ISeB (2006) 4: 361–393
DOI 10.1007/s10257-005-0024-3

ORIGINAL PAPER

Mark S. Silver

# Browser-based applications: popular but flawed?

Published online: 16 February 2006


**Abstract** Browser-based applications (BBAs), applications built on top of web browsers, dominate the world of Internet Applications today but are fundamentally flawed because the web browser is a weak platform for applications. Three characteristics of the browser—page orientation, statelessness, and limited computation—combine to produce a set of practical problems for BBA users. These problems include delays and discontinuities, confusion and errors, clumsy interfacing and limited functionality, printing problems, and filing difficulties. The paper analyzes these usability problems, providing numerous examples and tracing them back to underlying browser characteristics. The paper also examines the factors that make BBAs so popular despite their flaws. The paper concludes by considering directions for understanding the phenomenon better and for improving the current state of Internet Applications.

**Keywords** Browser-based application · Internet application · Weblication · Web-based application · Usability

Why are so many e-commerce web sites so bad? Many factors contribute to the problem, but even were we to transcend them all, the fundamental reality is this:

Today's web browsers are the wrong software for e-commerce.

How can that be? Have not millions of shoppers used their browsers to make billions of dollars worth of purchases on-line? Is not browsing, in fact, an important part of shopping?

M. S. Silver
Fordham University, Suite 604C 113 W. 60th Street, New York, NY 10023, USA
e-mail: msilver@acm.org

Yes, web browsers can be great for finding and comparing products. After all, a key component of traditional shopping is browsing and one of the advantages of e-commerce is the ease of browsing multiple offerings. But purchasing has two components: searching and transacting. And the transactional aspects of buying a product are a poor fit for web browsers. The mismatch between applications' needs and browsers' capabilities leads to numerous practical problems. Consider the notice the Bureau of Public Debt offers users of its TreasuryDirect Electronic Services:

> Warning: If you fail to follow the following directions, you can seemingly conduct a transaction—and even receive a confirmation number—when, in fact, no transaction was processed

Such a flawed system, commonplace on the web, would be unacceptable as a traditional transaction processing application. These deficient systems are not generally the fault of the developers; they arise from the constraints imposed by using technology—the web browser—inappropriate for the task. Even in cases where building an application on top of the browser does not force the system developer to adopt a poor design, the browser-based approach nonetheless encourages design flaws.

Electronic commerce is far from the only instance of this phenomenon. Web browsers are being used in place of specialized clients for a variety of tasks for which they are not well suited. The broader issue, therefore, is this:

> Browser-based applications (BBAs) are proliferating, yet most are fundamentally flawed. These flaws produce significant usability problems, which lead to the negative consequences one would expect: errors, wasted time, and dissatisfaction.

This paper analyzes the benefits and limitations of BBAs vis a vis specialized-client programs with an emphasis on usability issues. The paper begins with a brief overview of BBAs, noting the factors that make them popular as well as the underlying characteristics of browsers that make BBAs problematic. The paper then concentrates on studying the many practical problems that plague BBA users, tracing their origins to browser characteristics. Next, the paper considers more fully the factors that make BBAs popular despite their many weaknesses. After identifying a variety of questions that emerge from this paper that require further research, the paper concludes by considering briefly the most important and challenging of these issues, how next generation Internet Applications might be implemented.

## 1 Understanding BBAs, their strengths, and their weaknesses

For decades, Internet Applications were based on specialized client and server software. Then the World Wide Web arrived. The web began as a largely passive application, used for retrieving information but not for processing it. Browser and server capabilities were later extended, enabling browser users to send information to web servers (by completing forms) and servers to perform much more extensive information processing. This gave birth to a new breed of application, the BBA, also known as a web-based

application or weblication. Instead of being coded in a programming language and running directly on the operating system, the client side of these applications is coded in HTML (and javascript) and runs on top of the web browser. BBAs derive their functionality from a combination of the browser's (limited) functionality and the server's extensive information-processing capabilities, accessed through server-generated web pages. In short, when the client is browser-based, the servers do nearly all the work.

Today, BBAs dominate the world of Internet Applications. They derive their popularity, in part, from the immense popularity of the web itself. Using BBAs is a natural extension of using browsers, and just about anyone using the Internet can use a web browser. So people are comfortable with BBAs, and there is a widespread perception—true or not—that BBAs require little or no training. Nearly every computer connected to the Internet already has a web browser installed, so BBAs can be run without installing additional client software. Even users away from their own computers can be assured easy access to BBAs, since any machine with Internet connectivity and a browser will do. Moreover, BBAs are extremely useful, enabling people to do many things they never dreamed of doing on-line. Running in the browser window, and drawing on the hyperlinking of the web, BBAs create a seamless user experience, as the user moves naturally from finding a BBA to running it. Not surprisingly, BBAs also provide good browsing, which many applications—such as shopping on-line—require. And system developers enjoy the benefit of building platform-independent clients. For a compelling discussion of the upside of web-based applications for developers see Graham (2004).

Note that not all browser-launched applications are BBAs. For example, some web sites launch Java applets from the browser. Although these applets may run in a browser window, they constitute a distinct class of Internet Applications, since they derive their features from what Java—rather than the browser—has to offer.

Three technological aspects of web browsers are key to assessing their capabilities for supporting BBAs:[1]

- Browsers are page-oriented (in the sense of web pages, not printed pages). The basic unit of transmission from the server to the browser is precisely one web page, which the browser immediately renders on the screen. The page is coded in HTML, a presentation language, which tells the browser how the page should appear, but little else.
- Browsers are stateless, in the sense that each transaction between the client and server is an independent event. While browsers maintain a history of web pages visited, they view this history as a sequence of independent pages.[2]

[1] This assessment of pure browser characteristics does not reflect the features of such add-on technologies as Java applets and Flash applications, considered later in the paper.

[2] Cookies were introduced, in part, to overcome the independence problem. But while cookies are stored on the client machine by the browser (as agent for the server), it is the server, not the browser, that pays attention to their content. The browser remains stateless. Moreover, HTTP cookies are problematic in a number of ways (Fielding and Taylor 2002).

– Browsers are extremely limited in their capability to perform local computation; they only retrieve and display resources (such as web pages and images). When true information processing is required, they must rely on servers to do it for them.[3]

These three perfectly reasonable characteristics for a browser constitute significant limitations for a more general-purpose Internet client. While web browsers receive and display a whole page at a time, applications such as transaction processing require interactions at the level of records or individual fields. While browsers treat each page independently, most applications process information as a sequence of connected steps. And while very thin clients may not process information locally, most client–server architectures do draw upon the client for some local computation. These deficiencies lead to numerous practical problems for BBAs.

## 2 The practical problems of BBAs

While BBAs generally get the job done, they do not generally get it done well. Users of browser-based e-commerce, e-mail, e-learning, and other applications suffer through numerous difficulties (Landgrove 2003; Platt 2001; Tognazzini and Nielsen 2001). Consider the following representative problems:

– People frequently misuse the browser's "Back" button, confusing themselves and, often, the application, too.
– People bang buttons repeatedly when nothing happens soon enough. Doing so is problematic when the button authorizes such transactions as selling stock or buying an airplane ticket.
– BBAs often require people to submit entire forms and wait for replies before simple input errors are detected. Recovery from these errors is often poor.
– Screen updates that should take place automatically occur only if the user explicitly presses an "Update" button.
– Whenever the display changes—even if only a single field has changed—the web browser must receive a new page from the server, clear the screen, and render the new page. This delays human–machine interaction and interrupts visual continuity, degrading usability.
– Functionality is often weak and the interface clumsy.
– Printing what you want, in the format you want, is difficult or impossible.
– Saving what you want electronically, where you want it saved, is even more difficult.

These practical problems—and others like them—generally reflect the combined effect of the three key characteristics of browsers: page-orientation, statelessness, and limited local computation. Usability problems also follow from the limited user interface capabilities that browsers afford and the

[3] The introduction of javascript provided browsers with some local computation. But javascript was implemented as a kludge (Togazzini and Nielsen 2001) and the information-processing capabilities it affords are limited in significant ways.

awkwardness of constructing one application on top of another (the browser). The various practical problems merit more formal examination.

The following analysis of BBAs contemplates the vast majority of applications that do not take advantage of the added capabilities that Java applets, XML, Flash, and .NET may offer. These significant technologies are deferred to the end of the paper where solutions to the problems of current BBAs are considered.

### 2.1 Delays and discontinuities

Graphical user interfaces (GUIs)—sported by most web browsers—are designed to provide users immediate visual responses to their actions, but most users of BBAs do not experience the sub-second response times required for effective human–machine interaction. Because the browser is page-oriented, users are subjected to frequent temporal delays and visual discontinuities that interfere with their ability to interact smoothly with the application. Each time the browser receives information from the server, the user suffers through all of the following:

1. The new page arriving over the Internet.
2. The browser window being cleared.
3. The new page being rendered.

This phenomenon is not a problem when using a browser for browsing; each user action is intended to retrieve a new document, so some retrieval time is unavoidable and the screen-clearing is natural. But for most other applications, users do not require an entire new page with each interaction; they typically require only a few fields of information. Often the newly arrived page differs minimally from the previous one, yet the user not only waits for all the information to arrive again, but his or her visual continuity is also broken by the clearing and rendering. If the web page does not fit on a single display screen, the user has the further visual disorientation that accompanies scrolling down to locate the original place in the page. A good application would update only the changed display fields—immediately and without destroying visual continuity.

Consider, for example, the way the ''Add to Cart'' button is implemented in many shopping-cart sites. Clicking ''Add to Cart'' while scanning a catalog page triggers transmission of a new catalog page, differing from the original only in the two or three fields that summarize the cart's contents. Even if the new page arrives rapidly—which is often not the case—the consumer must still reorient by scrolling back down the page to his or her original position. Finding that position is even more difficult when the catalog page offers many similar products.[4]

[4] Clicking the ''Add to Cart'' button at Barnes&Noble.com can be even more disorienting for the buyer (and counterproductive for the e-seller). Netscape Navigator has on occasion responded by retrieving and rendering a page from Netscape.com beckoning the viewer to buy books at Amazon.com. It is difficult to imagine such a perverse phenomenon in a non-browser-based application.

The process for hiding the images shown in on-line catalogs is similarly flawed. Clicking the "Hide Images" button on a catalog page implies that you want to see less information than your computer is currently displaying, but ironically, your browser must request additional information from the server—a new web page without images. Instead of having the images disappear nearly instantly, you typically suffer a delay while the new page is transmitted, the screen is cleared, the imageless page is rendered, and you reorient yourself in the page.

Comparing a browser-based chat system with a specialized chat client such as an IRC client highlights the BBA discontinuities.[5] In both cases, each chat participant submits messages to the chat server and expects to receive reflections of the messages posted by others in (more or less) real-time. With an IRC-client, reflected messages are received one at a time and added to a scrolling window of messages. Users can read the messages as they arrive and scroll down to review older messages, if necessary. But with a browser-based system, reflected messages are received a page at a time. Each newly arrived page typically contains some new messages together with some that have already been seen. Once the screen has been cleared and the new page of messages rendered, the user must reorient visually to determine where in the list of messages to continue reading. If the user needs to go back in the conversation, it may sometimes be necessary to use the browser's "Back" button to return to an earlier page of messages that will partially overlap those in the current page. And, of course, while all this is going on, new messages are accumulating on the server that will be received by the user later as part of another new web page (leading to the same difficulties once again). The key problem here is that the basic unit in a chat is the message. A specialized chat client such as IRC operates at the message level and supports the chat smoothly. But the page-orientation of the browser-based system creates confusion and inconvenience by receiving and displaying a page of messages at a time, where which messages appear on a given page is a matter of chance.

BlackBoard, a BBA that supports course management and e-learning, illustrates how delays and discontinuities can follow from good intentions. BlackBoard is very good at acknowledging each action instructors take, such as posting an announcement or sending an e-mail. But because the browser is page-oriented, processing the acknowledgment involves the following steps: the screen is cleared, the sparse acknowledgment page is rendered (see Fig. 1), the user clicks "OK" (to acknowledge the acknowledgment), the screen is cleared, and a new page is rendered, returning the user to more or less where he or she was before taking the action. This is much overhead for a simple acknowledgment. And it is not just the acknowledgments that bog BlackBoard down. BlackBoard is a highly interactive program, but due to the browser's page orientation, each client–server interaction suffers from the same excessive overhead. For instance, before BlackBoard lists all students enrolled in a course, it warns the instructor that this function may be time

[5] This paragraph considers a pure browser-based approach to chat to illustrate the differences between BBAs and specialized-client applications (SCAs). In practice, many web-based chat systems use Java applets to avoid the problems of a pure browser-based approach.


**Fig. 1** One of BlackBoard's many acknowledgment pages

consuming and asks for confirmation. The warning is a good idea, but this page-level interaction with the server often takes more time than it does to generate the full student list. A client-generated warning would be more efficient for the user.

Delays and discontinuities in the user experience follow not only from the browser's page orientation but also from its lack of local computation. The client–server architecture is based on the notion that processing is distributed between the client and server in a way that makes most efficient use of the computing resources. In the typical application, some tasks are best performed locally by the client. For example, if a customer changes the number of units he or she orders, one would expect the client software to perform the minimal calculations necessary to arrive at a new subtotal, recalculate any sales tax, and produce a grand total. But BBAs do not do this; they rely on the servers to do the processing. So instead of a near instantaneous recalculation, the user must suffer through the latency associated with sending the new information over the Internet to the remote server and receiving the updated information back. Moreover, while this computational burden on the server may seem small, the composite effect of all users' computations being shifted to servers may overburden them, introducing further delays.

Consider, for example, a consumer using a shopping-cart website who decides to order an additional copy of a book. He or she typically must go through the following arduous sequence:

- Update the quantity field.
- Press the ''Update Quantity'' button.
- Wait for a new page (that differs only slightly from the old one) to arrive.
- Reorient himself or herself after the screen is cleared and rendered again with a nearly identical display. Scrolling may be necessary to find the previous location on the page.

In contrast, a specialized application could update the screen in-place immediately upon the buyer's increasing the quantity. In general, it would not be necessary to interact with the server at this point, because the

computation could take place locally. And even if interaction were required, only the necessary fields would be received and updated in-place. In short, the delays and discontinuities degrade the BBA's usability relative to a specialized client.

Notice how the lack of local computation interacts with the browser's page-orientation to exacerbate the delays and discontinuities. If servers are performing computations that might be better suited for the client, then at least we would like the server and client to exchange information efficiently. But this would imply exchanging data at the record or field level, not the page level. Conversely, if each interaction incurs the pain of retrieving and rendering a full web page, then we would like to do it infrequently, by having the client do more computation. But the web browser is not up to the task. Consequently, the combination of limited local computation and no small retrieval units is double trouble for the BBA.

## 2.2 Confusion and errors

### *2.2.1 Button banging*

Many people, competent at browsing with browsers, are prone to confusion and errors when using BBAs. Some of the confusion and errors may be second-order effects of the delays and discontinuities. Users often press a button repeatedly when they do not get an immediate response. In the case of e-consumers pressing the "Submit" button, this behavior leads to multiple transactions. Stories abound of electronic shoppers getting stuck with two airline tickets for the same itinerary and electronic traders selling the same lot of stock twice (ending up short).

How significant is this problem? Evidently a few seconds of non-response are enough for users to pound the "Submit" button, necessitating warnings such as the following:

> After you click the submit button, wait for your confirmation to appear. Do not click the "Submit" button more than one time.

Warnings of this type are very common on the web, especially among financial institutions. Rather than warn in advance and hope for the best, a good application would immediately acknowledge that the transaction is in progress and then lock out the extra "submits." But the stateless web browser, having no sense that a transaction sequence is in progress, simply notifies the server of the user's latest request. In this case, if the server does not prevent the duplicate transaction, the e-consumer is out of luck.

The bugaboos of button banging are not limited to e-commerce applications. When users of Prentice-Hall's Train & Assess IT e-learning system click repeatedly on a testing button, each click invokes the test anew. This earns the student a zero on the first instance of the test and may prevent him or her from retaking the test. This problem is quite common since students often become impatient waiting first for the Macromedia Authorware plug-in to load and then for the test to arrive over the Internet.

### *2.2.2 The "Back" button*

Another significant source of confusion and errors is the browser's "Back" button. Interacting with an application—especially a transaction processing application such as on-line shopping—typically requires a sequence of steps. Many BBAs divide the transaction sequence into a series of web pages. The user might first select products, then quantities, then shipping options, then payment options, and so forth. Since the browser is stateless, it sees these simply as a series of independent pages. The server must manage the state of the transaction. Now what is a buyer to do if he or she needs to back up a step in the process? The most natural action would be to click the browser's "Back" button, especially since this is the most frequently used browser feature (Nielsen 2000a). But doing so only tells the browser—blissfully ignorant of the state of the transaction—to redisplay the previous page. The server may or may not be notified. And depending on how the buyer moves forward again (pressing the "Forward" button or clicking a button within the page), and how well the server keeps track of what the buyer is doing, both the buyer and application can become confused about where the buyer is in the sequence.[6] Did the buyer just modify a previous transaction or submit a new one? Indeed, this is another way of getting two airplane tickets or selling the same stock twice.

The "Back" button presents other usability problems, as well. Under some circumstances, pressing the "Back" button in Netscape Navigator 7 produces the following message, unintelligible to many users:


Given this cryptic message, some users are likely to be unable to reach their desired page while others are likely to resubmit transactions that have already been processed and should not be resent. Might this just be a weakness of Netscape Navigator? Microsoft Internet Explorer (6.0) seems to generate a more understandable message when the "Back" button is pressed under the same circumstances:

> Warning: Page has Expired
> The page you requested was created using information you submitted in a form. This page is no longer available. As a security precaution, Internet Explorer does not automatically resubmit your information for you.

[6] Ironically, cookies—which were intended to help compensate for the browser's statelessness—can actually contribute to this confusion surrounding the state of the application (Fielding and Taylor 2002).

> To resubmit your information and view this Web page, click the Refresh button.

But the ''warning'' fails to warn users of the potentially dangerous consequence of their actions—the risk of duplicate transactions. Moreover, pressing ''Refresh'' yields the following dialog box, even more confusing than the one in Netscape:


So the hardship for users is not an artifact of a particular browser but of using the browser—any browser—as a platform for the application.

How serious are the various problems with the ''Back'' button? The Pennsylvania Department of Revenue offers this warning to taxpayers filing their returns on-line:

> Please DO NOT use the BACK or FORWARD buttons on your browser's toolbar. If you use either button, your information WILL NOT BE SAVED and you will have to fill out the appropriate form(s) again. Please use one of the navigation buttons within the pa.direct.file application to navigate.

Even worse than having to complete a complex form again would be believing you had conducted a transaction successfully, because you received a confirmation number, when, in fact, no transaction was processed. And the Bureau of the Public Debt warns its TreasuryDirect customers that this can happen to them if they press the ''Back'' button while in the process of purchasing U.S. government notes and bills. In contrast, customers who pay their AT&T Universal Mastercard bills on-line and press the ''Back'' button may have the opposite experience. They may successfully complete the transaction but nonetheless believe that they failed, necessitating this warning to prevent them from trying to pay the bill again:

> Note: If you hit the back button, you will go back to the payment screen with a zero dollar amount. Be assured that your payment was accepted, and the zero amount in the payment window does not mean it was cancelled.

So pressing the ''Back'' button in BBAs can be quite dangerous. It would be bad enough if BBA users simply had to unlearn their favorite browser feature—the ''Back'' button. But some BBAs demand the opposite behavior—that users do employ the ''Back'' button. For years, American Express offered this guidance to its customers:

> To change the amount you wish to pay, please use your browser's BACK button.

Perhaps the least usable of all are those sites that are inconsistent. Opening an account with Discover Bank brings this ominous instruction:

> Please do not use your browser's "Back" button so that you do not lose the information you have entered.

But elsewhere on the same site users were (until recently) told the opposite:

> If you are registered for the Account Center, please use your browser's "Back" button to return to the Login page and try again.

Needing to keep track of when you *may* use the "Back" button, when you *must* use it, and when you *must not* places a cognitive burden on the user and, not surprisingly, leads to many errors.

### *2.2.3 The "Stop" button*

The "Stop" button can also be problematic, necessitating the following warning when transferring funds at Wachovia:

> Please know that clicking the browser's "Stop" button may or may not stop your transaction.

Since browsers were designed for browsing, "Stop" tells a browser to stop downloading or rendering the file, not to cancel a transaction in progress. But a user might reasonably expect "Stop" to mean "Stop."

While in some cases pressing "Stop" may have no effect on the transaction in process, in other cases it may leave the user stuck in the middle, with a partially processed transaction. Prentice Hall warned students trying to register for its Active Book site about this as follows:

> Please note that it may take several seconds to validate your ACCESS CODE. Please do not click your browser's "Stop" button while the validation process is taking place.

Failing to heed this warning may leave the student unregistered for the site but with a "used" access code that is no longer valid for registration. Here, too, instead of warning users, a normal application would lock out the illegal actions.

### *2.2.4 Updating the server*

When contemplating delays and discontinuities we already encountered another major source of confusion and errors: the need for servers to generate new shopping-cart pages when consumers change quantities. Most shopping-cart sites use an "Update Quantity" button to force the transmission of a new web page when consumers change order quantities. But this button is problematic because until the user clicks the button, he or she is looking at an inconsistent screen whose totals may not match the quantity apparently ordered. If printed, this inconsistency is preserved for posterity,

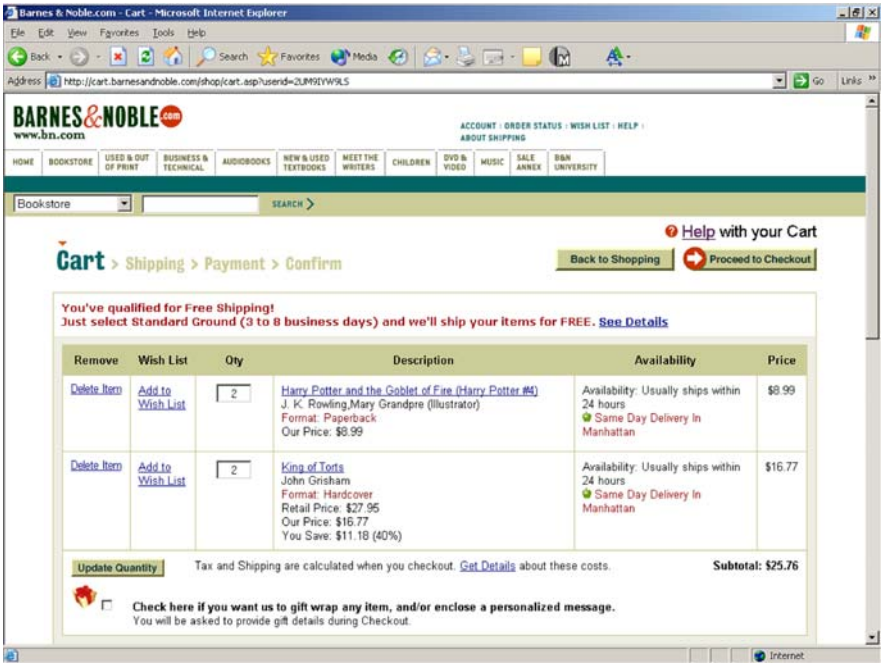



**Fig. 2** Inconsistency: line-item totals and subtotal do not match the quantities ordered

as in Fig. 2. And at many sites, if the user never clicks the ''Update'' button the updates are lost. Some BBAs are coded to invoke the server automatically when a user changes a field, but this solution can be even more disruptive, especially if the user is in the midst of a series of changes.

Inconsistencies within a site concerning the consequences of not pressing the ''Update Quantity'' button make for even more confusion and errors. At both Amazon.com and Barnes&Noble.com you are forgiven for not clicking the ''Update'' button if you proceed directly to ''Checkout.'' But if you ''Continue Shopping'' the updates are lost. And since these actions take the consumer to a page that does not display the shopping cart, the customer is not likely to know whether the changes took effect or not. Compounding these internal consistency problems are the differences across sites. For example, BBAs vary widely in how they behave when users press the ''Enter'' key after entering a changed quantity (Silver and Ward 2004).

### *2.2.5 Bookmarking*

A fifth source of errors and confusion is fairly unique to BBAs because it pertains to bookmarks. Users have become accustomed to setting bookmarks when browsing the web, but when these same users bookmark pages in the middle of a BBA's transaction sequence, the consequences can be

severe. A user jumping into the middle of an application is not a problem often encountered by traditional applications.

### *2.2.6 Multiple browser windows*

Consider yet another problem: the use of multiple browser windows (Wroblewski and Rantanen 2001). Users of BBAs that open secondary browser windows may find the primary window out-of-date. Prentice Hall's Train&Assess IT uses its primary window for navigation and status information, but the actual training and testing are performed in a secondary window. When the secondary window closes on completion of a lesson or test, the information in the primary window often does not reflect the activity just completed, since the browser is stateless and sees the browser windows as independent. Refreshing the primary window is required to bring it up-to-date. This confuses many students, who think the system did not credit them with the training or testing.

BBAs that open new windows may encounter another problem. To avoid unwanted advertisements, many users set their browsers to block "pop-up windows." But this is problematic for BBAs that use such windows for required functionality. For example, at least one major airline uses a pop-up window to provide travelers with their on-line boarding passes. If pop-ups are blocked, the boarding passes never appear. Users cannot rectify this problem by unblocking pop-ups and trying again because by then the server has recorded the boarding pass as issued and will not generate another one. This difficulty is an artifact of using the browser as a platform for e-commerce. It is difficult to imagine a traditional application behaving this way.

The secondary window problem is not limited to BBAs that automatically open new windows for their users. Most BBAs allow users to open pages in new browser windows by right-clicking the link. When users engage in transactions in such a window, status information in the original window becomes out-of-synch with the true status. Sometimes such actions even crash the session by confusing the server-side of the application, which is the only part of the application keeping track of what is going on. One might argue that opening a new window is bad behavior by the user, but of course the BBA is truly at fault for allowing this behavior. Ultimately, the problem traces back to the use of browser technology where it is not appropriate.

## 2.3 Clumsy interfacing and limited functionality

### *2.3.1 Limited interfacing techniques and capabilities*

Since browsers hardly compute, the server must provide the bulk of a BBA's functionality. And since relying on the server leads to delays and visual discontinuities, the user interface to a BBA's functional capabilities is often clumsy. Moreover, since interactions between the browser and server are at the page-level, the BBA—despite its GUI—does not provide a direct

manipulation interface. Interactions can also be awkward because the BBA's interface is constrained by what the browser can be made to do with HTML and javascript. In short, as Graham (2004) puts it, web pages are "lame" as a user interface.[7]

While most applications implemented in a modern GUI employ toolbar icons, pull-down menus, short-cut menus, and key-combinations to invoke functionality, BBAs implement most of their controls as hotspots (links) in the web page, often identified by graphical buttons. Since most users will be familiar with their GUI's traditional widgets, these non-standard buttons degrade usability. They may be scattered throughout the page, rather than clustered at the top of the screen, so the user must hunt for them. They are often difficult to pick out. Even an experienced user who knows where to find the button may still suffer a usability loss by having to scroll the display or move the mouse a substantial distance to reach it. And there are no key-combinations for the power user to invoke as an alternative to pointing and clicking. Application-specific shortcut menus are also absent; right-clicking activates the browser's shortcut menu, which offers options at best irrelevant to, and in some cases highly inappropriate for, the BBA.

BlackBoard, for example, always places its "Cancel" and "Submit" buttons at the bottom right of the web page. In many ways the design reflects great attention to usability: the consistency across the site is laudable; the "Cancel" button precedes the "Submit" button, decreasing the likelihood of errors; and the buttons are on the right-side, a very short distance from the scroll bar. But placing the buttons at the bottom of the web page, which is often very long with the most important elements near the top, forces users to scroll down substantially each time they need to complete an action. Constantly having to scroll down impairs usability.[8]

Most GUIs feature such direct-manipulation facilities as "drag and drop" for moving items. But BBAs do not. If an instructor wants to rearrange items in BlackBoard, instead of dragging and dropping the items, he or she must renumber them by putting new numbers in the associated text boxes. And as each item is moved, a new page is sent from the server. Renaming items is even more tedious than rearranging them.

### *2.3.2 Piggybacking applications*

Another difficulty with BBA interfaces is not so much a function of the browser's technical characteristics as it is a consequence of building one application on top of another. BBAs confront their users with two sets of

[7] Graham goes on to claim that web pages are "just good enough," however, given the other benefits of web-based applications.

[8] One might argue that BlackBoard could have worked around this problem the way many other BBAs have by putting the buttons at the top and bottom of the page. But the very need to employ a work-around highlights this inherent BBA interfacing weakness. Moreover, BlackBoard likely had good reason for not placing buttons at the top of the page. BlackBoard takes a restrictive approach that forces the user to confront the full form before submitting it. A non-browser-based approach, however, could have implemented this restrictiveness without requiring so much scrolling.

interface elements: (1) the menus, toolbars, and other elements of the browser's interface and (2) the controls and inputs scattered around the web pages that define the BBA. The browser's interface, of course, is common across all BBAs and is usually consistent with other applications running on the same GUI. In contrast, the BBA's own interface elements are unique.

Presenting users with two sets of interface elements increases the complexity of the design, decreases the intuitiveness of the interaction, and degrades usability. Nielsen (2000a) notes that controls need to do what the user expects them to do. The double interface makes this more difficult to accomplish. For instance, as previously observed, right-clicking on a BBA widget yields not the BBA's short-cut menu, but the browser's, whose options are either irrelevant or inappropriate for the BBA. Sometimes the user must choose between similar, or even seemingly identical, options in the two interfaces. In particular, this dilemma adds to the confusion surrounding the "Back" button. Do I hit the "Back" button on the browser's toolbar to cancel an operation or the "Cancel" button that was placed on the web page for that purpose (Nielsen 2000b)? Do I use the browser's "Back" button or the "Back" button embedded in the web page? Similarly, when I move forward again, do I use the browser's "Forward" button or some button on the page? For some BBAs, it doesn't matter. For others, an incorrect choice can be disastrous. Users of the Amtrak site, for instance, can stumble upon this error message:

> Problem With Navigating Through the Site. The problem you are experiencing may be related to using your browser's Back' and Forward' buttons in the booking process. In general, you will get better results by using the buttons on each page to progress through the booking process. The buttons are usually orange in color, and labeled Next', Submit', etc.

### *2.3.3 Limited functionality*

Restricting functionality can reduce the clumsiness of the interface, and this is what many BBA designers do. Between the things a BBA simply cannot do, and the things that BBA designers choose not to do, many BBA's offer limited functionality. For example, early browser-based e-mail systems often lacked spell checking, ways of sorting messages, ways of filing messages, and ways of attaching files. And even those BBAs that now provide these functions often suffer from constraints on the functionality and poor interfacing. For example, specialized e-mail clients typically provide continuous spell checking whereas BBAs are usually limited to spell checking the whole document on request. Perhaps these functional limitations are one reason Nielsen (2000a) recommends not "trying to implement advanced applications inside a Web browser."

### *2.3.4 Browser incompatibilities*

It is well known that many BBAs work better in one browser than in another. Many sites even inform their users of this fact. But while in some cases

the differences simply reflect the quality of the user experience, or the extent of the functionality supported, in other cases a BBA may work in one browser and fail without explanation in another. To some extent, of course, this is a fault of the designers, who should have tested the BBA for compatibility with all the major browsers. But again, the root cause of this problem—and the reason that this issue is painful for developers even when handled properly—is that applications are being piggy-backed on top of browsers.

Incompatibilities are not limited to different browsers. Different versions of the same browser can be problematic. BBA developers often choose between implementing systems with features that will not function on older versions of the browser and limiting functionality to maintain compatibility with the older versions. Even users of the latest version of a browser may encounter problems if they have not installed all the updates. Consider, for example, these messages from AT&T Universal Mastercard and Discovercard, respectively:

> If you currently use Internet Explorer 6.0 and experience problems requesting a Click-to-Pay Payment, you may need to download a software update from Microsoft®.

> Internet Explorer users: If you are getting this error, you may need the latest browser updates.

### *2.3.5 Cookies*

Many BBAs depend on cookies. When users block cookies, some BBAs notify them that cookies must be accepted, while others either malfunction or cease to function without explanation. Yes, BBAs that malfunction or fail without explanation are poorly constructed, but we should not hold the browser-based approach blameless. Since cookies are a work-around for the statelessness of browsers, this problem, too, is ultimately a consequence of adopting the BBA approach.

### *2.3.6 General clumsiness*

Figure 3 could well be the most glaring example of BBA clumsiness. When using Internet Explorer—a Microsoft product—to download and install MSN Messenger—another Microsoft product—users were are at one time warned that they were about to have a choice but they had to choose the first option. What normal application would behave this way?

## 2.4 Printing

Printing is another good example of how the lack of local computation combines with the page-orientation to make BBAs problematic. Unable to send information to the printer directly, BBAs generally print by having

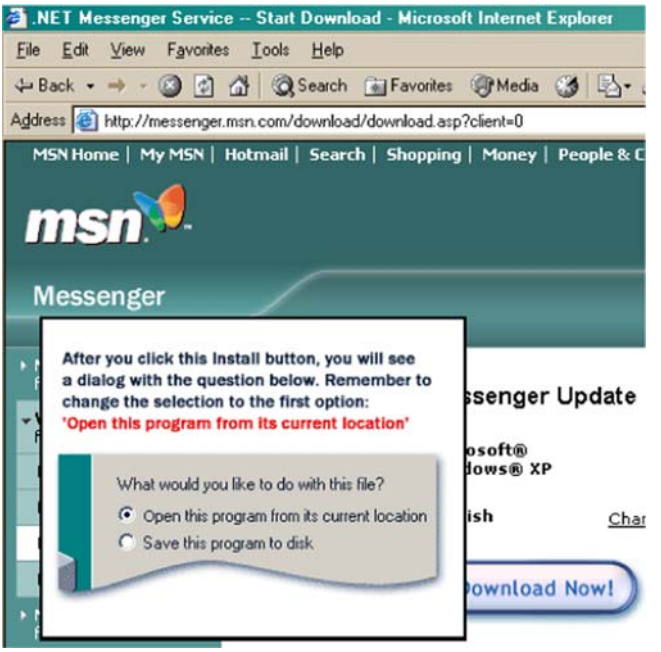



**Fig. 3** A Microsoft kludge

users invoke their browsers' print functions. But browsers print entire web pages, which is usually not what the applications should be printing:

- Web pages combine information presentation with navigation (Nielsen 2000a), so the typical web page contains the information to be printed as well as navigational buttons and navigational guidance. When such a web page is printed the printout is cluttered with the extraneous navigational elements.
- While specialized clients sometimes find it useful to print a screen image—that is, exactly what is on the screen—applications usually need to print more or less than what currently appears. And they may need to print it in a different layout from what appears, since screen-oriented and print-oriented documents are constructed differently (Nielsen 2000a). So the browser's print function, which prints the entire web page being viewed, does not usually print what is needed. Being constrained to print the page, the whole page, and nothing but the page, is a significant limitation for BBAs.

The typical work-around for this problem is to put a "Printable Version" link on the page that, when selected, generates a new web page showing exactly what should be printed. But this, too, is problematic because it degrades usability. Consider the sequence the user must follow to print something:

1. After clicking on the "Printable Version" button, the user must wait while the new page is retrieved over the Internet and rendered on the screen.
2. Then the user must instruct the browser to print the current web page.

3. Finally, after printing, the user must leave the special print-oriented page and return to where he or she was previously.

Usability suffers in several ways:

- A sequence of at least two (usually three) commands is performed by the user to take one simple action (to print something).
- Some BBAs commingle the two elements of the interface, first having the user select a link on the page to generate the "printable version" (step 1) and then having him or her employ the browser's controls (a toolbar icon, menu item, or key combination) to invoke the print function (step 2). Those BBAs that avoid this problem by placing a "Print" link on the printable version page violate the purity of the printable version, since the link appears in the printout.
- The user suffers two needless delays: (1) He or she must wait for retrieval of the special printable page over the Internet even though the page typically contains information already on the client. (2) He or she must also wait for the page to be rendered on the screen even though the page is being retrieved for printing and not viewing.
- The user must also find a way to navigate back from the page displaying the printable version to where he or she was prior to printing. BBAs take one of two approaches. Many BBAs open the printable version in a secondary browser window, so after printing the user closes the extra window. Other BBAs render the printable version in the current browser window, generally requiring the user to use the "Back" button to return to the previous page. The secondary window approach is typically the more usable, but each requires an added step by the user and each has its problems.

  When the printable version appears in a secondary window, users must close this window after printing. If the BBA puts a "Close" button on the page—as many do—the purity of the printed version is violated. On the other hand, if the page does not contain a "Close" button, the user may be unsure how to exit the printable version. Users may be reluctant to click the browser's "Close" button because if the BBA did not open a new window, the browser will close, terminating the application in progress. Usability may suffer as the user tries to determine whether or not the BBA opened a new window.

  Rendering the printable version in the current browser window can be even more problematic than opening a secondary window. Since a pure printable version contains no navigational buttons, it is a dead-end in hyperspace. This approach usually requires the user to invoke the browser's "Back" button. But he or she might first search exhaustively for a more intuitive possibility. As already noted, the "Back" button is hazardous in many BBAs, so users take a risk by clicking it. Moreover, in those circumstances where users are going back to a page generated by a form posting, pressing "Back" generates the cryptic messages about "expired" pages discussed earlier. Even for experienced users who may not be bothered by those messages, reposting the form data adds more keystrokes and delays.

Some BBAs solve the dead-end problem by putting an unobtrusive link somewhere in the printer-friendly page to take the user back after printing. But this is a no-win situation. If the link does not blend in well, then it violates the purity of the navigation-free printable version. And if it does blend in well, then the user cannot find it to use it.
Perhaps the greatest danger of the same-window approach is that if users mistakenly think a new window was opened—as in many BBAs—they will close the window, exiting the browser and the BBA unintentionally.

Whichever approach is used, the user loses visual continuity by leaving and returning to the focal page (which sometimes has to be rendered again by the browser).

Two more difficulties:

- Browser print functions themselves are relatively unsophisticated and afford page authors little control over the structure of the printout. So, when all is said and done, the pages that emerge from the print function often suffer from poor pagination and other problems.
- Websites that use frames introduce an additional problem, because users are easily confused about which frame they are—or should be—printing.

Some websites use the Adobe Acrobat Reader to handle documents that may require printing. While Acrobat offers a number of benefits, it does not solve any of the problems just discussed other than pagination. And Acrobat introduces costs of its own: The user must wait for Acrobat to load and users are often confused by the double toolbars—one for the browser and one for the Acrobat plug-in. In particular, using the browser's print icon rather than Acrobat's—an easy error to make—will often fail to print the document and leave users baffled by the cryptic error message they receive.

Nielsen (2000a) notes that HTML offers a tag for identifying an alternative printable version of a page. In browsers that support this tag, this approach would eliminate the cumbersome three-step printing process as well as the need to return to the initial focal page. But the approach does not solve all of the BBA printing problems. It does not avoid the need to transmit essentially the same information twice. And it does not support the more customized, user-driven printing that many applications should provide.

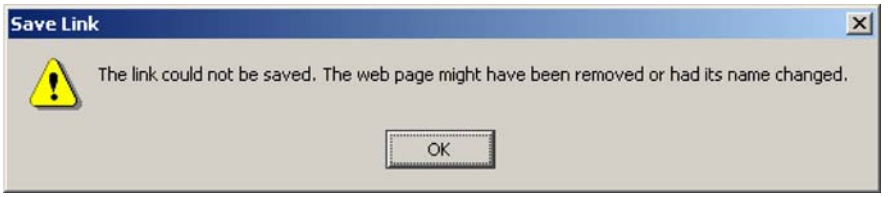
## 2.5 Personal electronic filing

Since printing is problematic for BBAs, it would seem to be good news that e-mail, e-commerce, e-learning, and all the other "e"s are moving us steadily—if slowly—in the direction of electronic filing rather than paper printing. But BBAs have trouble filing information for much the same reason they have trouble printing it: Browsers save whole web pages and what needs to be filed is often not a page. A consumer may only need to file a few key numbers from each month's on-line statement, but instead of just adding a new record to an existing database he or she must save an entire web page as a separate HTML file. The consumer could cut and paste fields from the web

page into a database management system or some other application—and many people do—but this work-around is tedious and error-prone.

Not only do BBAs generally constrain their users to storing information as web pages, but their support for filing these pages locally is also weak:

- Consumers trying to "go electronic" will likely want to organize the many pages they are saving. But with a BBA, all they can do is rely on the operating system's directory structures and explicitly choose a subdirectory each time they save a page. Users would likely want to group documents from a given BBA together and might even want to have subgroupings of documents—for example, one subdirectory for monthly statements and another for confirmations. A proactive application that exploited the directory structure could make selecting a destination directory much easier or even automatic. But BBAs cannot do that.
- Many BBAs stop short of what they could do—namely, assigning the pages distinct and meaningful filenames—so users must devise their own naming conventions as well. Relying on the default filename generally is ineffective because many BBA-generated pages have the same default filename. For example, statements from financial institutions typically default to such filenames as "Statements.htm" rather than to more descriptive names including the institution name and statement date.
- Some websites are designed in such a way that the page cannot be saved at all. One receives this message from Netscape Navigator when trying to save a purchase confirmation from cvs.com:




Some BBAs offer improved local filing by generating downloadable data files that can be imported into other applications. One longstanding example of this approach is the ability to download files in formats suitable for Quicken or Microsoft Money. Other examples are becoming more prevalent. Many financial institutions will deliver your monthly or quarterly statement in PDF format. Similarly, BlackBoard supports downloading the student grade book as a CSV file that can be imported into spreadsheet programs and other applications. While this approach improves on saving web pages, it still falls short of providing the local storage and retrieval functions that a typical non-BBA application could provide. Moreover, since the file download capabilities still rely on the browser, they suffer from the same weaknesses in naming and organizing files as do the facilities for saving web pages. And using the BBA's downloading features will often be non-trivial. Here, for example, are the instructions BlackBoard used to provide:

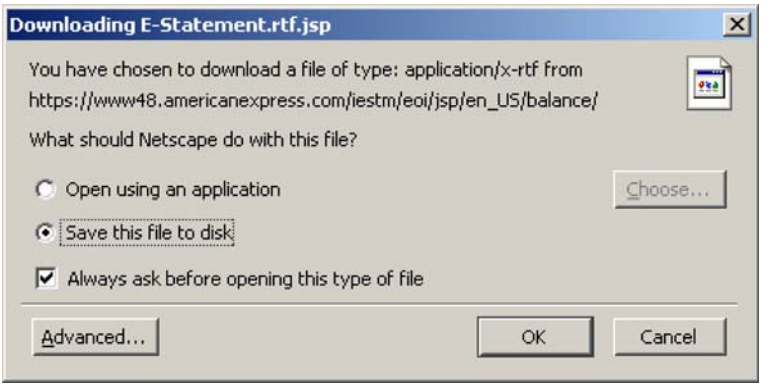
Instructions

The gradebook has been saved to a file. To download this file to your computer, follow the instructions below.

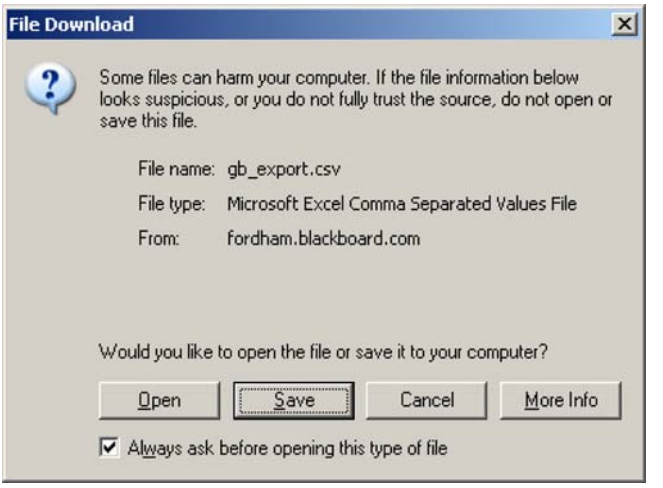
1. After saving the file, open Microsoft Excel or a similar program, and go to the File menu. Select Open. On a Macintosh, this may be the only way to open the file, as the system may not automatically associate the saved file with Excel.
2. Locate the directory where the file is saved and double-click the filename (the file is named gb_export.csv by default.)

Download Tip: When downloading the gradebook, you may want to save it to a more permanent location, rather than the default location. This will help you locate the gradebook more easily once download is complete.

Other BBAs—especially those that support downloading in multiple formats—provide even more complex instructions, while still others provide no instructions at all, leaving the user to guess. In any case, somewhere along the way the user will confront a dialog box such as this one in Netscape Navigator




or this one in Microsoft Internet Explorer



Once again, much of the complexity, added steps, potential confusion, and room for errors in this "improved" approach results from piggy-backing the application on top of the browser, relying on the browser for functionality that a specialized application would provide itself.

All told, BBAs are inherently deficient in their ability to support personal electronic filing. Perhaps the difficulties with electronic filing are why so many e-commerce BBAs recommend that users print their confirmation pages.[9]

## 2.6 Summation

The foregoing discussion, summarized in Table 1, suggests that BBAs pose significant practical problems. These problems can all be traced to the browser-based approach (see Table 2) and most reflect some combination of the three key browser characteristics. One might ask, however, to what extent these common weaknesses are unavoidable consequences of the browser-based approach and to what extent they simply reflect poorly designed BBAs. Not all of these problems are unavoidable. Some can be eliminated by extensive server-side programming or clever use of javascript. Others are inescapable, although the better designed BBAs sometimes reduce their magnitude somewhat. In any case, the frequency and severity of the browser-based problems in practice suggest that many web developers are not successfully avoiding them or reducing their severity. Moreover, many of the solutions are kludges that bring with them their own problems. So while there is certainly much room for designing better BBAs, better BBA design does not resolve the issue.

That said, over the last 2 years we have slowly been seeing some BBAs finding ways to reduce somewhat some of the negative consequences of browser-based limitations. Consider the following:

- JPMorganChase improved its printable version capability in a number of ways, most significantly by using javascript to invoke the browser's print function automatically once the printable-version window is loaded. This approach simplifies matters for the user, reducing the steps from three to two (for a process that ought to require just one step, of course). Most BBAs have not made this modification.[10]
- Many web sites improved their treatment of printable versions by opening a new browser window for the printable version, thereby eliminating some of the difficulties described earlier. Many others, however, retain the single-window method. But those retaining the single window are often

[9] In fairness to BBAs, there is another obstacle to more widespread electronic filing: Most users are not yet sufficiently disciplined in backing up their files, and home users and many office users are not on networks that backup automatically, so printing is a safer alternative to electronic filing if preserving the information is important.

[10] Express Scripts reduced the process to a single step by automatically invoking the browser's print function on the printable version page and then immediately loading the next page required by the user. If the downloading and rendering take place fast enough, the user may hardly notice the intermediate printable version page. But the kludginess of this clever approach—which has the potential to confuse the user—is itself testimony to the deficiency of the browser as a platform.

**Table 1** Summary of practical problems

| Practical problems | Explanation and specific manifestations |
|---|---|
| Delays and discontinuities | All user actions—such as clicking "Add to My Cart" or "Update Quantity"—require transmission and rendering of a complete new web page introducing temporal delays and interrupting the flow of the interaction. The intervening clearing of the display further interrupts the flow, introducing visual discontinuities.<br>This problem is exacerbated when users must scroll down to reorient within the newly arrived page. |
| Confusion and errors | Button banging: Pressing the same button repeatedly can trigger duplicate transactions.<br>Pressing the "Back" button often causes undesirable effects in a transaction sequence.<br>Pressing the "Stop" button does not cancel a transaction in progress though users may expect it to do so.<br>Changing the quantity ordered in a shopping-cart site can be troublesome because the user must click an "Update Quantity" Button.<br>•Users may fail to register the update (and not know it).<br>•Temporary inaccuracies may appear on screen. These may be preserved in print.<br>Bookmarks may enable users to jump into the middle of a transaction sequence.<br>Activity in secondary browser windows may leave the primary window out-of-date or confuse the application. |
| Clumsy interfacing and limited functionality | The BBA does not provide a direct manipulation interface (although it is a GUI).<br>Functionality is often restricted due to browser limitations or to reduce interface awkwardness.<br>Users must hunt for buttons on web pages instead of using toolbar icons, pull-down menus, key-combinations, and shortcut menus.<br>Users confront two sets of interface elements due to the piggy-backing of one application on top of another (the browser).<br>Incompatibilities across major web browsers cause functionality and BBAs to fail, often without explanation.<br>Problems with cookie handling cause features and BBAs to fail, often without explanation. |
| Printing | Users can print only the web page they are viewing:<br>•Web pages are typically cluttered with navigational information that should not be printed.<br>•Even without navigational clutter, applications often need to print something other than exactly what appears on the screen.<br>•Even if the web page contains exactly what needs to be printed, screen- and print-oriented displays require different formatting.<br>•Pagination is poor.<br>The various workarounds (e.g., using "printable versions" or Adobe Acrobat) provide incomplete solutions or introduce problems of their own. |
| Personal electronic filing | Users need to store and retrieve information—records or fields—locally, but BBAs generally let users store only entire web pages. Some BBAs will not even do that.<br>Web Browsers provide limited support for naming and organizing web pages (and any other files) stored locally, so files are difficult to find when they are needed. |

**Table 2** Practical problems following from fundamental browser characteristics

| Browser characteristic | Practical problems |
|---|---|
| Page orientation | Delays and discontinuities:<br>• Lack of more granular information exchanges necessitates transmitting entire web pages between the server and client<br>Printing problems:<br>• BBAs print only whole web pages, not the more customized printouts typically needed by applications and users<br>Filing problems:<br>• BBAs can file only whole web pages, not specific fields or records |
| Limited local computation | Delays and discontinuities:<br>• Limited local computation leads to frequent round-trips to the server<br>Confusion and errors:<br>• Updated quantities may be lost<br>• Totals on the screen may not reflect updated-quantities |
| Statelessness | Confusion and errors:<br>• Problems with button banging, the "Back" button, and the "Stop" button, which can crash the application, create duplicate transactions, and cause other problems<br>• Problems keeping track of which items and how many of each are in the shopping cart<br>Clumsy interfacing and limited functionality:<br>• Problems with cookies, a work-around for statelessness, causing applications to fail (often without any explanation) |
| Piggy-backing of applications (BBA on top of browser) | Clumsy interfacing and limited functionality:<br>• BBAs do not provide direct manipulation interfaces—only the browser's interface<br>• Users must hunt for buttons on web pages instead of using toolbar icons, pull-down menus, key-combinations, and shortcut menus<br>• Users confront two sets of interface elements |
| Other browser characteristics | Confusion and errors:<br>• Users may try to jump to a bookmark in the middle of a BBA<br>• Secondary browser windows cause various problems |

coding their websites in a way that avoids the "postdata expired from cache" hassle.
– AT&T Universal Mastercard users can now save their confirmation web pages locally following on-line bill payment.
– Norwegian Cruise Line now supports users' opening link targets in new browser windows. Previously users trying this would open a useless window containing an inapplicable error message.

No doubt that by the time you read this some of the specific examples identified in this paper will also have been remedied by the site designers. But while some BBAs have worked around various browser-based problems with more extensive server-side coding, greater use of javascript, or modified designs, most have not. How are we to understand this phenomenon—that is, the relatively slow pace of improvement? A number of potential explanations come to mind:

- Some BBA developers may be unaware of their BBA's weaknesses. This would not be surprising since many websites pay insufficient attention to usability and usability-testing (Shneiderman et al. 2003; Nielsen 2000a), since the problems documented here have not previously been described so extensively, and since many BBA developers are non-professionals.
- Some BBA developers may recognize the problems but be unaware of the solutions other developers have pioneered. Put differently, some of these work-arounds may be diffusing slowly through the developer population. This may be especially likely since some BBAs are professionally developed but others are not.
- Some BBA developers may be aware of these problems but discount their significance, concluding that most of their sites' visitors either do not perceive the problems or do not suffer unduly because of them.
- Some BBAs may not be well maintained, remaining relatively unchanged over time despite their weaknesses.
- BBA developers, especially for complex applications, generally rely on software development tools rather than writing extensive HTML and javascript themselves. These tools may not support the alternative approaches, leaving the developer either unable to implement the work-arounds at all or unable to implement them without substantial effort.
- While some of these work-arounds may be easy to implement, others may require significant effort, especially when server-side coding is involved. For some BBA teams, fixing the problems identified here may be a low priority, not deemed worth the effort. This may be especially true since the workarounds do not solve the major BBA weaknesses and may provide only marginal improvements.
- Some BBA developers may purposefully choose to forego the solutions. Since some kludges that reduce a given problem either introduce another or force a shift in design, a given BBA's designers may prefer the original approach, despite its limitations, to the alternatives.

Whatever the reasons may be, BBA problem severity is being reduced relatively slowly. More importantly, even were the pace to quicken, these efforts would provide only marginal improvements and could not solve the major BBA problems. Before turning to more fundamental solutions, an assessment of BBA strengths—found in the following section—is in order.

## 3 Understanding BBA popularity

Despite their many problems, BBAs dominate the world of Internet Applications. Several years ago, Nielsen (2000c) noted that since the arrival of the World Wide Web, almost no progress had been made in specialized Internet clients. And the situation has not changed much since then. Given BBAs' many weaknesses, how can we explain their great popularity?

This question is not difficult. BBAs are popular because web browsers—like the World Wide Web itself—are immensely popular. Nearly every computer connected to the Internet already has a web browser installed, so BBAs can be run without installing any additional software. Even users away

from their own computers can be assured easy access to BBAs, since any machine with Internet connectivity and a browser will do. In fact, using the browser as a platform provides a double benefit, both eliminating the need for installing applications and making them (largely) platform independent. These benefits are especially significant for organizations deploying internal applications to a large number of employees who may use a variety of machines and operating systems.

Not only do most computers have web browsers installed, but most users are familiar with browsers. Indeed, many Internet users view themselves as accomplished browser users. Even web novices can become familiar with browsers quickly since most are easy to learn and use. User familiarity and competence with browsers translate into user comfort with the applications that run in browsers—the BBAs. BBAs seem familiar to the user, due to the familiar browser interface, and the user feels competent to use them, given his or her competence with browsers. The many people who use multiple BBAs may feel even more comfortable since the various BBAs all share the common browser-based interface.

Would it be fair to argue that this sense of BBA familiarity and competence translates not only into user comfort but also ease-of-use? The argument would be as follows:

> Since browsers are easy to learn and use, and especially since many people already know how to use browsers, applications based on browsers (BBAs) are easy to learn and use. In other words, if you can use a browser—and who cannot?—using a BBA is easy. Moreover, since most Internet Applications are BBAs, learning and using one more BBA is easy since it has the same browser-based interface as the others. In human-factors lingo, usability is enhanced by positive transference from one BBA to another.

This argument, however, does not hold up to scrutiny. BBA comfort may lead people to believe that BBAs are easy to use, but the ease-of-use may be illusory. When an Internet Application is implemented on top of a web browser, the user invokes the application's functionality by interacting with the application's web pages, filling in forms and clicking on appropriate buttons to perform activities. Although the browser's controls may be familiar and easy-to-use, the BBA's usability is determined by how easily the application-specific web pages can be used. There is no reason to believe these would be any easier to use than a specialized client performing the same functions. In fact, the BBA is likely to be more difficult to use, due to the many practical problems described earlier (especially those in the "Clumsy interfacing and limited functionality" section). Trying to use the various browser-based e-mail systems illustrates these observations.

But what about the positive transference claim—that the common browser interface across numerous BBAs contributes to ease-of-use as one moves from one BBA to another? This argument also fails because each BBA sports its own forms and buttons. Even within a particular application type—say, e-mail—one finds significant differences in the browser-based systems. From a usability perspective, someone with multiple e-mail

providers would be much better off using a single, specialized e-mail client than employing the sundry web-based e-mail systems. Indeed, very different specialized applications that share a common GUI might seem more similar to each other than would various BBAs that share the browser platform (but rely on page-based widgets for most of their interaction).

Put differently, the potential for positive transference that follows from what BBAs have in common may go unrealized, or even be surmounted by negative transference, due to the numerous differences among BBAs. Here are just a few of the examples identified by Silver and Ward (2004):

- We have already seen that BBAs differ one from another in how they handle the "Back" button and printable versions. Misusing these features leads to potentially severe penalties (such as generating duplicate transactions or exiting the browser prematurely).
- BBAs differ in how they treat the "Enter" key when a user is completing a form. Some BBAs treat "Enter" as a signal to post the form; others do not. In the former case, if the user mistakenly presses "Enter" prematurely, the partially completed form is sent to the server. At best, the user is inconvenienced by the delays and discontinuities associated with having the page returned to him or her for completion. At worst, the BBA acts on the form as though it were complete, possibly causing all sorts of problems for the user.
- Financial institutions—banks, credit card providers, insurance companies, investment firms—typically provide a logout button on their pages, but the button goes by various names—logout, signoff, exit, and so forth—and is positioned in varying locations. So a person who uses many financial BBAs, despite being very familiar with such applications, must still search for this important button in any given application.
- Shopping-cart sites differ in their behavior if someone proceeds to checkout without pressing the "Update Quantity" button: Some BBAs process the change; others ignore it.

While BBA familiarity certainly affords some benefits, the substantial and consequential differences among even similar applications impose significant costs. These inconsistencies can make it difficult to learn new BBAs or remember how to use old ones.

While BBA ease-of-use may be more perception than reality, BBAs do offer two other benefits to their users. First, by using the browser as a unifying platform, users do not have the inconvenience of jumping in and out of a variety of applications programs. Running all Internet Applications on top of the browser contributes a sense of seamlessness and continuity to the user experience by eliminating the need for users to have the operating system open and close applications. In some sense, you can live your on-line life in a single window (the browser window). When one follows hyperlinks to jump from one BBA to another, the experience is even more seamless. Such is the case when a person follows an advertisement from one website to another or when an e-mail message embeds a

link to an on-line store. Moreover, browsers are perfectly suited for searching for applications. Using the same software to run the application as you used to find it creates a seamless experience; you arrive at the site and come up running.

A final factor favoring BBAs is that many Internet Applications require, or at least benefit from, some information browsing capacity, so building these applications on top of a browser affords a ready-made, high quality, highly integrated browsing capability. E-commerce applications fit this description; browsers may be terrible for transacting business, but they can be great for browsing products.[11]

So the main benefits of BBAs are these:

- Application availability (due to web browser availability) and platform independence (limited somewhat by browser and version incompatibilities).
- User comfort (due to web browser familiarity, web browser competence, and perceived ease-of-use—however illusory it may be).
- A seamless (integrated) user experience.
- A good browsing capability.

It is easy to see why, historically, BBAs became pervasive in the world of Internet Applications. Perhaps some Internet developers made the conscious, rational choice to base their applications on browsers given the benefits for users: availability, comfort, perceived ease-of-use, seamlessness, and good browsing. Perhaps some saw the benefits for themselves: ease-of-deployment, since updates would only have to be performed on the server (Graham 2004), and ease-of-development, since they would not need to develop a new client (although the latter may also be illusory since server-side development can be so messy). And for some maybe the move to BBAs was not so much a deliberate choice as the only avenue considered at a time when, to many, the World Wide Web and the Internet seemed synonymous.

## 4 The future of Internet Applications: research challenges

The foregoing analysis suggests a troubling situation: BBAs dominate the world of Internet Applications, and are likely to continue to do so for some time, yet they are fundamentally flawed. Further study of this phenomenon

[11] Support for browsing is a natural for BBAs but this virtue does not belong exclusively to them. SCAs can also offer good browsing. Increasingly the few specialized Internet client applications that we have are including scaled-down browsers (mini-browsers) or the ability to launch a full browser. For example, such media players as Windows Media Player, RealPlayer, MMJukeBox and WinAmp provide their users with access to web-based media guides and the like. Quicken embeds a mini-browser for accessing financial news. Yahoo's FinanceVision—now defunct but an excellent example of an SCA—included a generalized browser window. Indeed, Microsoft Internet Explorer and Netscape Gecko (the engine behind Netscape Navigator, Mozilla, and Firefox) can be embedded in specialized applications.

has two main thrusts: (1) to examine more fully and empirically the consequences of employing BBAs and (2) to propose and analyze ways to improve upon the current situation.

### 4.1 Studying actual and perceived BBA usability

The analysis in this paper makes strong conceptual claims that BBAs pose significant usability problems. But these pejorative assertions require empirical verification. In particular, the following questions need empirical study:

- To what extent are the BBA usability problems identified in this paper manifest in actual use?
- If manifest, how consequential are these problems for BBA users and for companies that employ BBAs? For instance, do on-line shoppers lose substantial amounts of time? Do on-line merchants lose substantial amounts of sales?

If the problems identified here theoretically are found to be consequential empirically, further research will be required to reconcile the deficiencies of BBAs with their apparent popularity. The apparent popularity of BBAs notwithstanding their deficiencies might simply reflect the dominance of the combined strengths of BBAs over their weaknesses. A more intriguing explanation would be that people may perceive BBAs to be easy-to-use despite their many usability problems. This possibility raises a third empirical question:

- To what extent do people perceive BBAs as easy to use?

Should empirical studies bear out this conjecture—that perceptions of usability are high but actual usability as traditionally measured is low—still more work will be required to understand the anomaly.

Empirical findings concerning the perceptions, extent, and consequences of BBA problems might also shed light on another issue touched on earlier:

- Why have developers been slow to adopt work-arounds that could reduce BBA difficulties?

For instance, if the usability consequences of BBA deficiencies are not so great, or if users do not perceive them to be so great, developers may have little incentive to invest great effort in remedying them.

Since BBA usability may be degraded not only by the endemic deficiencies of the browser as a platform—the primary focus of this paper—but also by inconsistencies in how different BBAs implement similar features, another set of questions requiring empirical study is the following:

- How great are the inconsistencies across BBAs? To what extent do such inconsistencies impair usability? To what extent do people perceive the inconsistencies and any concomitant usability losses?

**Table 3** Further research in BBA usability: topics for empirical study

| |
|---|
| • The manifestation and severity of BBA usability problems |
| • The consequences of BBA usability problems |
| • User perceptions of BBA usability |
| • Reconciling perceived and actual BBA usability |
| • The development and diffusion of BBA work-arounds |
| • Usability issues surrounding inconsistencies across BBAs |

All told, concerns over BBA usability open the door to an extensive program of empirical research. Table 3 summarizes the behavioral issues requiring investigation.

## 4.2 Improving the situation

If we find it troubling that a class of defective applications dominates the Internet, then the key question becomes what can be done to remedy this situation. A full treatment of this question is well beyond the scope of this paper, but we can contemplate the nature and range of possible solutions.

The challenge for research and practice is to develop technological alternatives to the current BBA approach that remedy BBA deficiencies while leveraging BBA strengths. Any true solution would need to provide clients with local computation, statefulness, more granular information exchanges, and more extensive interface capabilities while retaining the availability, platform independence, and ease of deployment of BBAs as well as their perceived usability, seamlessness, and support for good browsing.

At first glance, it is tempting to divide the approaches into two sets: browser-based solutions and specialized-client solutions (such as Eudora and Microsoft Outlook for e-mail). But these approaches need not be distinct. Indeed the most promising solutions might include elements of each, since they would need to incorporate the strengths of each.

Consider first the extremes. The pure BBA solution would be to extend the browser further, just as HTML and javascript were extended over the years. Indeed, such advances as XML-enabled browsers and the recently introduced XForms promise to solve some, but not all, of the BBA problems. The obvious strength of this approach is that it retains the full benefits of browsers. The weaknesses are that this approach will likely fall short of remedying all the BBA deficiencies and that it may involve constructing additional kludges on top of an already shaky scaffold. Moreover, relying on enhanced browsers poses availability and deployment problems since users would require newer versions of their browsers and, as Nielsen (2000a) notes, many users are slow to migrate to new versions of the browser. The incremental "extend-the-browser" approach also needs to be viewed cautiously since, ironically, today's troubled BBAs were brought into existence by such browser extensions in the past.

At the other extreme are specialized-client applications (SCAs, sometimes referred to as "thick clients"). SCAs would solve the BBA problems—since they are not constrained by the browser—but if deployed as stand-alone applications SCAs would not have the availability, platform independence,

and seamlessness of today's BBAs. Indeed, many organizations may believe with good reason that continuing to provide their employees and customers with platform-independent applications requiring no installation is more important than offering more usable applications. Lack of platform-independence and ready availability therefore pose significant barriers to SCA adoption.

The most promising approach could be "hybrids" that retain the role of the browser but include the power to produce applications with all the features of SCAs. A hybrid approach would need to produce applications that (1) provide the full features—functionality and interface—of an SCA, (2) can be launched from the browser, (3) are downloadable dynamically, requiring no installation, and (4) are platform-independent. Here are some possible hybrid approaches:

- Some web sites launch Java applets from the browser to leverage the benefits of browsers while avoiding their constraints. This is largely why Java was invented: to provide a full-featured programming language while using the browser rather than the operating system as a platform. Combining Java with XML to produce applications is an even more powerful step in this direction (Bosak 1997). But users are sometimes confused to find themselves using the browser to interact with an applet rather than with a web page.
- Plug-ins such as Macromedia's Flash could be used to provide runtime environments for applications. The applications would run on top of the plug-in rather than directly on top of the browser. Thus they would still be running in the browser but they could take advantage of the plug-in's functionality and interface. Various companies have recently used Flash to build such specialized applications (Nielsen 2002). While these applications are not constrained by the browser's limitations, they are limited by what the plug-in environment can support, require installation of the plug-in, and may confuse users who expect to interact with these applications as they would with a web page.
- Microsoft's .NET initiative can facilitate the development and deployment of Internet Applications by using such features as .NET Windows Forms, .NET Web Services, and the .NET Framework (Platt 2001; Landgrove 2003). This approach could retain much of the availability and seamlessness that makes BBAs popular while providing the functionality and interface of an SCA. In particular, the applications could be launched from a browser, without requiring prior installation. And they could be coded in various programming languages. But they would not actually run inside the browser and would depend on Microsoft's .NET Framework as a platform.

We need not be limited by existing technologies. New languages could be invented to meet the various needs of Internet Applications. Bos (2004), for instance, offers an interesting proposal for a light-weight web-application language that could run on top of the browser or on some other user agent. The "W3C Workshop on Web Applications and Compound Documents (2004)" site (http://www.w3.org/2004/04/webapps-cdf-ws/) contains a variety of proposals for the future of web applications.

**Table 4** Initial requirements for the next generation of Internet Applications

| |
|---|
| **Retain BBA strengths** |
| • Availability (no installation required) |
| • Platform independence |
| • Perceived usability |
| • Seamlessness |
| • Good browsing |
| **Remedy BBA weaknesses** |
| • Support more granular information exchanges |
| • Expand local computation |
| • Provide statefulness |
| • Offer a better GUI (with direct manipulation and without piggy-backing problems) |
| **Offer a viable (attractive) migration path to the new technology** |

Each approach—improving BBAs, replacing them with SCAs, or a hybrid technology—has its benefits and limitations. The hybrid approaches seem most promising, given the tensions and trade-offs between pure-BBAs and SCAs. But, of course, not all hybrid approaches are equally desirable. The first task in designing the next generation of Internet Applications must be to specify a clear set of requirements. Based on the discussion here, Table 4 provides a starting point for this endeavor. Given a set of requirements, each candidate technology must be fully elaborated and its benefits and limitations assessed. While the technical merits of each candidate solution are important, so is its likelihood of adoption. Given such widespread current deployment of BBAs, how easy will it be for those who deploy, and those who employ, today's BBAs to migrate to the proposed alternative? And how likely are they to do so? No matter how desirable the proposed technology, altering the status quo is likely to be a considerable challenge.

## 5 Conclusion

In recent years, much of the technical attention concerning Internet Applications has focused more on the server-side or back-end of the application. However important and technically challenging the server-side may be, we must not lose sight of the front-end, the client-side of the interaction. Past choices in the development of applications—deliberate or not—have brought us an Internet where BBAs dominate. These BBAs possess a number of strengths, including user comfort, availability (no installation required), platform independence, and seamlessness. But as we have seen, the weaknesses of the browser as a platform—its statelessness, page-orientation, and limited local computation—as well as the piggy-backing of one application on top of another—lead to numerous usability problems for BBAs (Tables 1, 2). As we look to the future of Internet Applications, we must be more deliberate in pursuing a course that will rid us of the problems that plague today's BBAs, through some combination of improving and replacing the

browser as a platform. The immediate challenges facing us are twofold: (1) to understand more fully the practical consequences of today's BBAs and (2) to propose and assess viable technological solutions that can remedy the BBA problems while retaining their strengths. We need to be aware, as well, that the degree of investment—financial and psychological—in the current browser-based approach and the concomitant cost and difficulty of departing from it may pose a nearly insurmountable barrier to improvement.

**Acknowledgment** I appreciate the helpful comments made by Lynne Markus, Amjad Umar, Sidne Ward, Burt Swanson, Michael Shaw, and two anonymous reviewers on earlier versions of this paper

## References

Bos B (2004) Setting the scope for light-weight web-based applications, World Wide Web Consortium. http://www.w3.org/People/Bos/webapps.html

Bosak J (1997) XML, Java, and the future of the Web, Sun Microsystems. http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html

Fielding RT, Taylor RN (2002) Principled design of the modern web architecture. ACM Transact Internet Technol 2:115–150

Graham P (2004) Hackers and painters: big ideas from the computer age. O'Reilly Media, Sebastopol

Landgrove T (2003) Web front-ends versus Windows. TechRepublic, ZDNet

Nielsen J (2000a) Designing web usability: the practice of simplicity. New Riders Publishing, Indianapolis

Nielsen J (2000b) Reset and cancel buttons, Alertbox. http://www.useit.com/alertbox. Cited 16 April 2000

Nielsen J (2000c) Finally progress in Internet client design, Alertbox. http://www.useit.com/alertbox. Cited 30 April 2000

Nielsen J (2002) Flash usability and web-based applications, Alertbox. http://www.useit.com/alertbox. Cited 15 November 2002

Platt DS (2001) Introducing Microsoft.NET. Microsoft Press, Redmond

Shneiderman B, Lazar J, Ivory M (2003) Introduction: web navigation. IT Soc 3(1):i–vii

Silver MS, Ward SG (2004) Browser-based applications: positive or negative transference? In: Proceedings of the 10th Americas conference on information systems, New York, pp 3169–3176

Tognazzini B, Nielsen J (2001) Beyond the browser, eWeek. http://www.eweek.com/article2/0,4149,1252468,00.asp. Cited 26 March 2001

W3C Workshop on Web Applications and Compound Documents (2004) http://www.w3.org/2004/04/webapps-cdf-ws/

Wroblewski L, Rantanen EM (2001) Design considerations for web-based applications. In: Proceedings of the 45th annual meeting of the Human Factors and Ergonomics Society, Santa Monica