WebOffice

It's somewhat difficult to choose the right save model for a web-based application. WebOffice's method is similar to traditional desktop applications. When working in a desktop application like Microsoft Word, users must save that document for changes to take effect. If users try to close the document without saving, MS Word prompts them. This is what we refer to as an *explicit save model*. With this model, users need to remember to save. When they forget, the application prompts them.

When users interact with WebOffice, their experience is slightly different. In MS Word, users work primarily on one screen. However, WebOffice's users jump around from screen to screen. WebOffice's designers assumed that if users click **Home** without clicking **Save** first, their changes should automatically cancel. Is that right? Shouldn't they ask the user, "did you want to save your changes"? While asking the question may irritate users, losing their changes will likely frustrate them more.

Designers must weigh the likelihood of these events against one another. In WebOffice's case, users can only change a few things per screen. As a result, if some changes are lost accidentally, it's not the end of the world.

Another option is the *implicit save model*, where an application saves changes all the time. With this model, when WebOffice users type in the new site name, **Banana Split**, the application immediately updates and saves the site name.

When users makes a change and then immediately regret it, an implicit save model often makes use of the undo command. In some situations, a simple single undo is enough, but in others, designers may want to give users the capability to undo any change, reaching back in time through many different changes or actions. Later in this report, we will examine Writely, a web application, which does exactly that.

3 SERENATA FLOWERS

WWW.SERENATAFLOWERS.COM

Serenata Flowers (www.serenataflowers.com) is a British-based online florist designed for anyone in the U.K. who wants to order flowers online. When it comes to ordering flowers, customers have many questions: What's appropriate to send for a new baby? What should I send to my wife after backing her car into a post? When I forget an important birthday?

With a telephone order, florists guide customers through the purchase process and offer valuable advice and recommendations, including what type of flowers to buy and how much to pay. Let's examine how Serenata Flowers guides users through the purchase process with filtering controls.

3.1 WHAT DO WE BUY FOR BABY?

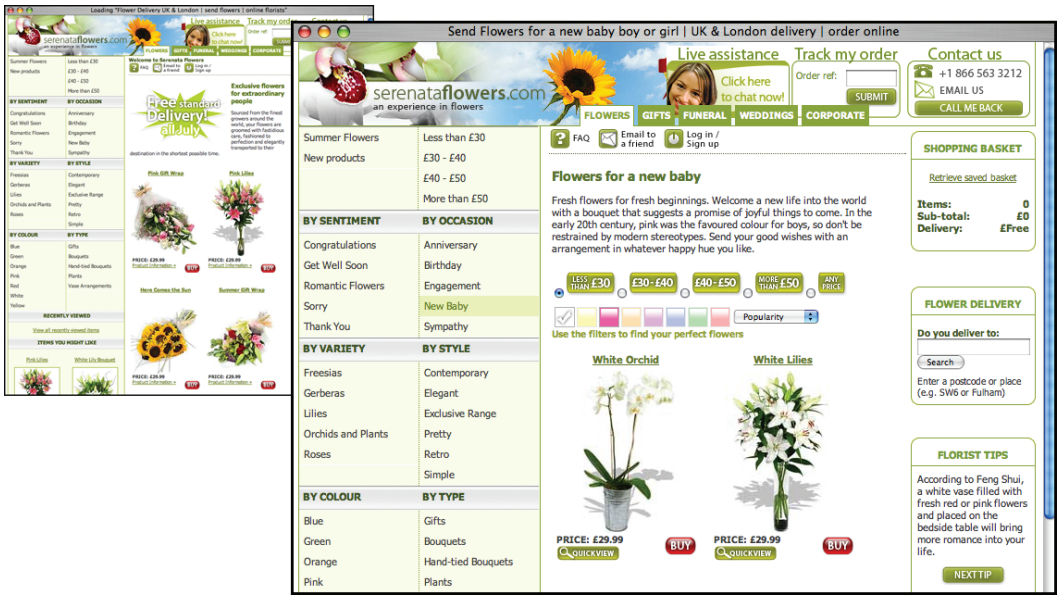

Figure 3-1: Finding flower arrangements at Serenata Flowers

Serenata's home page (Figure 3-1, left) displays sample flower arrangements that include flower photos, prices, and a **Buy** button. Compared to other flower sites, there's nothing exceptional or new to see here until you notice some less common controls on the left for filtering and displaying a subset of flowers.

Let's imagine a user needs some flowers for a newborn. Our user begins by clicking **New Baby** and immediately sees a reduced list of flowers on the right that are suitable. (Figure 3-1, right) Without a page refresh, the controls on the left filter the information displayed on the right. (Editors Note: This only works in some browsers.)

We've seen a number of web applications use this kind of filtering control: it's quick, easy to understand, and useful. However, Serenata Flowers takes this strategy even further.

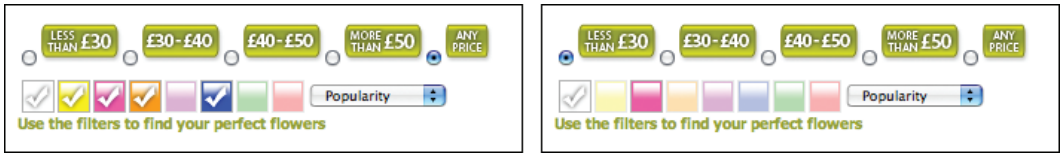

Figure 3-2: Serenata's filtering controls

Our user still has too many flower choices for New Baby and wants to further narrow them down. Serenata Flowers includes controls right above the flowers. (Figure 3-2, left) The first row shows prices: **Less than £30, £30-£40, £40-£50, More than £50**, and **Any Price**. The default setting is currently **any price**. Because our user only has £30 to spend, they'll want to eliminate expensive arrangements. When our user clicks the appropriate radio button, the application reduces the list of flowers to display only the arrangements priced less than £30. Jolly good!

Below the prices, we find a row of colorful checkboxes. These checkboxes correspond with flower colors. Our user likes white flowers and un-checks the squares until only white is checked (Figure 3-1, right.) The arrangements disappear from the list below until only white flowers priced less than £30 remain.

Serenata Flowers' filtering is very powerful and useful. Users can grasp how to interact with the filtering controls quickly because they can see the changes happening right in front of them by clicking and un-clicking the controls. We call this *dynamic filtering*.

Dynamic filtering changes the list of items in real time, updating the information as users apply each filter. Users can immediately see the results of applying a filter and make changes accordingly. It comes as no surprise, with these advantages, that users find dynamic filtering very satisfying. We expect to see much more dynamic filtering in the next few years of web application design.

3.2 FILTERS THAT REVEAL THE DATA

Another strength of filters is that they reflect the underlying data. In Figure 3-3 on the left, the flowers range in price from **£30-£40**. When we examine the checkboxes, we see that the application has already reduced the color choices with only five colors users can choose.

When users change the price range to more than **£50,** they only have three colors available to them. (Figure 3-3, right) In addition to un-checking the other choices, Serenata Flowers' designers have also disabled them. Users cannot check the options because they're not in the specified price range—it's a reflection of the data. We call this *indicative filtering*.

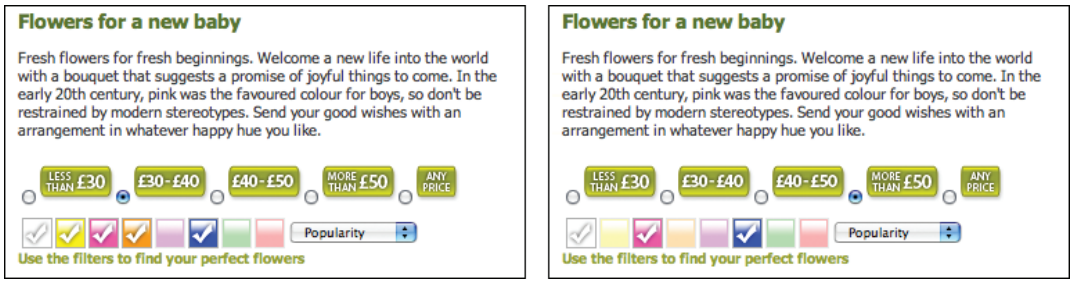

Figure 3-3: Depending on the price range, different colors are available.

Indicative filtering provides users with information about the data, answering such questions as, "how will applying this filter affect the data?" and, "how will it affect other filters?" When users understand them, these can be extremely powerful tools.

The designers of Serenata Flowers make excellent use of indicative filtering, yet they don't take it quite far enough. For example, wouldn't it be nice if users could choose flower arrangements based on color? Users could check the blue color chip and the price ranges would update to show only those ranges with blue flowers available. Unfortunately, with the existing design, if users want to see price ranges with blue flowers, they have to click on each of the price ranges and scan the checkboxes.

Figure 3-4: When should a filter be turned off?

One problem designers face when creating filtering interfaces is how to determine which filters should be in effect, and how to enable users to disable all or some of the filters and begin the task again.

For example, imagine that our user selects **By Sentiment: Romantic Flowers**, then **Pink** and **Red**. (Figure 3-4) Then, our user changes their mind and clicks on **By Style: Elegant**. Should the **Pink** and **Red** filters still be active? How can designers determine which behavior is more desirable? How will users know if the filter is active?

All types of filters face serious design challenges that require real creativity and user testing. We believe Serenata Flowers' controls are definitely a step in the right direction.

3.3 FILTERING ON THE HORIZON

When designing filters, it's critical to determine *the attributes users can filter.* Serenata's designers created a set of filters for a handful of quantitative attributes: **price**, **color**, **sentiment**, **variety**, **style** and **occasion**. However, their users cannot filter on the type of vase or the physical dimensions of the arrangement. (Maybe those attributes were too difficult to implement or they felt there were too many attributes already, making the design more complicated?)

Once you know the attributes, the next step is to provide controls to apply the various filters. For price, Serenata Flowers offers only a few price ranges, which is too bad. Many numeric filters are often too limiting and may not match users' needs. For example, with the existing design, users can't consider *any* flowers that cost less than $40. They have to click on the **Less than $30** radio button, examine the list, then click on the **$30-$40** radio button and examine that list too. If users want to consider flowers between **$25** and **$45**, it's not possible with Serenata Flowers' existing controls. It's important to consider *how users will want to filter.*

As web applications grow to manage larger and larger amounts of information, the quality of filtering controls will become increasingly important. Serenata Flowers' designers put forth a great starting point and their design can serve as a model for all of us.

4 BACKPACK

WWW.37SIGNALS.COM

Backpack helps users collect various types of information from different sources. To illustrate how Backpack works, let's imagine a bride-to-be, Sally, is planning her wedding. Sally decides to use Backpack to create one page where she can list all of her to-do items, keep notes about her progress, and store images and links to different things she finds in the process. With Backpack, Sally can share this information with anyone she chooses. We like to think of Backpack as a to-do list on steroids.

4.1 TO HAVE AND TO HOLD...

Let's see what Sally has in her Backpack.

We see four tabs across the top of this page: **Pages**, **Calendar**, **Reminders**, and **Writeboards**. (Figure 4-1) Sally is currently on the **Pages** tab.

Sally combined many different types of information on the page. The first thing is a **List**. She's using the list to show her to-do items, such as **Buy good shoes**. Further down, we see some checked items that Sally has completed, such as **Buy dress**.

After the list, we see some **Notes**, where Sally made a recording about formal versus informal weddings.

Further down is a **Writeboards** section containing a link to **Vows**.

Finally, at the bottom, we have **Links** to other Backpack pages, which include **Bride's Maids'** information, **Groom's Men** information, **Honeymoon**, and **Music**.

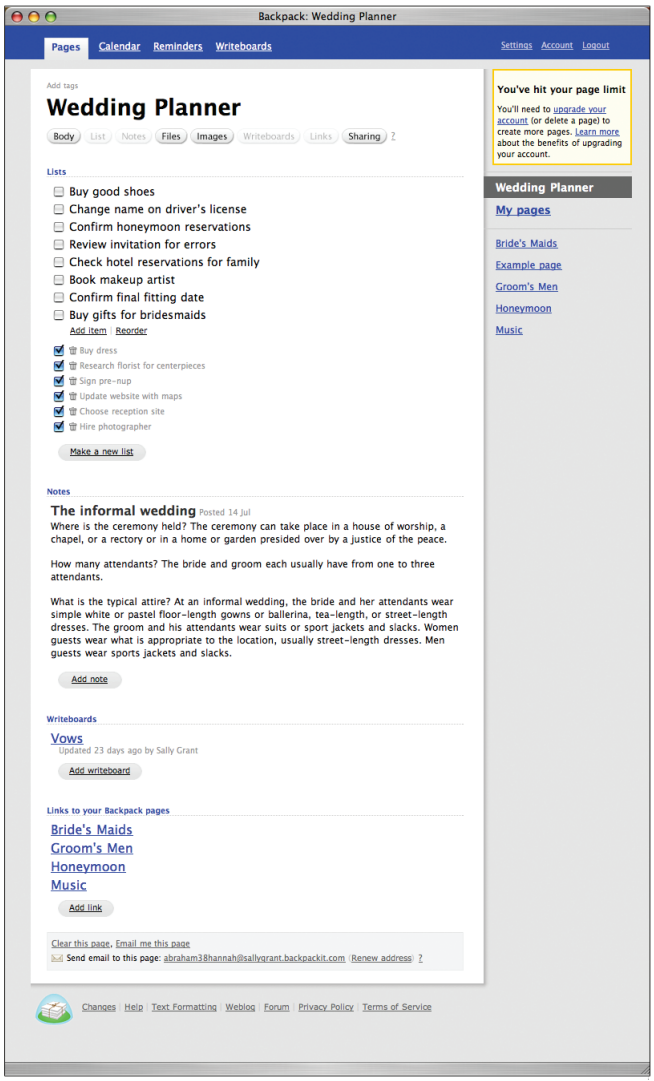

Figure 4-1: Backpack page for planning a wedding

On the top right-hand side, we see a box that warns **you've hit your page limit**. This contains information about upgrading Sally's backpack from the free version to the paid version. Below that, we see links to other pages, such as **Bride's Maids, Example Page, and Groom's Men**. This is a typical Backpack user page. Now, let's investigate what makes Backpack special.

4.2 KEEP IT SIMPLE

Backpack's designers took considerable effort to keep their design simple. At Two Rivers Consulting, our approach to making existing products appear simple is to focus on reducing the number of features. Backpack's clearly embraced this philosophy with a vengeance.

For example, Sally can't put her **Lists** just anywhere on the page. Lists always come first. She can add more lists and choose which of her many lists will go first, but all lists together will appear first on the page. As a result, Sally has limited choices about what she can do with the application. If users had the capability to choose where the Lists could go, Backpack would need more features.

Every time designers add a product feature, they pay a price: they have to design, build, test, and maintain that new feature. The users also pay a price. Each additional feature introduces more commands for a user to interact with and understand.

If an application has too many commands on a page, designers have the option to create menus and tabs to hide them from view and save space. But then our user, Sally, has to hunt around to find each command. Even a hip, internet-savvy user becomes lost and frustrated under these conditions.

The best applications avoid adding features, remove commands users hardly ever need, choose meaningful defaults (without letting users change them), and remove customization features that make things more complex.

Designers sometime argue users can ignore any extra commands they don't want. It's not that simple. In fact, it's a bit like overloading the dishwasher. When you add too many dishes, *none* of the dishes are cleaned. Similarly, when you add too many commands to an application, not only do users have problems finding the last few extra commands—users can't find *any* of the commands.

Backpack's designers embraced the principle: *Keep it simple.*

4.3 KEEP IT NEARBY TOO!

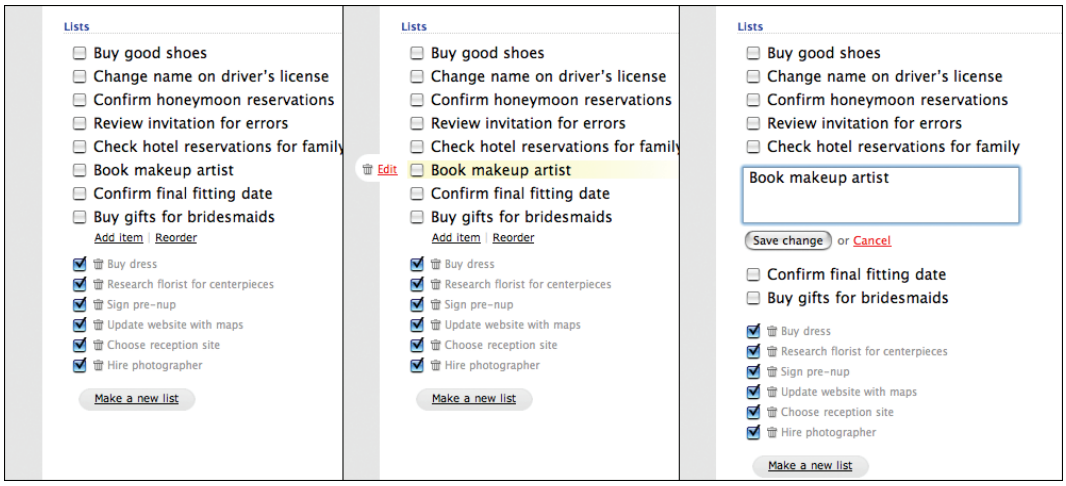

Figure 4-2 Commands appear near the items they affect

Backpack's designers chose to place commands contextually in the design. When a command affects an item, they place it right next to the item. As a result, Sally can add items to the list by going to the bottom of the list and clicking the **Add Item** link.

If Sally edits an item in the list, she just moves her mouse over the item and an **Edit** link appears. (Figure 4-2, center) When Sally clicks on the word **Edit**, an edit box replaces the read-only text and Sally makes her changes right there. Sally types in the new text and then saves her changes by clicking the button called, **Save change**. (Figure 4-2, right) By using AJAX, Backpack makes it possible for Sally to make the changes right on the page rather than in a popup window or dialog box somewhere else.

Backpack's designers take advantage of contextual design throughout Backpack. Let's look at another example. Sally can create multiple pages in her Backpack and link them together. If Sally goes to a new page, we can see these contextual commands in action, waiting for her to start filling in new information: **Add item**, **Start a new writeboard**, **link writeboard**, and **add link**. At the very bottom, you will find **Remove from sidebar**, **Delete this page**, and **Email me this page**. (Figure 4-3)

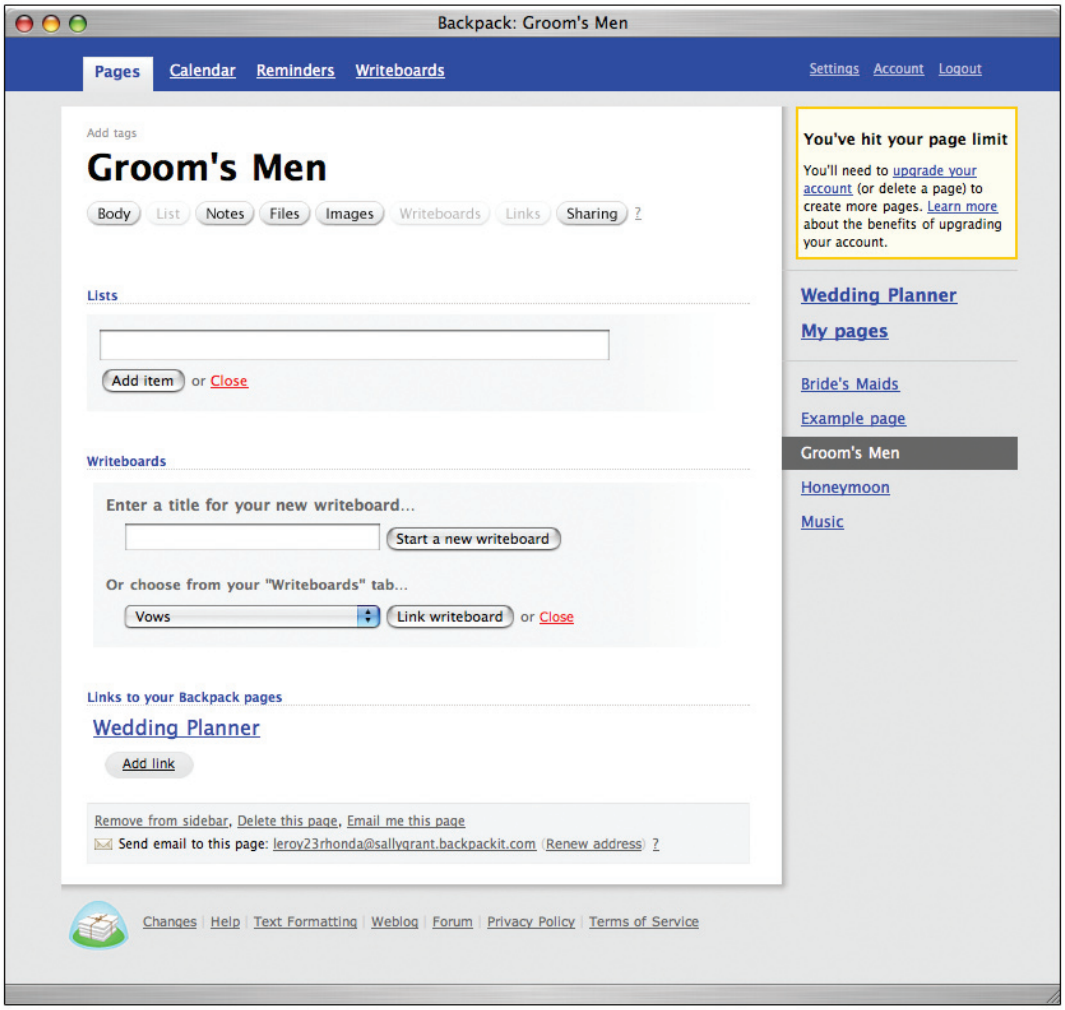

Figure 4-3: The page for Groom's Men

Sally is most likely to find the commands right next to the item she's working on, whenever she wishes to make a change. The commands she needs don't hide in a menu bar or in a toolbar at the top of the screen. The commands also don't have fancy icons—they are simple links and buttons, using short phrases to describe each command's function.

It's very difficult to design this way. When we contrast Backpack's custom page design with Salesforce's templates, every area on every screen in Backpack is slightly different according to that screen's information. Designers have to work hard to leave space for these commands, ensure the commands aren't getting lost among the items, and display only the necessary commands. While the design and implementation are tricky, the usability rewards are high.

While Backpack's design may not be suitable for every product, it's a promising approach. We expect to see many applications taking interaction design cues from Backpack in the future.

5 APPLE'S CONFIGURATOR

WWW.APPLE.COM

One feature of Apple.com helps users customize computer hardware when they place an online order. Many in the industry refer to this type of application as a *configurator*.

5.1 WHAT'S A CONFIGURATOR?

An advantage of configurators is users have the capability to tweak and alter the options for items they are thinking of purchasing. We've seen online configurators for most customizable items, including cars, bags, and custom order plastics.

When using a configurator, users can add or remove features and see how the changes affect the price. Configurators also help to reveal interdependencies. For example, if users configure a car, they can learn a red exterior only comes with the gray interior. As you might imagine, configurators can be very complex and sophisticated.

Figure 5-1: Apple's configurator

Apple's configurator is straightforward. In the top right corner of the screen, we see the summary for a currently configured computer. (Figure 5-1) The summary includes the current price of $1,699 and the specifications, such as the amount of memory and the size of the hard drive. We're at our starting point: the as-is, unconfigured computer. If a shopper likes this configuration, they can buy it without any further action.

The rest of the screen's real estate displays the components users can change about their computer. The first item on the list is **Memory**, where the configurator selects and highlights the **512MB** option. Apple's designers devoted extra time to emphasize the selection. The radio button is already on an option and the blue background draws attention to the selected memory.

Apple's designers also make good use of images to show users what they are configuring. This design choice is somewhat questionable. Does the average user know what memory looks like? Probably not. If we were conducting usability tests on Apple.com, we'd look to see if the pictures enhanced the sales process or if they were distracting to the users.

After Memory is **Hard Drive** and so forth, through each of the various configuration categories: **Graphics Memory, Modem, Software, Keyboard** and **Mouse**.

5.2 WHAT THE HECK IS THAT?

One of the tricky things about selling computers is that many buyers aren't knowledgeable about computer technology. They don't know RAM from ROM, how memory works, or how much they need. How can such users make intelligent choices when configuring a computer?

Under the category of **Memory**, we see a **Learn More** link. Thanks to DHTML and AJAX technologies, when users click on the link, they see a box containing help, without reloading the page. It offers the following easy to understand background information:

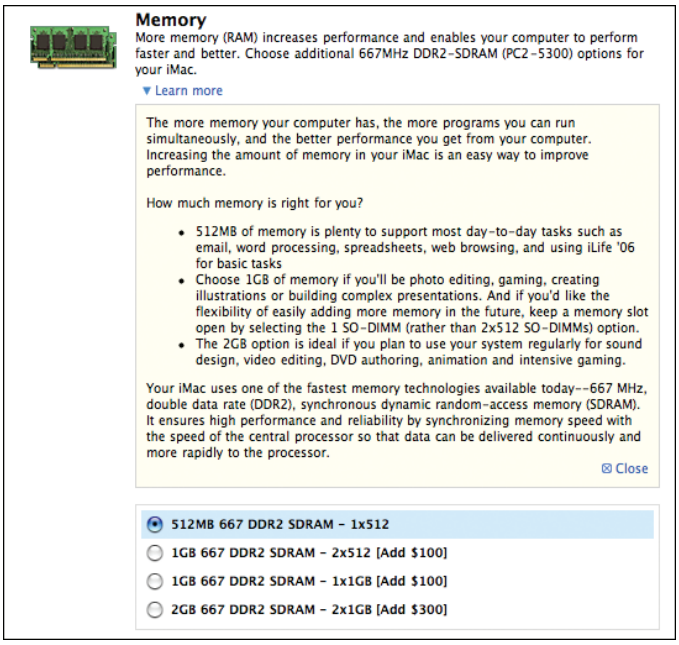

Figure 5-2: Help on choosing memory

> The more memory your computer has, the more programs you can run simultaneously, and the better performance you get from your computer. Increasing the amount of memory in your iMac is an easy way to improve performance.

It goes on to provide concrete advice for choosing between the different memory options:

> 512MB of memory is plenty to support most day-to-day tasks such as email, word processing, spreadsheets, web browsing, and using iLife '06 for basic tasks
>
> Choose 1GB of memory if you'll be photo editing, gaming, creating illustrations, or building complex presentations. And if you'd like the flexibility of easily adding more memory in the future, keep a memory slot open by selecting the 1 SO-DIMM (rather than 2x512 SO-DIMMs) option.
>
> The 2GB option is ideal if you plan to use your system regularly for sound design, video editing, DVD authoring, animation, and intensive gaming.

The text offers useful advice to users and assists them in making a choice. Teams often overlook high quality writing in the design of applications. Too many consumer products use copy that is so stiff and formal that it almost seems to come from a legal textbook. Still, others use copy so brief that it is vague, and others, so packed with marketing terminology that it's indecipherable. For successful application design, it's critical to choose the right words to effectively communicate with your users.

5.3 SHOW ME WHAT I'VE GOT

One of a configurator's best features is users can see how the product changes as they configure it. For example, when configuring a car, users can watch the exterior color change and see the car spoiler added to the rear. A computer's configuration is somewhat less obvious. The outside appearance of Apple's iMac stays the same regardless of how the user changes the configuration. However, what's inside does change and they've found a way to show that.

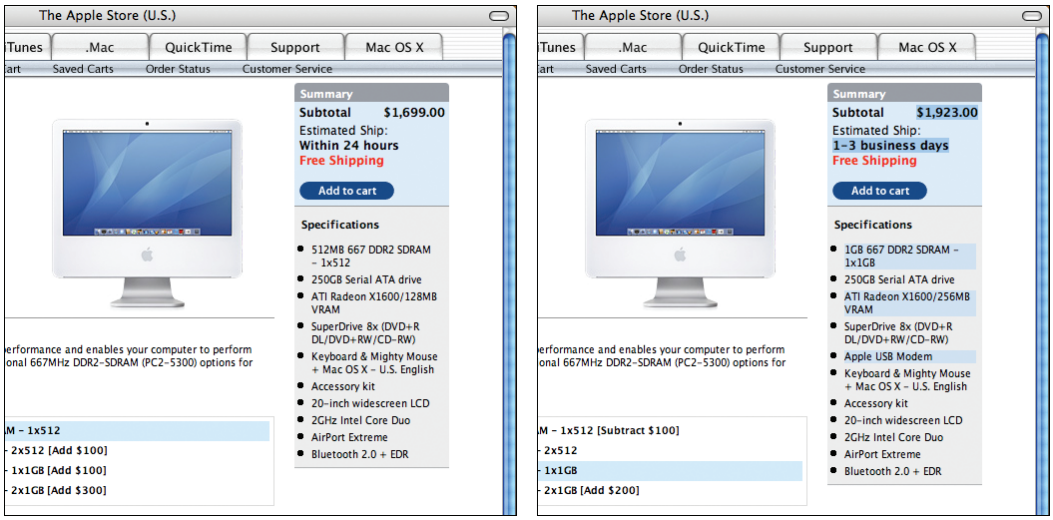

Figure 5-3: Adding upgrades updates the specifications and cost

Apple's designers have chosen to use the **Summary** area in the upper right corner to keep track of changes. On the left, we see the standard configuration. On the right, we've changed the memory, video card, and added a modem. (Figure 5-3) You can see that the price has changed, the estimated ship time is a little longer, and some of the specifications have changed. The list highlights each change so it's easy to find.

One missing detail from Apple's configurator is users can't click on an item in the summary and jump to it on the page. This functionality would enable users to make as many changes as they would like. It's possible, since the Apple Configurator is so short (2 screens), the designers concluded the feature wasn't essential.

In short, we think this is a great example of a successful configurator.

6 SURVEYMONKEY

SURVEYMONKEY.COM

SurveyMonkey is a tool for building online electronic surveys, collecting answers, and analyzing the results. The application is for anyone looking to gather survey data from a group of people.

Like Backpack, SurveyMonkey has a free version available with limited features. Monthly subscriptions are available to access the full set of features.

The SurveyMonkey application is fundamentally an *editor* of surveys. As an editor tool, users construct survey components any way they choose. Traditionally, most editors were desktop applications, such as Microsoft Word or Adobe Photoshop. Web applications focused on data management and manipulation, rather than free form editing. Up until recently, many designers and engineers felt it was just too difficult to build an editor using traditional HTML.

6.1 START FROM SCRATCH

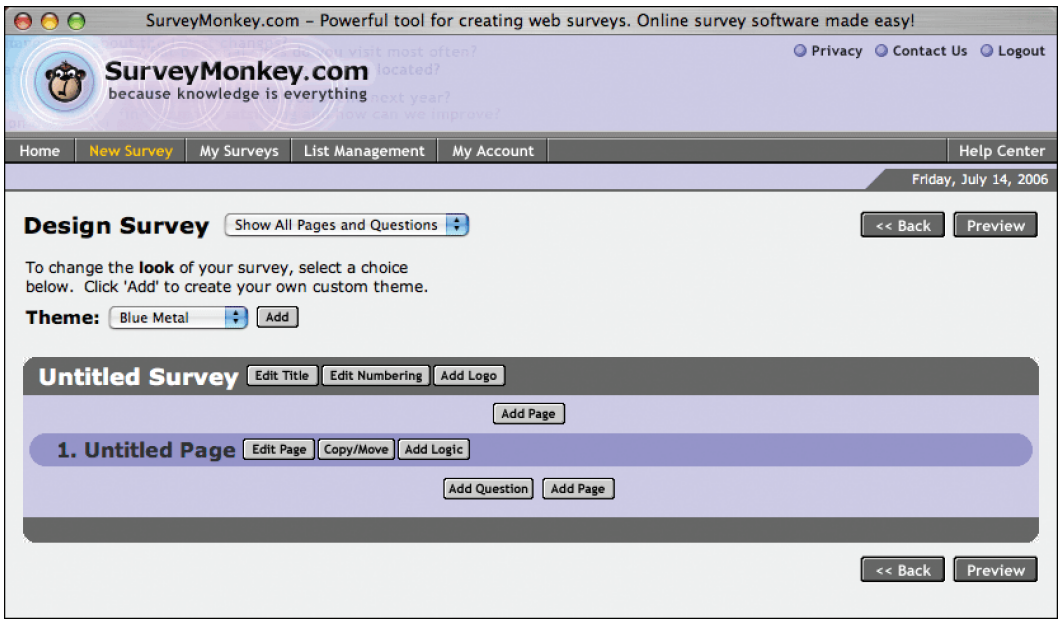

Figure 6-1: A new, blank survey

SurveyMonkey employs a very traditional WYSIWYG (What You See Is What You Get) model for building a survey. Users begin a new survey with an empty, untitled survey with no questions on the **Design Survey** screen. (Figure 6-1) Two elements of the survey are already in place: **Untitled Survey** and **Untitled Page**.

Next to **Untitled Survey** are a few buttons: **Edit Title**, **Edit Numbering**, or **Add Logo**. There is an **Add Page** button, and next to **Untitled Page** are three more buttons: **Edit Page**, **Copy/Move Pages**, and **Add Logic**. Finally, after the page, we have **Add Question** and **Add Page**.

Similar to Backpack's design, all of these commands appear in context on the screen. For example, to add a question, users look in the question portion of the page and click the **Add Question** button.

This characteristic is even more obvious when it comes to adding a page. SurveyMonkey actually has two **Add Page** buttons, one before **Untitled Page** and one after it. If users click on the first button, SurveyMonkey adds the new page *before* the **Untitled Page**. If users click on the second button, SurveyMonkey adds the new page *after* the **Untitled Page**. The great advantage to contextual commands is that they're easy to find and easy to decipher. The great disadvantage is that they can rapidly clutter a design and designers need to manage them carefully.

6.2 AJAX IS JUST A DREAM

The designers of SurveyMonkey built it when AJAX was still just a pipe dream. As a result, each button on this page takes users to a different screen where they can make changes. The **Design Survey** page is the *hub* of this application.

Figure 6-2: Changing the survey title with SurveyMonkey

Users begin with an empty survey. (Figure 6-2, top) When they click **Edit Title**, the **Edit Survey Titles** page opens. (Figure 6-2, middle) Each editing page is a *spoke* off the hub. Users visit the **Edit Survey Titles** page (a spoke), make changes, and then return to the **Design Survey** hub where they can see their changes in place. Let's look at an example.

We can change a number of things on this page, but let's assume we just change the **title** of the survey to, "What do you think?" then press **Update**. When we return to the **Design Survey** hub, we see the new title. (Figure 6-2, bottom)

> This page includes a **Back** button (for canceling) and an **Update** button for saving changes. In our designs, we don't like to label buttons, "Back", as this may become confused with the browser's back button.
>
> Many web applications break when users click the browser's back button. Therefore, we spend a lot of time ensuring that users never have to do that. Instead, we always have Cancel buttons on our screens. When we label those buttons as "Back," it may bring the users' attention to the browser's Back button and they're more likely to use it.

Users can stop working on the survey at any time. Each time they return to the Hub, the application displays the entire survey and users can always see the status of their work.

One interesting design exercise is to consider how you would change SurveyMonkey to use AJAX. For example, users could edit titles in place without visiting a new screen, just by showing an **Edit** link while hovering over the **Title**. Compare this design with Backpack, which we saw earlier, and Writely, which we'll look at later.

6.3 I HAVE A QUESTION...

Let's look at another example to see how a user adds a question. The process for adding a question is very similar to editing the title. Users click **Add Question** (top left, Figure 6-3) and go to the **Add Question** screen (top right, Figure 6-3), where they must choose the type of question.

Here again, we can see the designers place commands and help contextually throughout the design. If users aren't sure which type of question to choose, they can click the **type of question** link to find out.

Based on their choice, the screen updates using a form to gather both the question and, in this case, multiple choice answers. (bottom right, Figure 6-3) Once users press the **Add** button, they return to the survey and see their new question in place.

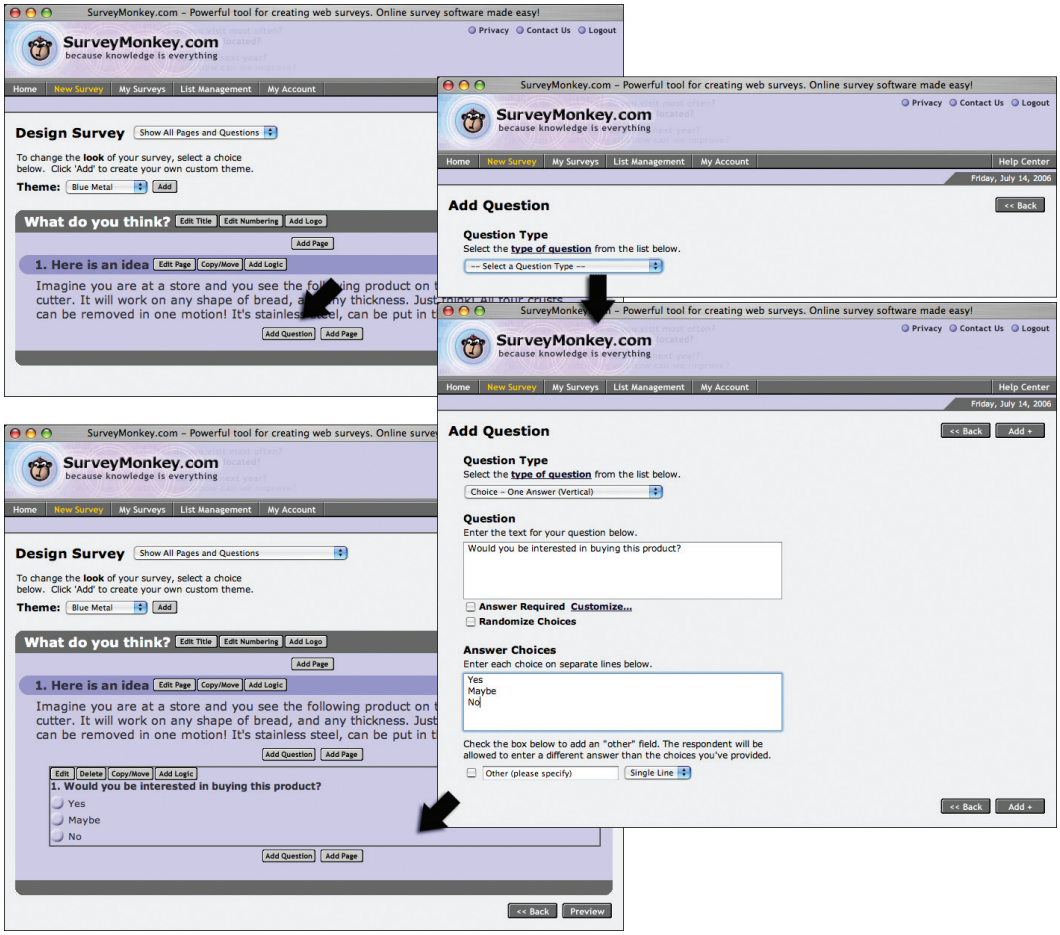

Figure 6-3: Adding a question to the survey

When users spend a lot of time building complex surveys, they'll quickly find the number of page loads to be time consuming and slow. While SurveyMonkey is a very good HTML-based application, the designers could choose to improve the process with AJAX on the **Design Survey** page.

By adding AJAX, SurveyMonkey will be speedier and improve users' rate of data entry. If the designers chose to use AJAX, when users click **Add Question**, a box could expand on the page showing the options for questions and allowing users to make the changes right there.

6.4 ALL MY SURVEYS IN ONE PLACE

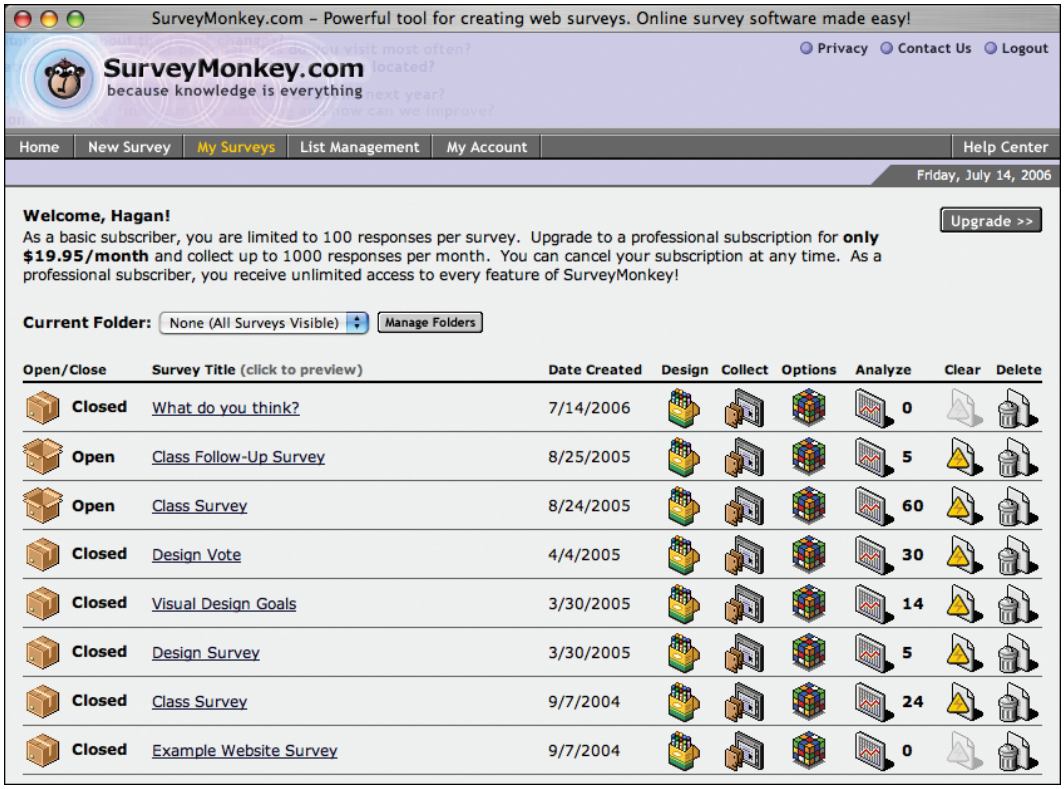

Figure 6-4: My surveys

When users finish work on the survey, they view it on the **My Surveys** tab, which shows both current and past surveys. All of the icons in this table are buttons that initiate an action. These buttons are (from left to right):

- **Open/Close** the survey: Users can make the survey available for collecting information
- **Design** the survey: Users open the survey editor again
- **Collect**: Users walk through the process of notifying other people about the survey
- **Options**: Users can specify how the survey data is collected (Will there be a password? What page should open when the survey is complete?)
- **Analyze:** Users can interact with the analysis tools
- **Clear**: Users clear the responses that they've collected so far
- **Delete:** Users remove the survey

The application's use of icons in the rows saves a great deal of space. However, we've spotted a few problems.

First, it's not obvious these are clickable actions. It's difficult to tell where designers used artwork as simply *status* (such as **This Survey is Closed**) versus when the artwork is an active *button* (such as **Close this survey**).

In SurveyMonkey, the icons serve both functions. The **Open/Close** icon shows the status of the survey and the **Clear** button becomes disabled when there are no results to clear. In general, when icons are clickable objects, they should appear clickable, with a button shape or, more subtly, a drop shadow. At the very least, we think button icons need to look different from status icons.

The second problem is comprehension. Without the column labels, users may not know what the buttons represents. Some, like trash can, may be obvious to users, but the box of crayons and the Rubix™ cube may be less intuitive (and possibly culturally specific.)

It's usually not a good idea to replace text entirely with an icon unless users clearly understand the icon's meaning. In this case, the icons don't really replace text, since text appears in the column headers. However, as the page gets longer and the column headers scroll out of view, users must decipher the icons without the text.

The third problem is the repetition. As the number of rows increase in the table, the repetition of the icons can become overwhelming. If users only have a handful of surveys (as shown here), this isn't a problem. Yet, when users have hundreds of surveys, displaying the same commands repeatedly in each row can be annoying and cumbersome for them.

The designers have several alternative options. They can use a text menu, which is clearer than buttons. It also means, if the SurveyMonkey team needs to add more commands, they would have the room to do so. The downside of menus is commands hide from view. If SurveyMonkey's designers add more commands, users may not discover them.

Another solution is to offer users a feature to select one or more rows and use toolbar buttons along the top to perform actions on the selected rows. This requires several clicks to finish an action, and it moves the commands further away from the objects they affect.

This is where user research comes in. If the research shows most users only have a handful of surveys in their list, then the design is probably fine as it stands, but with more commands and more items, it would rapidly break down.

6.5 SHOW ME THE MONKEY… I MEAN SURVEY

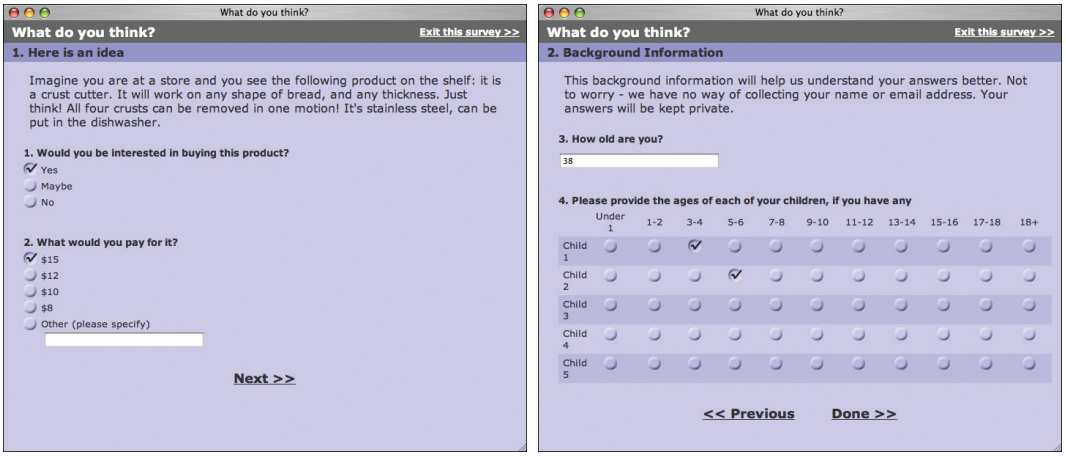

Figure 6-5: The final survey

With the application, users can preview their surveys and see them just as participants do. (Figure 6-5) They can review each question and answer them as a participant would, checking to see if all is well with the survey design.

With build-and-preview designs, one thing we always desire is a quick way for users to access the editor of the screen they are currently viewing. For example, with the existing design, if users are looking at the question "**How old are you"** on the right in Figure 6-5, and they want to change the question to "**What year were you born"** they have to exit the survey, open it in the editor, find the question, and make the changes there. It would be a lot faster if users had the capability to jump from the question in preview directly to the editor. This is another way that AJAX could really make this application sing.

6.6 ANALYZING THE DATA – THE RETURN OF FILTERS

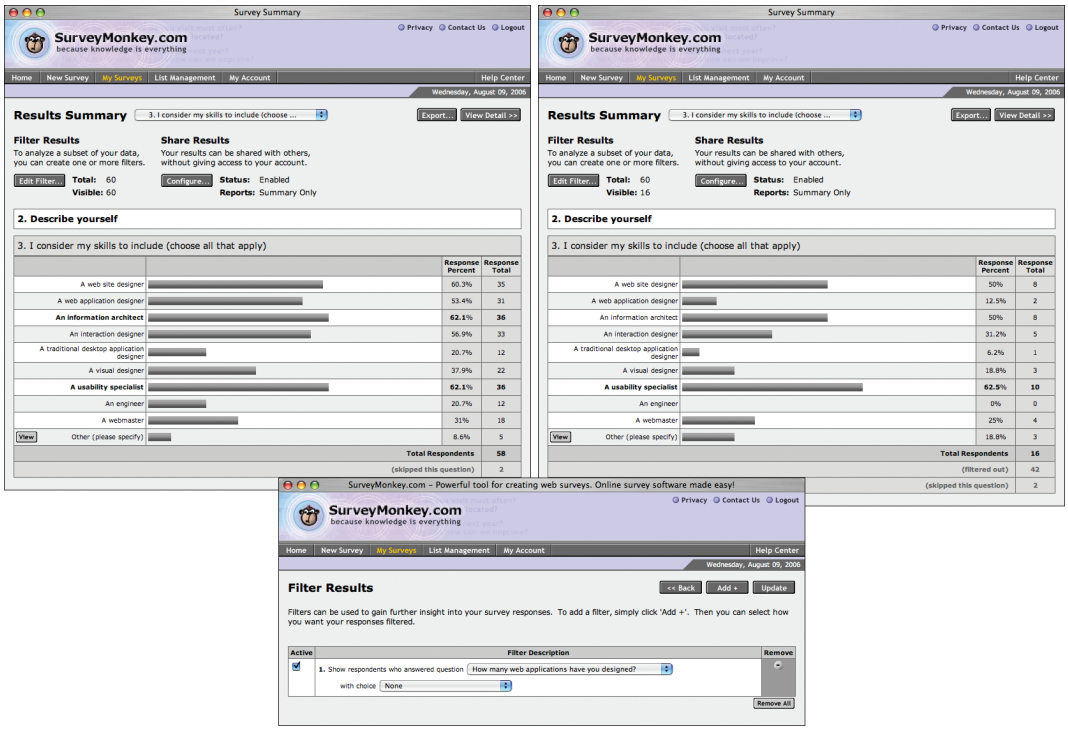

Figure 6-6: Analyzing results

SurveyMonkey also provides tools for analysis of the survey results. The application presents results in bar charts and other graphs, along with the numeric data, as shown in the top left of Figure 6-6.

> Bar charts have the advantage of being easy to create using simple HTML. SurveyMonkey did this by making a single-pixel wide image for the bar, then stretching it to whatever percentage they needed to display. However, we believe some better graphing and charting tools would improve the analysis and reporting process. SurveyMonkey's designers finesse this issue by allowing users to download the results into Excel and work their spreadsheet magic there.

One nice feature of the analysis is that users can filter the results. For example, in Figure 6-6 (top left), our user is viewing the answer to the question, "**I consider my skills to include".** When users first view this question, they see all the survey respondents. Maybe they want to know more about a specific group of respondents.

They could apply a filter to show them only those respondents who answered **"none"** to the question, "**how many web applications have you designed?"** (Figure 6-6 center, bottom.) Then, in Figure 6-6 top right, we can see that the data changed to show only the respondents that meet the criteria. Now they can answer the question: "For respondents who have never designed any web applications, what do they consider their skills to include?"

It can be hard to tell when a filter is in use here. The interface on the right only shows the effect of the filter: by changing the **Visible** results to be **16** rather than **60.** Of course, they also update the displayed data, but that is very difficult to spot.

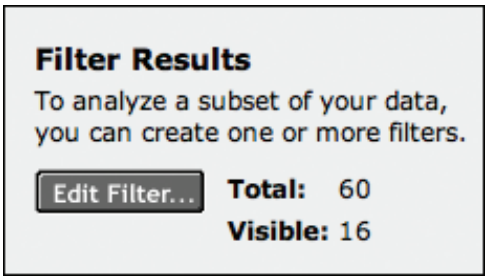

Figure 6-7: This is what indicates that a filter is in use

When there are substantial changes to the presented data, it's important to have a visible indication of the change. In this case, we recommend an obvious label that shows the filtering in effect.

SurveyMonkey has been around for years. As it has grown, it's showing its age a little, as we've pointed out in this section. Overall, this design has aged well and remains an excellent example of an HTML-based editor.

7 WRITELY

WWW.WRITELY.COM

Editors note: While we were preparing this report for publication, Writely's parent company, Google, re-released Writely as "Google Docs" (available at docs.google.com). At the time of publication, it's the same application as we've described below, just with a new name and a small number of visual changes.

Writely is a *web word processor*. It's also a *collaborative* web word processor. With Writely, multiple users can simultaneously edit and work on a document. The application targets a wide audience, yet most users have at least some experience with Microsoft Word or other desktop word processors.

7.1 JUST LIKE THEY USED TO MAKE 'EM

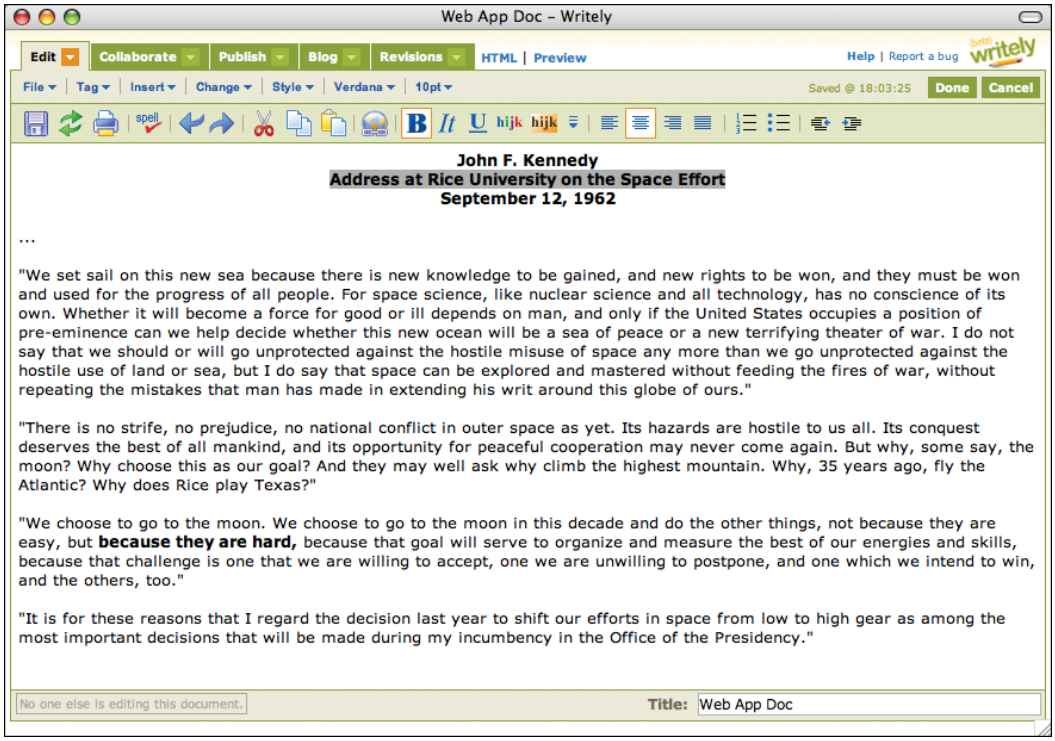

Figure 7-1: Editing a document on Writely

Figure 7-1 shows an example of a document that several users have edited. In many ways, the Writely editor looks very much like a desktop word processor.

Along the top of the window are five tabs: **Edit**, **Collaborate**, **Publish**, **Blog**, and **Revisions**. The current tab is **Edit**. In this figure, our user is editing a document.

Below **Edit** is a menu bar of commands: **File**, **Tag**, **Insert**, **Change**, **Style**, **Verdana** and **10pt**. This is the menu bar for our editor. This menu bar isn't the same as a typical desktop application with File, Edit, and View menus, but it's very close. Moreover, the different presentation makes good sense once users start to work with the application.

Below the menu bar, we see a toolbar of icons that use the same symbols we've seen in desktop applications for decades. These icons are familiar to users, which means that Writely's designers can avoid using text labels and save space. This is a perfect example of when icons can entirely replace text. Below the toolbar is the document itself.

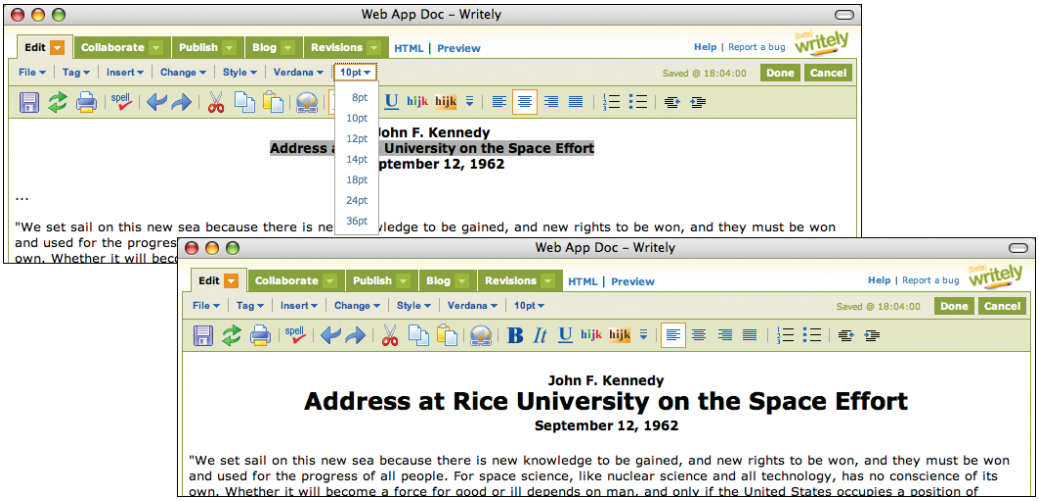

Figure 7-2: Changing the font size

The Writely interaction method is also familiar to anyone who has worked on a desktop word processor. For example, to change the font size of this text, the user first selects the text (text that has a gray background is selected), then opens the font size menu, labeled **10pt**, (see Figure 7-2), and chooses a new size. Writely then updates the text.

With applications like Writely, it's easy to see that the differences between a desktop application designer and a web application designer simply vanish. They are interaction designers—the only differences between desktop and web are the underlying technology and implementation.

7.2 POP GOES THE WINDOW

One thing we haven't talked about at all in this report is dialog windows or popup windows.

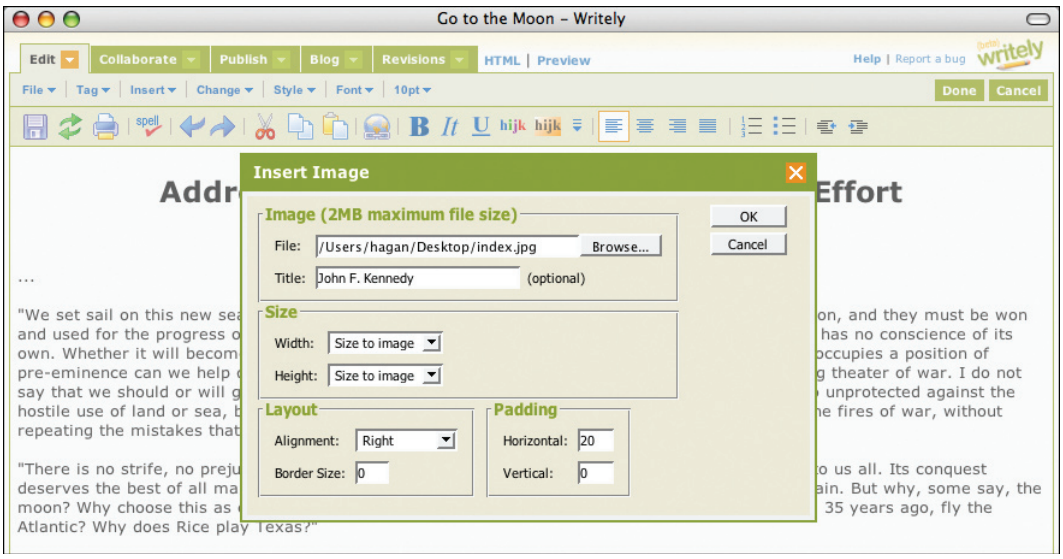

Figure 7-3: Adding an image

Let's see how users add an image to this document. First, they choose **Insert Image** from the Insert menu. Then, they encounter a popup window. (See Figure 7-3 where users can browse for the image file on their own computer and choose layout options for the image.)

The open window isn't actually a separate browser window – it's an HTML layer that appears on top of the editor. Notice that users cannot click on the editor, or the controls in the background, because the designers have faded out these options. Users can move the popup window as long as it remains within the browser window.

Why would the Writely designers go to the trouble of creating a special type of window in an HTML layer, rather than just opening the contents of that window in a new browser window? To be blunt, that's not how a *desktop* application works—a desktop application just displays a new popup window. There are a number of good reasons to use a layer instead of a window, even though there are some engineering challenges along the way.

It's true that desktop applications bring up new windows to complete commands, and Writely is clearly trying to duplicate the experience of using a desktop application as much as possible. In design terms, these are *dialogs*. If users insert an image into a Microsoft Word document, they expect to find that image using the file dialog, a separate window.

However, dialogs in desktop applications work very differently from web browser windows in three key ways:

- In a desktop application like Microsoft Word, the dialog boxes are associated with the application. If the user minimizes Word, the Word dialogs minimize along with it. If users bring the Word application to the front, the Word dialogs come to the front with it. The desktop window manager supplies this behavior free, by tracking all of the open windows in all applications.

- The desktop allows designers to create different types of dialogs. One key dialog type is the *modal dialog*. A modal dialog requires that users finish working in the dialog before they can return to the main application window. Therefore, when users are looking at the insert file window, they must either choose an image to insert or cancel the action before they can continue editing.

- These different types of Word dialogs always stay on top of other Word windows. In this case, it would be impossible to place the **Insert image** dialog *behind* the document itself.

When you open multiple web browser windows, though, the desktop window manager doesn't perform as well:

- The web browser does not have any concept of an application. Users who have a web browser open may have multiple browser windows. Some of these windows might be from your application, and others might be from a web site that they're viewing. As designers, there's no way to get the web browser to treat all of the windows associated with your application as a group. As a result, it's easy for your application's windows to be tangled up with the users' other browser windows.

- There's no built in concept of a *modal dialog* in web browsers. For example, users could say, **Insert image,** and then move the dialog aside and return to editing. What happens later on when they go back to the dialog and click, Insert? That's a sticky design problem!

- There's no way to keep some windows in front of other windows, so it's relatively easy for users to accidentally bring the main editing window on top of the **Insert image** window, potentially obscuring it.

For all of these reasons, and because of those pesky things known as popup-blockers, it's better to present dialogs within the application's single browser window as a layer.

7.3 MENUS EVERYWHERE

Writely's designers use a lot of menus in their design. Menus are a wonderful tool for saving space and grouping controls together. However, they also have a dark side. It's much more difficult for users to select menu commands than buttons or links. Menus require users to click down to open them, move their mouse with the mouse button held down, and release to choose the command. This requires significant concentration on the part of users.

Secondly, menus *hide* commands. While this is the intended purpose of menus, users still must know where to *find* the commands. If designers want their users to find commands in menus, they need to name the menus carefully and place them in context, where they make sense.

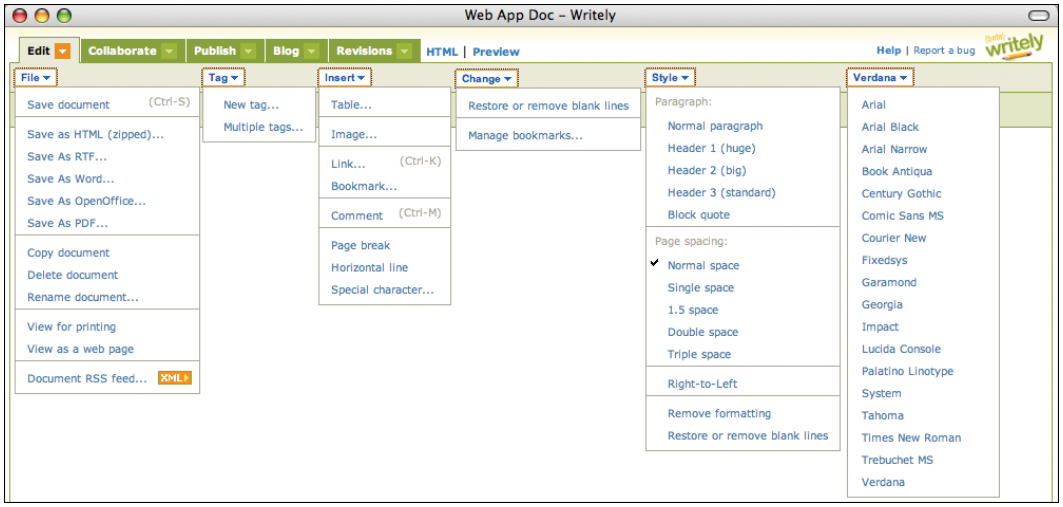

Figure 7-4: The menus of Writely

In Figure 7-4, we see the menu bar for the **Edit** tab in Writely: **File**, **Tag**, **Insert**, **Change**, **Style**, **Verdana** (the font size menu isn't shown.)

Within the File menu, we can see the first item is Save Document—and it has a keyboard accelerator. How's that for imitating a desktop application?

Notice that the designers used lines to separate groups of commands. In the **Style** menu, they even have labels for these groups: **Paragraph** and **Page spacing**. HTML provides us a lot of freedom in how we present menus and the Writely designers have used that freedom to create an excellent design.

We are delighted to see that the designers carefully avoided submenus because they can be difficult to use.

We think this is a top-notch menu design.

7.4 REVISIONS

Writely's designers have embraced the *implicit saving model*. Earlier in the report, we lauded the benefits of implicit saving. When designers create an implicit save model, the application saves the users' changes *as the user works*. Users never have to click a Save button. The model's danger is, if users do something that they don't like, they must go back and undo those changes.

Figure 7-5: Saving often

Each author edits the document in a *session*. During a session, the author's changes are automatically saved every 30 seconds. (Figure 7-5) Users can also explicitly save. At the end of each session, they can decide if they are happy with their work. If they are satisfied, they press **Done**. If not, they press **Cancel**.

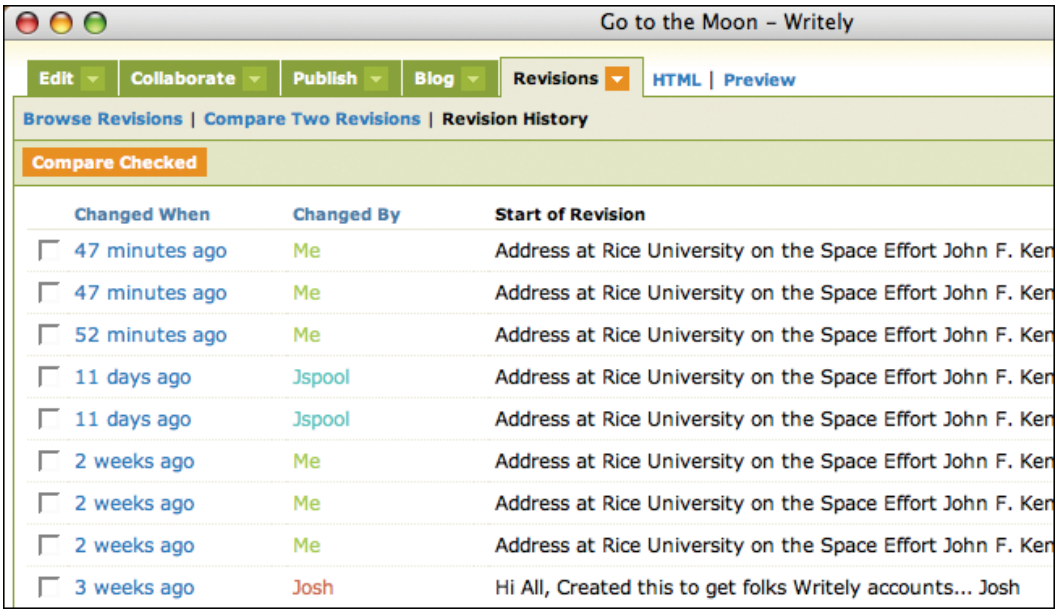

Figure 7-6: Revisions to this document

Writely keeps the user's work on a list of revisions along with the revisions of other authors. (Figure 7-6) Users can see who has made the changes, when they made them, and where the editing began in the document. If users want to open an older version of their document, they click on the link under the **Changed When** column.

How can users undo changes? It's a complex question—simply going back to the version **2 weeks ago** probably isn't enough. Users may want to grab an image or a paragraph from an earlier version and bring it into the new version. That's why Writely's designers allow users to pick any two revisions and compare them. (Figure 7-7)

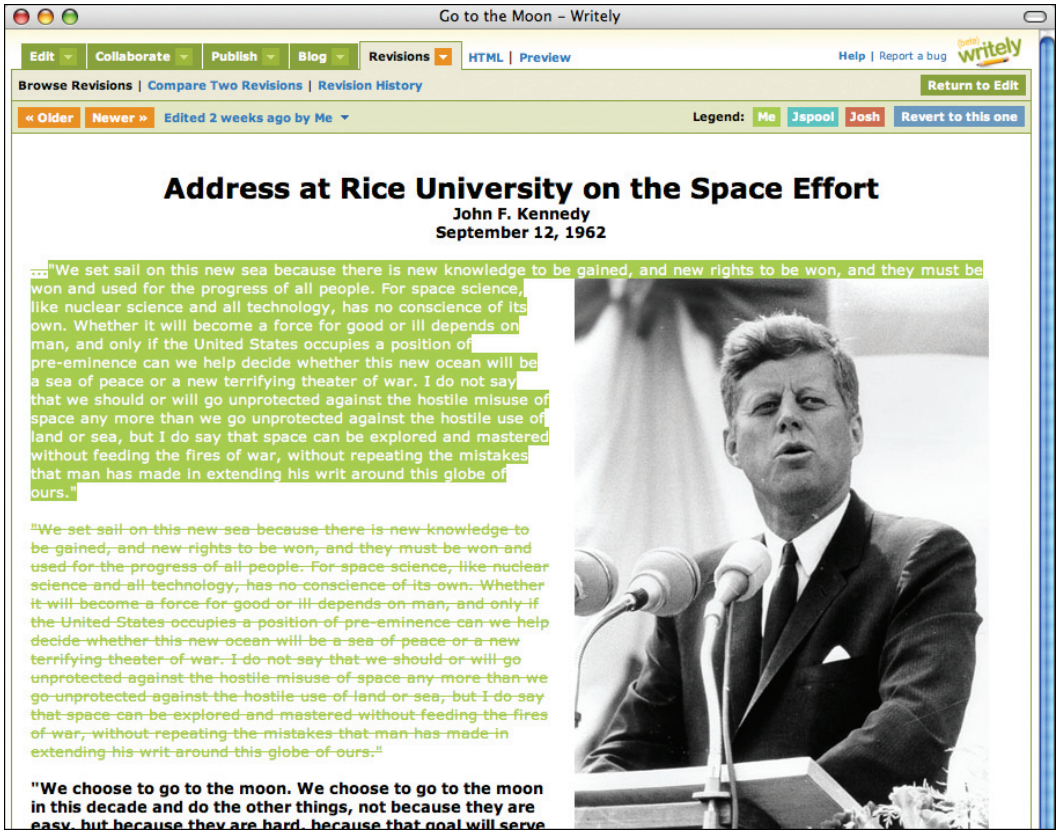

Figure 7-7: Comparing revisions

Writely's designers have done an effective job with their implicit save model. They never discard users' work. They can always revisit and delete changes and recover text. In addition, they never need to worry about remembering to save because the software does it for them, making it easy for them to recover from mistakes.

Because disk space is so inexpensive these days, it makes sense to keep much more of the users' editing work. The trick for designers is to build recovery systems that users can understand easily.

7.3 ALL MY DOCUMENTS

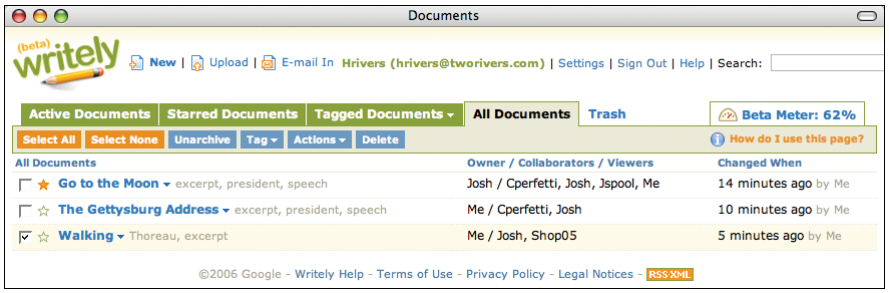

Figure 7-8: A list of documents

Writely keeps all of the authors' documents in a list. Users see this page when they first log in to Writely and before they begin editing documents. In this list, users see the **Active Documents** (the ones being actively edited), the **Starred Documents** (marked important), the **Tagged Documents**, and **All Documents.** (Figure 7-8)

Using Writely's first three tabs, users can filter the list in the **All Documents** tab. For example, when a user clicks on the **Starred Documents** tab, Writely filters the table to only include those documents marked important. Just like Salesforce and WebOffice, Writely effectively provides a set of prepared views for users.

Figure 7-9: Actions on documents

The list of documents is a table showing the document's **title**, **tags**, **owners**, **collaborators**, **viewers**, and when it was **last changed**. From here, users can select one or more documents and click the toolbar buttons to **unarchive** a document, **tag** it with a keyword, perform an **action** such as **save** or **star** it, or even **delete** it.

This list is Writely's *file system*. Just as a computer has a file system for organizing documents, Writely's users can organize the documents they've created. It's interesting to see how different this is from a traditional file system design and to think about what the desktop might be like if this kind of design were applied there. Then again, many things about Writely make us stop and think about the future of web application design.

AUTHOR INFORMATION

HAGAN RIVERS, TWO RIVERS CONSULTING

Hagan Rivers is a pioneer designer of web applications who has worked on several of the web's most sophisticated applications. Hagan's desire to design complex and elegant applications has placed her at the cutting edge of interaction design for each of the 15 years since receiving her Computer Science degree from MIT. She was the lead designer of Netscape from version 1.0 through 4.0, simultaneously designing on Windows, Mac, and Motif and in HTML. Hagan profoundly affected both the design of the internet and design for the internet.

Hagan worked on some of the very first web based interfaces and she continues to push the envelope of web application design in her current role as a partner at Two Rivers Consulting. She has designed and improved upon software for the general population, healthcare and insurance professionals, analysts, retailers, scientists, system administrators. Her clients have included large and emerging companies, as well as nonprofit organizations and philanthropic causes.

Hagan has been a top-rated presenter at dozens of speaking engagements around the world, including her many appearances at UIE's User Interface Conference.

DAVID RIVERS, TWO RIVERS CONSULTING

David Rivers is a visionary web application designer who passionately believes in the power of simplicity and understands just how much time and effort it takes to create a simple design.

David received an Engineering Psychology degree from Tufts University. He worked at Sun Microsystems, Apple Computer, and Netscape before founding Two Rivers Consulting 10 years ago.

David has a keen interest in the artistic side of web applications and he delights in taking ugly duckling applications and making them usable, useful and beautiful. He worked on the design of some of the earliest and most complex web applications, including the design of the Netscape Server. David has consulted with companies and organizations ranging in size from one or two employees to large corporations.

THE UIE FUNDAMENTALS SERIES is published by

User Interface Engineering
510 Turnpike Street, Suite 102
North Andover, MA 01845
(978) 327-5561
(800) 588-9855 (within the U.S. and Canada)
www.uie.com

COMPANY BACKGROUND

User Interface Engineering is a leading research, training, and consulting firm specializing in web site and product usability. Jared M. Spool founded the company back in 1988 and has built User Interface Engineering into the largest research organization of its kind in the world. With our in-depth research findings based on user observation, we empower development teams to create usable web sites that increase customer satisfaction and loyalty.

PUBLICATIONS

UIE Reports: Our in-depth reports detail the latest happenings in the world of design

UIE Brain Sparks Blog: UIE's place to share our latest research and musings with you. We'll be sharing our latest ideas and observations in the hope of sparking the same in you.

Brain Sparks Audio Library: Audio recordings of the latest thinking in design, including the new SpoolCast recordings: intruiging sessions with today's top thinkers about user experience.

UIEtips Email Newsletter: Our free e-mail newsletter highlighting our latest research, products, and public speaking engagements.

TRAINING EVENTS

UIE Web Application Summit—Monterey, CA—January 21-23, 2007

If you enjoyed Hagan Rivers' report, you'll want to come hear her present her full-day seminar, Deconstucting Web Applications, at the UIE Web Application Summit. For this three-day event, you'll meet the pioneers and world-class designers behind today's most successful web apps and come away inspired to create amazing applications that will delight your users.

UIE Virtual Seminars

Here is your opportunity to learn the design, information architecture, and usability insights used by today's most successful web sites. UIE's monthly Virtual Seminars will give you the chance to **hear the latest perspectives in the world of design from the field's premier experts.**

Instead of traveling to a training course, you and your colleagues can hear the latest insights on the most important design topics right from your office.

User Interface 12 Conference—Cambridge, MA—November 5 – 8, 2007

Take your organization's web design and usability practices to the next level. At User Interface 12, we've invited the most knowledgeable experts to offer practical design tips. You'll spend 4 days learning advanced techniques from top design, information architecture, and usability experts.

Learn more at http://www.uie.com.

User Interface Engineering
presents

UIE Web App Summit

Monterey, California
January 21– 23, 2007

WebAppSummit.com

Premier experts from Yahoo!/Flickr, Netflix, eBay, H&R Block and Adaptive Path will be joining us. How about you?

Web-based applications are quickly becoming critical components of the strategy of many organizations. However, the knowledge and skills to make a great application is still only available as scattered lore. *Until now.*

You'll meet the pioneers and world-class designers behind today's most successful web apps and come away inspired to create amazing applications that will *delight your users.*

Day 1 Full-Day Tutorials: Features Hagan Rivers' **full-day tutorial**, Deconstructing Web Applications: Learning from the Best Designs

Day 2 Foundations: Topics include Design Patterns, Web App Structure and Flows, Form Design Best Practices, and Communicating with Comics

Day 3 The Latest Perspectives in Web Apps: Speakers include Stewart Butterfield of Flickr, Sean Kane of Netflix, and Christian Rohrer from eBay

Expert Speakers:

Hagan Rivers
David Malouf
Bill Scott
Larry Constantine
Peter Merholz
Brandon Schauer
Kevin Cheng
Matt May
Luke Wroblewski
Stewart Butterfield
Sean Kane
Thomas Vander Wal
Jim Whitney
Chris Whalen
Rashmi Sinha
Christian Rohrer

Plus UIE's own,
Jared M. Spool
Joshua Porter

Until January 16th, you can register for all three days for only $1,999.

UIE WEBAPP SUMMIT

Visit WebAppSummit.com for more details and online registration!