Users and Trust: A Microsoft Case Study

CHRIS NODDER

BACK IN THE GOOD OLD DAYS, if you lived in a small town, you wouldn't think twice about leaving your house unlocked while you ran errands, about letting kids play in the streets, or about sharing details of your family's life with other people in the town.

However, as the small town grew, and more new people started to arrive, you might have started to hear about unusual things happening: property disappearing, park benches getting vandalized, strange behavior from the new neighbors.

Over time, you'd learn that maybe it was safer to lock your door, to ask your kids where they would be going, not to lend out your lawnmower. Normally, you would learn this through newspaper articles or stories that friends told you. Sometimes, if you were unlucky, you'd learn through personal experience of having something bad happen to you—something that would never have happened in the good old days.

The Internet has paralleled this move from small town to larger city life. With the advent of the first HTML browsers, the Internet became the World Wide Web, and many new neighbors moved in to what had previously been a relatively trusting small town. The new neighbors brought with them confidence tricks, unwanted mail, viruses, and lots of candy that it really wasn't safe to take.

The major difference between real-life small towns and the Internet is the compressed time scale of the Internet's growth. That growth rate, along with the relative anonymity afforded by the Internet and the extreme ease of creating a presence on the Web, has meant that many regular users of the Internet have not had enough time to build or adjust their perceptions of trust to deal well with the online environment.

Instead, the responsibility for helping users decide whom to trust online has fallen to the infrastructure providers: manufacturers of browsers and email programs, antivirus applications, and spyware scanners.

In the early days of the World Wide Web, fewer people were attempting to exploit the gaps in technological or social trust online. As the technologies matured and the user base grew, such exploits became more lucrative.

To counter this rise in the number of exploits, the infrastructure providers have incorporated technologies and user interface elements aimed at shaping users' behaviors, teaching them whom they can trust, and, where necessary, giving them the cues they need to make trust decisions. However, the code that infrastructure providers produce is much better at dealing with problems that have a logical right and wrong outcome (virus/no virus) than problems that have shades of emotional response, such as social engineering attacks.

Obviously, Microsoft is one of those technology providers. This chapter describes how research into users' trust mechanisms led to changes in user interface design philosophy for Internet Explorer and several other products at Microsoft. The changes represent a first step in respecting the emotional aspect of trust decisions, and in giving users the information they need to make good trust decisions within Microsoft applications.

Users and Trust

As part of continual usability research, usability engineers at Microsoft had observed hundreds of users answering questions posed by the computer (consent dialogs) in Internet Explorer, Windows Client and Server, Microsoft applications, and other companies' products. It was clear that users often weren't following the recommendations that the products made. The question was: why not?

Having seen this behavior over multiple usability sessions, we ran some specific studies to gain more insight. We conducted in-depth interviews about trust with 7 participants, and lab-based research with 14 more. We then used the results of this work to develop user interface prototypes that incorporated design elements suggested by the initial research, and observed a further 50 participants working with various iterations of the designs in different trust scenarios. Later, we had the chance to verify the concepts and designs with participants who were helping us evaluate the interface for Windows XP Service Pack 2 both in multiple lab sessions and through feedback and instrumentation from a very large user panel.

We found that it was not just that users didn't understand the questions being posed by the computer, although that was definitely part of it. It was also that the computer was not their only source of trust information. It turns out that users aggregate many "clues" about trustworthiness and then trade those off against how much they want the item in question. Interestingly, computers weren't presenting all of the clues that they could have to help users, and some of the clues they were presenting were so obscure that they just confused users.

WHAT IS USER RESEARCH AT MICROSOFT?

A User Researcher's role is, specifically, to bring data to the table about how people interact with PCs, what they want to do but can't do, and what's coming around the corner technologically that they'll need to do but don't even have a clue about. Then, the researcher works closely with designers, user assistance creators, and the feature program managers to ensure that we build the right features to meet user scenarios, that those features work the way users expect, and that all the other myriad design considerations are taken into account.

User Research at Microsoft draws on multiple data sources to build a picture of user behavior and user needs. Along with traditional lab-based studies of everything from paper prototypes to finished code, we also conduct site visits to watch users in their own environments, perform in-depth interviews on specific topics, and administer large-scale international surveys. In addition, we rely on community feedback and our panel of instrumented users. This user panel is composed of regular people who have opted to run special software that provides us with data on their computer settings and their behavior. Interpolating from all of these areas as well as market research and published academic studies helps us to understand what drives users.

Lab work (usability "testing") is actually quite a small part of what User Researchers do. While we're in the lab, though, along with measuring users' success on tasks, we also measure things like desirability, learnability, and comprehension. Having the controlled environment of the usability lab allows us to isolate specific issues more easily. We iterate the design and test again with more users until we get the user experience to a point where participants can be successful and satisfied with the task.

The data serves other purposes too. Knowing what proportion of users are likely to perform a certain task—say, one that keeps them more secure—is very useful in meetings where other team members are inclined to make wild guesses based on their own experiences. The realities can be very sobering. Being able to show how that proportion grows after injection of some user-centered design into the task is a major encouragement for teams to think and design in a user-centered way.

Users' Reactions to Trust Questions

Trust questions appear at many points in computer interfaces. Typically, they are shown as dialogs when the computer requires input or consent from the user before proceeding—for example, before downloading a file or before performing an action that could lead to data loss.

These trust question dialogs are often designed to serve a useful dual purpose of both informing users and requesting input. During usability research at Microsoft, we found that these dialogs regularly failed on both counts from users' perspectives. Some observations we made about the information and questions in trust dialogs were:

Often, the question being presented is a dilemma rather than a decision
: In such cases, the user feels that he has no way of choosing between the options being presented. Without suitable assistance, the user will be forced into making a choice that may or may not be the right one for him. Superstitious behavior builds up this way.

Computers can't help interpret emotional clues because they behave in a purely logical way
: This means that computer software has to defer decisions to users even if the outcomes of those decisions look logically "bad."

Users don't want to deal with the trust issues presented to them
: The larger the scope of the decision, and the less context that is given, the more likely they are not to consent to the action being presented to them.

Users don't want to reveal personal data
: The closer the question being asked is to revealing personal data, the less likely users will be to comply.

So, users do not respond to dialogs the way we might anticipate. This is because they are often forced to make a decision that is at odds with their understanding of the situation, and the information being provided is both incomplete and only partially intelligible to them.

Users' Behavior in Trust Situations

The research I performed also showed that users have some interesting things going on in their heads during their interactions with trust situations on their computers:

What users say they'll do and what they actually do often differ
: For example, while users may claim to run virus-checking software, and be careful to whom they give personal data, in reality they are more lax than they describe.

Users don't necessarily want to think about the consequences of their behavior
: They may "forget" that they've changed a setting or allowed a certain application to access their data, and thus be confused when they suffer consequences such as a broken user experience or unexpected email.

Users make one-off decisions about trust

Trying to get them to make a global decision to "always do X" will upset them and potentially lead to their declining that global decision where, in fact, they would want to accept in some specific instances.

Users conceive of security and privacy issues differently than developers do

Users don't have the background understanding of issues, are surrounded by myths and hoaxes, and have a different relationship with "junk" mail than application developers do.

Users have many superstitions about how viruses are propagated

They confuse hacking and viruses. They also interchange terms for software bugs and viruses. They often fall prey to virus hoaxes in an attempt to protect themselves, while simultaneously engaging in risky behavior likely to lead to virus transmission.

Users do not tend to consider events requiring trust decisions in the same way that technologists do. This is because their focus is not on the technology, but *on the outcome of the trust event and its impact on their lives.*

Security Versus Convenience

The worst dilemma for users, and the one that is also the hardest to resolve through user experience design, is that from a user perspective, increases in security are most frequently accompanied by a reduction in convenience. Likewise, when users try to accomplish a task in a convenient way, they often encounter security warnings.

For instance, choosing to set the browser security level to High in Internet Explorer or other browser products will turn off many of the features of the product that can be used to exploit users. However, this same action can degrade the browsing experience to a point where most users will be dissatisfied, as they will no longer have access to the plug-in components and scripting functions that they have come to expect on a web site. It is this dilemma that user experience designers must seek to resolve for users, presenting them instead with understandable options that allow them to perform their tasks with a minimum of inconvenience.

Making Decisions Versus Supporting Decisions

It is important to note that the emphasis here is not on allowing the computer to make trust decisions, but on how a computer can assist users with their trust decisions. Of course, there are some instances where the computer can make that decision—for instance, when it detects the presence of a known virus in something the user plans to download. Here, the decision is easy—protect the user from the virus. Computers can be programmed to make this kind of decision. Most of the time, however, the decision is less clear cut, and so it still rests with the user. The challenge is to achieve the correct balance between exhausting the user with multiple questions and automating the process to the point where the computer runs the risk of making erroneous decisions.

Having observed that users have a tendency to simply dismiss any dialog that gets in their way, the tendency among interface designers is often to try to remove the dialog. If the dialog can be completely removed (if the computer can make the decision), that's great. If, however, the dialog still needs to exist, our studies have shown that users make a much more secure, appropriate, reasoned decision if the dialog is presented in the context of their task.

Placing the decision in an initial options screen or hiding it in a settings dialog removed in space and time from the point where users carry out their task requires them to think in a logical rather than an emotional way about a class of task rather than about a specific instance.

As noted earlier, users found it easier to make a specific decision rather than a generic decision. It was much easier for them to agree to trust a specific person at a specific time for a specific transaction than to agree to trust a whole category of people every time a transaction occurred.

Users could easily make a decision without too much interruption to their task if the dialog presented the facts they needed in a way they could understand. We classified this as *presenting a decision, not a dilemma*.

For common or repetitive tasks, obviously the fewer interruptions a user experiences, the better. In these situations, it makes sense to give the user an option to always apply his current decision to the situation. If you can scope the situation suitably, the user will be happy to have that decision applied consistently.

For less common tasks, it's not necessarily the number of screens between a user and his goal that determines the quality of the interaction. Instead, a major factor is whether all of those screens are perceived by the user to be flowing toward his end goal.

After eliciting from users some of the clues they use, and understanding the philosophies that they bring to their trust interactions, we worked out which clues can be provided by a computer, and then worked out how and when to present them in the trust process such that they aided in the decision. The tone of the interaction was dictated to a large degree by a wish to stay within users' comfort zones while simultaneously educating them.

Consent Dialogs

The name of this much maligned interface element suggests something that isn't always apparent in the design—dialog boxes are supposed to be a conversation ("dialog") between the computer and the user. The consent dialog has a specific conversation topic: "Do you want this thing to happen?" Frequently, consent dialogs ask trust questions. If the question is well phrased, users should have little difficulty making a decision, completing the dialog, and continuing on their way. However, well-phrased dialogs seem to be difficult to design.

After observing many users working with dialogs in the lab, I would suggest the hierarchy of decision points shown in Figure 29-1 that users follow in order to continue with what they perceive as their real task—the one that the dialog box interrupted.

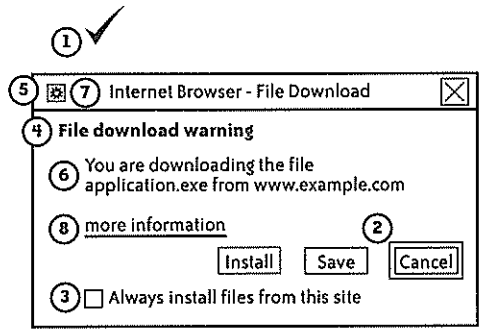

FIGURE 29-1. Decision points users make in evaluating a (hypothetical) consent dialog

Note that before users even start to evaluate the information provided in the dialog, they have already engaged their emotional feelings about the situation (1, in the figure). If these feelings are strong and positive (a friend or someone the user trusts recommended the application, for instance), then his desire to continue down this path may overrule his usual caution. In this situation, the user has made an emotional decision that is unlikely to be changed by any logical information or warning on the screen.

Users also gravitate toward the buttons and other controls (checkboxes, radio buttons) on the screen—they realize that these are the elements that will propel them toward their desired outcome, so they start by reading the text on and next to controls (2 and 3, in the figure) to see if these provide enough clues to let them continue. The wording of the buttons, and even which one is highlighted by default, are clues that users can employ in their trust decisions.

If they are unsuccessful after reading the button labels, users typically proceed to read the text on the screen. The primary statement is read first, normally because it is first on the main page area and in a larger or bold font (4, in the figure).

Dialogs with graphical elements also assist users with the trust decision—have they seen that graphic before? Is it from a company they trust? Obviously, if the only graphics are system elements such as the icon of the program that launched the dialog box, then users may gain a false sense of security (5, in the figure).

Now users are really scraping around for clues and cues to help with their trust decision. Body text in the dialog box, the title bar, and potentially help links from the dialog box are all read with increasing levels of irritation and desperation, if at all (6, 7, and 8, in the figure).

Consent Dialog Redesign

There is very little that a computer can do to counter an emotional decision by the user. Unfortunately, this is the place where social engineering attacks (e.g., so-called "phishing" attacks) happen—before the computer interface has a chance to influence the decision.

However, there are user interface design elements that can assist users with making trust decisions if they choose to make use of the information presented by the consent dialog. Figure 29-2 shows these elements laid out in a figurative consent dialog. For an example of how these elements are used in a real consent dialog, see Figure 29-8.

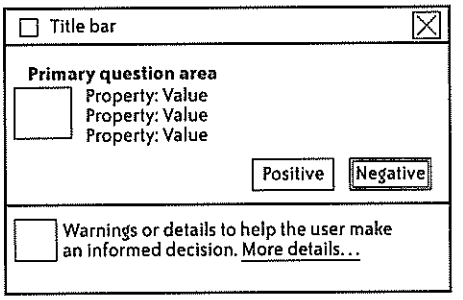

FIGURE 29-2. Figurative elements of a more informative (hypothetical) consent dialog

Considering that users tend to gravitate to the buttons on the dialog, the default button (which will be clicked if the user simply presses Enter) should always follow the path recommended by the computer. This may not necessarily be the most secure path ultimately, but instead may be the best tradeoff of security and convenience.

The button labels should also be verbose enough to allow users to make an informed decision. For instance, consider the difference in meaning between the button pair Install and Cancel and the button pair Install Anyway and Cancel. In the second instance, the button label itself contains a caution that something may require further investigation before installation. Of course, in conjunction with the button labels, the dialog text needs to make it clear what the issue is that requires investigation, and suggest a path of resolution.

The dialog should contain a summary statement or focused question placed in the primary area of the dialog box. This statement or question should provide users with an understanding of the decision they are being asked to make (what they are "consenting" to).

If further text is required, it should follow the summary statement, but what appears from testing to be more useful to users at this point is evidence that they can use to evaluate their response to the question being asked. In an ideal world, this information would probably contain recommendations from trusted sources; in reality, with today's software and certification mechanisms, the best that can be done is often to provide what information the application knows about the situation to the user in the form of clues he can use to help him arrive at a decision (the Property: Value pairs in Figure 29-2). Typically, this means items such as the filename, publisher, and download location of a file, the name of

an application or user who is requesting something, or other such short strings. The computer can often also assign some degree of confidence to these clues by letting the user know whether the statement is corroborated by a certificate. Although users often have totally erroneous concepts of what certification entails or guarantees, this can be abstracted to an extent within the dialog.

For users who still do not have sufficient information to make a decision, there is space to provide additional assistance text and even a link to a help article. Note that this information is placed in a separate area at the bottom of the dialog that does not interfere with the primary dialog controls. User testing has shown that people who are seeking out this information will still find it in this location, whereas if it is placed above the action buttons, it interferes with the task for users who do not require the additional, often generic, text. This information is accompanied by an icon that rates the severity of the decision outcome—informational, warning, or danger/stop.

Windows XP Service Pack 2—A Case Study

In August of 2003, senior executives at Microsoft made it clear that continuing with a reactive patching approach to fixing security exploits was insufficient. Instead, users needed a system that was able to deal with whole classes of exploits and to prevent infection. The technologies required to achieve this required a major rewrite of several components of the operating system, and also required us to make changes to the default behaviors in Windows.

While this last point may not sound like a big deal, this would be the first time in the history of Windows that a service pack release made significant changes to the user experience of the product. By working with the program managers for Service Pack 2, we presented evidence from user studies, surveys, instrumentation reports, and site visits to the Windows executives demonstrating that the release would not be successful without several large-scale changes to the user experience.

This data had been collected primarily from research into Passport, Hailstorm, XP Service Pack 1, and early work on Windows Codename "Longhorn"—the next version of Windows. It was fortuitous that it could be applied so directly to the problems that were addressed by XP Service Pack 2. We also worked to ensure that many of the changes could be controlled through group policy in order to satisfy corporate customers.

Some of the major attack points in Windows are found in the Internet-related features—Internet Explorer (IE), Outlook Express (OE), and the Application Execution Service (which prompts users to save or run downloaded or attached files). Various other system components, such as the Windows Firewall and Automatic Updates, are designed to help protect users but require user interaction in order to work correctly.

The user experience focus for SP2 was on making security and privacy "just work," and where that was not possible, making the security and privacy issues understandable to users so that they can make informed decisions.

To that end, we took several components of Windows and gave them a design overhaul. The following sections show before and after images with some commentary on the reasons for the changes. Most frequently, the reasons are based on the consent dialog redesign described earlier.

ActiveX Dialogs

It was very clear from user testing that the ActiveX dialog box in Windows XP and XP Service Pack 1 (XP SP1) (see Figure 29-3) was not very successful at giving users the information they needed in order to make an informed consent decision.

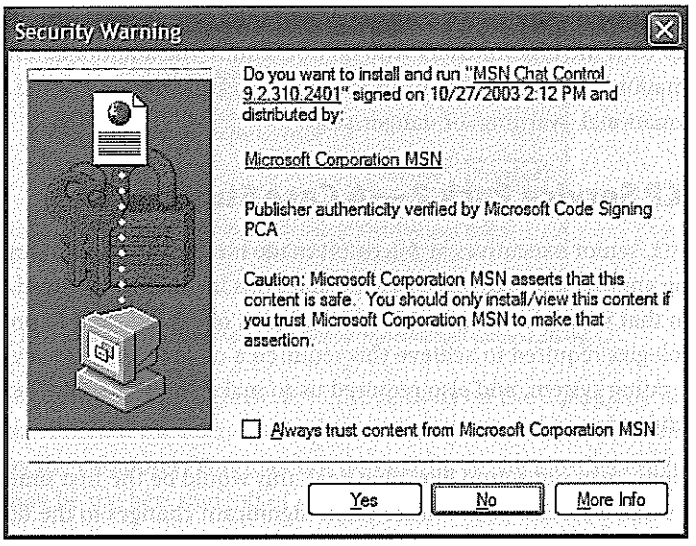

FIGURE 29-3. Original ActiveX consent dialog

Instead of just applying the consent dialog redesign to this interface, we considered ways of also increasing user satisfaction. This dialog box typically appears while a page is loading—often before any content has appeared on the first page of a site that a user visits. As such, it is both a frustration because it gets in the way of the user's primary task, and a dilemma because the user cannot judge the value of agreeing to the dialog box (if she even understands what it is asking).

Taking a design element from Outlook Express, where additional information about email format options and blocked content is displayed in a "bar" in the message title area, we developed an in-context but discreet notification called the Information Bar, which animates down from the Internet Explorer Toolbar.

The Information Bar (shown in Figure 29-4) provides the most condensed information possible to inform users in context that the web site they are on is requesting that they consent to a specific action. They can continue to browse the Web without responding to the Information Bar, or they can interact with the bar to enable downloads of files or ActiveX controls, or to enable blocked pop-ups.

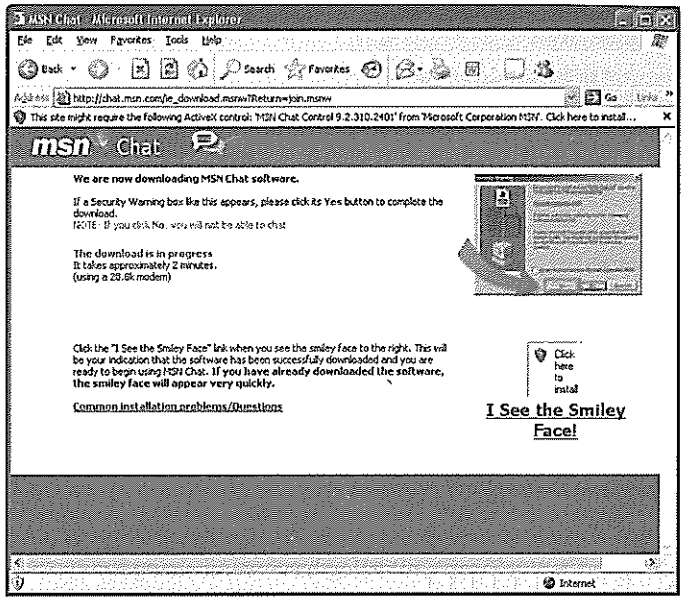

FIGURE 29-4. The Information Bar in Windows XP Service Pack 2

For ActiveX controls, clicking the Information Bar produces the redesigned consent dialog shown in Figure 29-5.

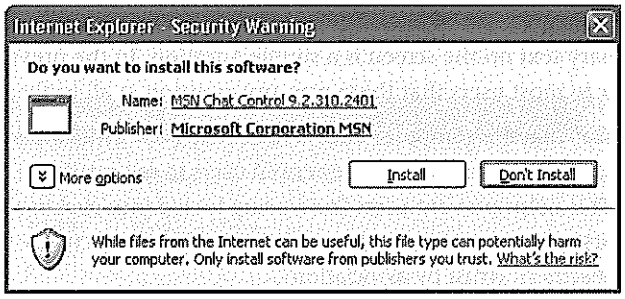

FIGURE 29-5. The updated Active X consent dialog; even if they read nothing else, users will see their options from the button labels

The consent dialog from XP SP1 had contained a checkbox option to "Always trust" the given publisher, and thus not be pestered by these consent dialogs in the future. Once ActiveX controls got co-opted for unsavory uses, we observed users wanting a checkbox for "Never trust" more frequently than for "Always trust." After trying several iterations of these designs, we ended up with the "More options" button, shown in its expanded state in Figure 29-6.

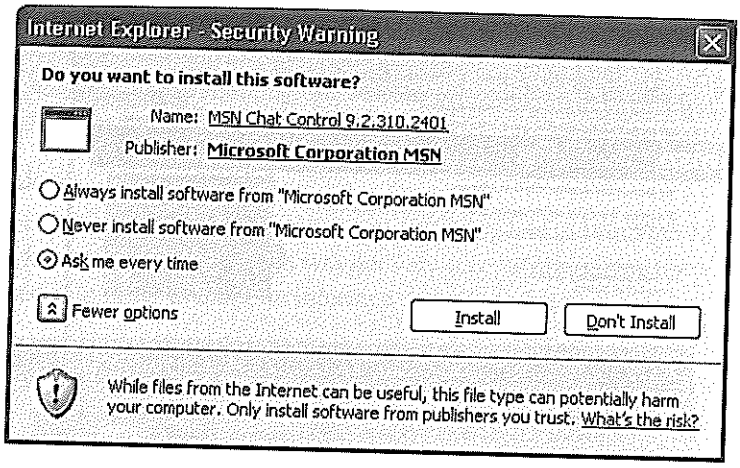

FIGURE 29-6. Expanded version of the redesigned ActiveX dialog; the options include a "never install" choice often requested by users

Note that the redesigned ActiveX consent dialog has several changes:

- *Button defaults.* Buttons have a secure default: clicking Enter chooses the Don't Install option.
- *Button labels.* Button labels change from OK and Cancel to Install and Don't Install. Users can now see the implication of clicking the button (something will be installed) without reading any other text on the dialog.
- *Primary text.* Primary text on the screen is a simple question. In the previous design, it wasn't even clear to most users that there was any question at all in the text.
- *Evidence.* "Evidence" that the computer knows to be true (via certificates) is presented to the user. Clicking on the items allows experienced users to see certificate details and the originating site. This helps prevent users from being fooled into clicking on a consent dialog that appears to be from the site they are on but that was actually produced by a pop-under advertisement (trust by association).
- *Assistance text.* Assistance text is shown in a separate area at the bottom of the screen, with a link to further assistance. This is useful if this is the first time a user has seen this dialog or if the user sees such dialogs infrequently. However, the text is out of the way if the user knows what he is doing.

File Download Dialogs

File download dialogs are mainly triggered by user actions in Internet Explorer. However, sometimes the user can be tricked into downloading software through a variety of code or social engineering manipulations. While XP SP2 removed several of the code exploits, it was still important to ensure that users knew what they were getting themselves into with file downloads, considering that the consequences could be catastrophic if the file were actually a virus.

The file download dialog in Windows XP and XP SP1 (shown in Figure 29-7) was a relatively innocuous dialog that took an informative approach but did not really help the user.

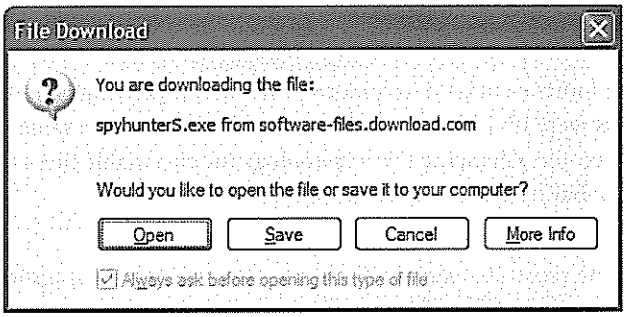

FIGURE 29-7. Original file download dialog; users have to read the entire dialog in order to see the question

Note the relatively useless text "You are downloading the file:", which takes precedence over the real question in the dialog—"Would you like to open the file or save it to your computer?" This question is also misleading. In the case of a Word document download, you truly are "opening" the file. However, in the case of an executable program file, from a user perspective you are not so much opening the file as running it.

Note also how the More Info button has as much precedence on the dialog as the much more important decision buttons.

Applying some of the trust concepts described earlier, we arrive at the dialog shown in Figure 29-8.

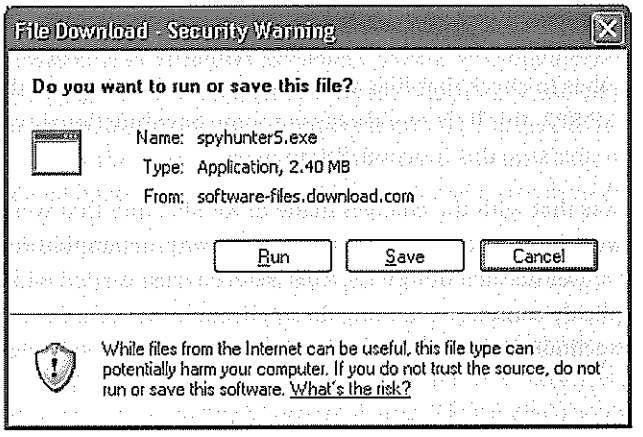

FIGURE 29-8. Redesigned file download dialog; the question is immediately apparent, and the additional information that users need for a trust decision is readily available

Note the purposeful similarities with the ActiveX control redesign:

Button defaults

Buttons have a secure default; clicking Enter chooses the Cancel option.

Button labels

Button labels are more accurate descriptions of the accompanying actions. In the previous version, users were invited to "open" an executable file in the same way that they might "open" a text file. Changing the terminology for executable files is one extra clue to users about the nature of the downloaded file.

Primary text

Primary text on the screen is a simple question. Also, making the question into the primary text reduced its length, as there was no need to refer back to the previous statement in the dialog.

Evidence

"Evidence" that the computer can ascertain is presented to the user. The user can see the originating site, check that the filename matches his expectations, and also check that the size of the file seems appropriate for the type of application he expected.

Assistance text

Assistance text is shown in a separate area at the bottom of the screen, with a link to further assistance. This link replaces the large and distracting More Info button.

Another problem with file downloads in previous Windows versions was that once the file was saved to the computer, it lost any identifying marks. Thus, a user could choose to download a file from the Internet, mindful of the risks, but then later open it on his machine without realizing its potentially dangerous history.

The Application Execution Service (AES) is a part of Windows that most users never have to think about. Its job is to check that files are pretty much what they say they are, and then run them. In XP SP2, this little workhorse got some new functionality, and the issue then became how to message this functionality to users.

The basic premise was that with the changes made in XP SP2, files that were downloaded from the Internet would be tagged as such, and would always retain that reduced level of "trustedness" until a user decided otherwise. This reduced trust carried with it a requirement that users explicitly consent to running the application. As such, the interface for the new Application Execution Service functionality became the consent dialog shown in Figure 29-9.

Again, this dialog has purposeful similarities with the File Download control redesign:

- *Button defaults.* Buttons have a secure default; clicking Enter chooses the Cancel option.
- *Primary text.* Primary text on the screen is a simple question. While it may seem to be an overly obvious question (after all, the user just double-clicked on an icon), often executable files masquerade as documents for just this reason. In these situations, the AES recognizes the subterfuge and presents the consent dialog, thus alerting the user to the true nature of the file.

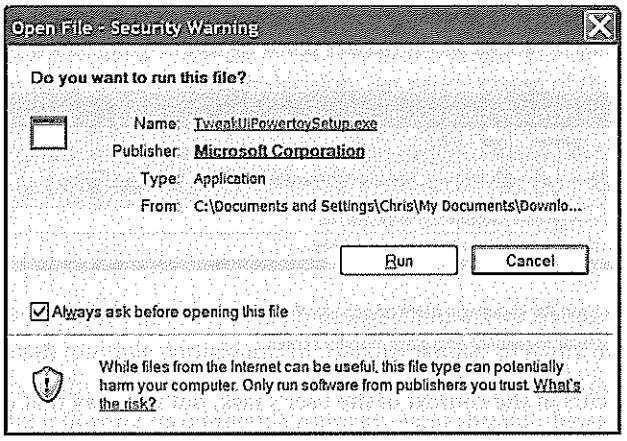

FIGURE 29-9. The Application Execution Service consent dialog; this dialog takes over from the file download dialog if users choose to save rather than run a downloaded file

- *Evidence.* "Evidence" that the computer can ascertain is presented to the user. The user can see the software publisher and check that the filename matches his expectations.
- *Checkbox.* There is a "don't show me this again" checkbox on the dialog, but it is purposefully phrased as a positive statement.
- *Assistance text.* Assistance text is shown in a separate link at the bottom of the screen, with a link to further assistance. This dialog is new, and most users will see it infrequently, so the assistance text is available but not in the way of the primary decision.

Pop-Up Blocking

Pop-up blocking was new to Windows XP Service Pack 2. Pop-ups are windows that are generated by a web page to show additional content. A legitimate use for pop-ups may be to provide a glossary term, assistance, or a product photo when users click a link on a web page. This would allow the user to stay on the same page, but gain additional information that was not deemed important enough to have screen real estate devoted to it.

Unfortunately, the legitimate uses of pop-ups are outweighed many times by the more frustrating or downright dangerous uses. Typically, pop-up technology has been used to display advertisements to users in separate windows, to open (or "spawn") additional windows when the user navigates away from a site in order to trap them on the site, or to cover parts of a dialog box so that it appears to users that they have fewer choices—for example, to hide a part of the ActiveX consent dialog (Figure 29-3) so that only the Yes button appears rather than the No and "More info" buttons.

The issue is that there are times when the pop-up may be seen as legitimate by users, so it is important that users be aware when pop-ups have been blocked, and also have the opportunity to view the pop-up if they think that it is one they invoked. The Information Bar, discussed already in the "ActiveX Dialogs" section, provides a useful mechanism both

for notifying users that a pop-up has been blocked and for giving them access to that pop-up should they choose to see it, as shown in Figure 29-10.

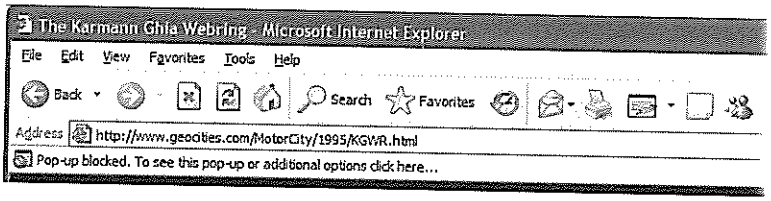

FIGURE 29-10. Information Bar showing a blocked pop-up; the bar animates down from the top of the page area, accompanied by a sound similar to tapping on a glass monitor screen

This notification occurs in the context of the user's task, but in a way that does not prevent him from continuing with his current task. In user testing, this implementation provided the best tradeoff between the ability for users to ignore unwanted pop-ups and to act on ones they wanted to see. It builds on and reinforces the use of the Information Bar for ActiveX control consent. Judging from users' positive remarks during testing and after release, this feature seems to have struck the right balance between security and convenience.

The Ideal

The redesigned consent dialogs presented in this chapter all follow the consent dialog redesign format that we based on user research into trust issues. During the process of creating this format, we came across infrastructure and technical constraints that prevented us from taking the design as far as we would have liked in the timeframe we had.

Certificate Authorities such as VeriSign allow Internet browsers to have a level of confidence that statements made by software publishers are true. However, only a few pieces of information are currently certified, such as publisher name and issuer name. With changes in the Certificate Authority model, it should be possible to also have certified graphics (for the icon area in the dialog redesign earlier), certified membership of communities such as TRUSTe, and even a certified audit of the software with accompanying statements about privacy, security, and reliability. Having a certified audit would bring the concept of certification much closer to users' ideals than today's reality.

Users typically see recommendations from a friend as trustable. This is true even when it is a tenuous recommendation (the friend sends a link to a site that wants to download something). Users also often have trusted friends whom they will ask whether something is safe to download. It is not difficult to imagine mechanisms that allow this type of interaction to occur from within the consent dialog using synchronous communication channels such as Instant Messaging.

Taking this one stage further, consider expert or community ratings. It is highly likely that many other people will already have downloaded a specific piece of software or will have consented to a certain action. Their comments on this software form a trust rating. Certain

users may subscribe to a particular guru's ratings; others may subscribe to Slashdot, CNET, or even Good Housekeeping's ratings. These ratings could appear as part of the consent dialog process to assist users with their decision. An early version of this process has already been implemented in SpyNet, the part of the (post-XP SP2) Windows antispyware application that analyzes user recommendations in order to make determinations as to what is and is not spyware.

Conclusion

Throughout this chapter, I have focused on design solutions for the user behavior that I have observed with multiple participants in usability studies of many products. I have shown examples of those design concepts being turned into practical dialogs within Internet Explorer. The following recommendations can—and should—be applied to any trust interaction on computers:

Let users make trust decisions in context
: Decisions made at the time the issue arises contain the scope users need, which would be missing in a disjointed experience such as a setup dialog. Users who are forced to make isolated decisions are often overly cautious, which can later hurt their experience with the application and lead to suboptimal user experiences.

Make the most trusted option the default selection
: Users are not well placed to make trust decisions. They seldom have all the details. As long as they trust your application, they will frequently select the defaults.

Present users with choices, not dilemmas
: Ensure that the user is able to understand the consequences of choosing a certain option, and that wherever possible there is a trusted way for him to complete his task.

Always respect the user's decision
: Once you have alerted the user to the consequences, the rest is up to him. Your application cannot know the emotional context of a trust decision, only the potential worst-case logical outcome. Users often place more weight on the emotional context.

This last point deserves further emphasis, as it is the cornerstone of a successful trust user experience design. Usable and useful trusted software has to accommodate the emotional and social aspects of users' experience. It must allow them instant gratification, help them to fill in the rest of the picture, and gracefully submit when a user chooses a different path for reasons the computer will never understand.

About the Author

Chris Nodder is a user experience specialist with Nielsen Norman Group. Before joining NN/g, Chris worked as a senior user researcher at Microsoft Corporation. During his seven years at Microsoft, Chris worked on products as diverse as videoconferencing, programming tools for web developers, home networking, online communities, and delivering Internet content over cell phones. In 2004, he was responsible for the user experience for XP Service Pack 2, a major upgrade to Windows XP. He has created personas, reality TV episodes, and even whole rooms ("usertoriums") as ways of getting developers to walk in their customers' shoes.