CHAPTER 6

How Did They Do It?: Monitoring Web Usability

Jakob Nielsen once said that in the first 10 years of the Web, doubling conversion rates was easy—you just had to be sure you weren't making silly mistakes with your web design.

Those easy days are over. Now a usable site is the price of admission. Visitors decide whether to leave or remain within seconds of first arriving. Modern sites must surprise and delight visitors by not only giving them what they need, but also by exceeding their expectations. Of course, if your product or service sucks, even the best website in the world won't save you. However, there are plenty of companies with excellent offerings that fail because users can't interact with their websites. We have to change how we think about usability—not just fixing it, but using it as the basis for innovation and differentiation.

Web Design Is a Hypothesis

Websites are invitations to interact. A site's designer intended for it to be used in a certain way, and its usability is a measure of how easily its target audience can interact with it. Counterintuitive sites discourage visitors, while well-designed ones steer visitors toward the desired goals.

User interaction designers base their interfaces on many factors. They strive to use well-understood controls, familiar conventions, consistent structure, and unambiguous terminology. All of these factors only lead to a hypothesis—an educated guess about how visitors will interact—that then needs to be tested and verified. Looking at websites in this way forces us to describe our assumptions about how visitors will use the site, which in turn tells us what we should be monitoring and testing.

Good designers mistrust their designs. They recognize that there's simply no way to know whether a site design will work without first conducting usability testing. They'll be able to tell you why they designed a particular part of the site a certain way, or point

to the elements of their designs about which they're least certain. And they're probably your strongest allies when it comes to collecting more data on interactions.

Bad designers, on the other hand, will lobby for their particular layouts or color schemes, regardless of whether they work well in production. They care more about seeing their artwork on the Internet than about whether that artwork helps the business. And they're convinced that they have the answers to usability problems. Bad designers who are eager to push the envelope forget that their visitors aren't like them.

Good design is often boring. Some of the simplest, most plainly designed websites rely on familiar controls like radio buttons and simple black-on-white text. While they work, they frustrate artists seeking a creative outlet.

Web design has a creative aspect, but the process of understanding how visitors interact with that design—and adjusting it accordingly—is unflinchingly scientific. The creative part occurs when inventing visualizations and methods of interacting. Once you have some designs to test out, it's time for science to take over from art. Monitoring how real visitors interact with the website settles all arguments on the matter.

Professional web design teams test usability before they go live. This is an essential step in the design process, but it's not enough. People behave differently on a site that's in production. Their expectations are set by where they've been beforehand and what they're hoping to do. They're constrained by their environments, the kinds of browsers they use, the speed of their network connections, and dozens of other factors you can't anticipate or simulate in prelaunch usability testing.

This is where web interaction monitoring comes in. On any given page, visitors can perform four basic actions:

- They can *consume* what's shown, which may involve scrolling, changing the browser's size, starting and stopping embedded media, or changing font size.
- They can *follow a link* contained in the HTML to continue their navigation, either within your site or elsewhere.
- They can *provide data, typically through a form*, using controls such as radio buttons, checkboxes, text fields, or drop-down lists.
- They can *use their mice and keyboards* to interact with elements of the page, which won't necessarily result in any server actions.

By collecting information about how visitors interact with your website, you see which of these actions visitors perform in response to your designs once the site is live. This is the domain of Web Interaction Analytics, or WIA, a term first coined by Tal Schwartz of ClickTale.

Four Kinds of Interaction

Website visitors misunderstand sites in a surprising number of ways. Lacking the designers' understanding of why the site exists and how data is structured, they'll often

fail to notice content or visual cues. They'll click on things that aren't links. They'll navigate through the application in unexpected ways.

We're going to look at four specific kinds of usability problems:

Visitors don't see what you wanted them to
: Content is displayed off the visible screen and visitors don't scroll down to view it.

Visitors don't interact as you intended
: They don't notice elements designed for interaction, such as buttons or links, or they try to interact with things that aren't part of the design you intended.

Visitors don't enter data correctly or completely
: They put the wrong information into a form, linger a long time on a particular form element, or abandon a form halfway through.

You have no idea what's happening on the site
: Visitors saw something you didn't intend, or behaved in unusual ways, and you want to know what they saw.

Let's look at these four issues in more detail.

Seeing the Content: Scrolling Behavior

Your site has many different kinds of pages. Some are simple and transactional, presenting visitors with a few options and asking for a decision. Others, such as search result pages or retailer catalogs, contain paginated lists through which a visitor can navigate. Still others, such as blogs and articles, contain content for the visitor to read.

Each page carries its own design and usability constraints.

- *Transactional pages* must offer visitors a clear decision and show possible courses of action.
- *Search result pages* should make it easy to browse through results, with elements such as pagination.
- *Content pages* should provide readable content alongside headings and images.

Any web designer will tell you that this means important content should appear at the top of the page so visitors can see it without having to scroll down. Of the three kinds of pages outlined above, those containing content are usually the longest. When a page extends beyond the visible window of a web browser, visitors need to scroll to read it all.

Scrolling behavior is a measurement of visitor attention. You can tell which content is better by the number of visitors who scroll to the bottom of an article. If all visitors stop scrolling at a certain point in a page, you know they've either found what they were after, or have tuned out.

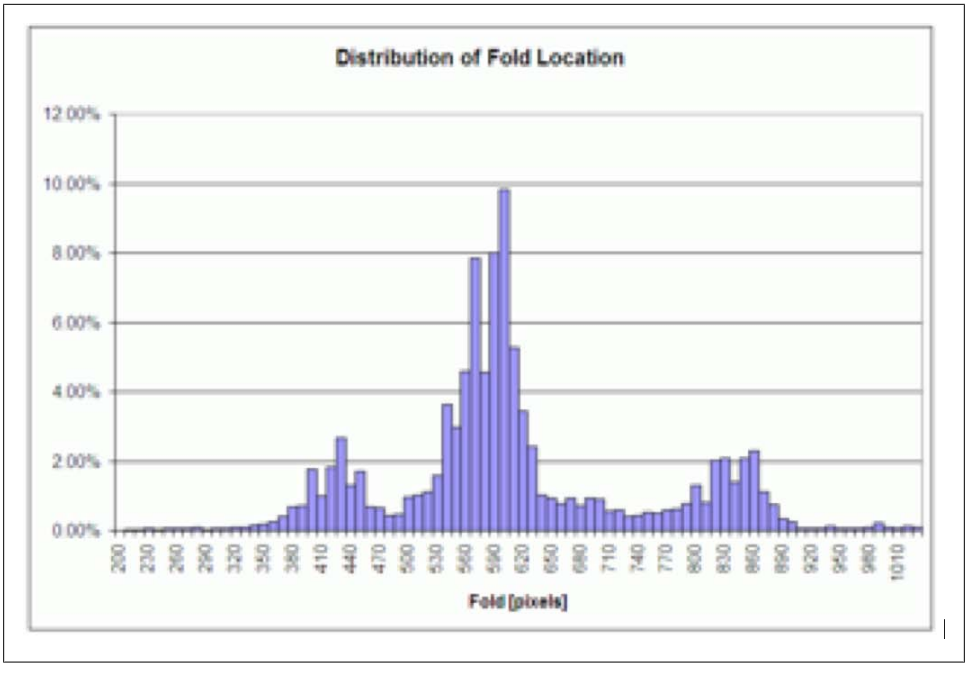


Figure 6-1. Height of the web page fold, in pixels, from the ClickTale study

Scrolling As a Metric of Visibility

In October 2007, WIA vendor ClickTale published a study of 80,000 page views collected over one month in 2007 (*http://blog.clicktale.com/2007/10/05/clicktale-scrolling-research-report-v20-part-1-visibility-and-scroll-reach/*). 91 percent of pages analyzed were long enough to contain a scrollbar—that is, when rendered, their heights in pixels exceeded that of the visitor's browser screen. Data below the visible page is considered "below the fold," and is less likely to be seen.

Scrolling depends a great deal on the typical height of a browser window. The fold varies from visitor to visitor based on the type of browser, any toolbars or plug-ins, and the desktop and window size. As Figure 6-1 shows, there are three clear peaks in the data from the study at 430, 600, and 860 pixels. These represent the most common browser and toolbar settings on the three most common screen heights at the time (800×600, 1024×768, and 1280×1024). Today's variety of monitor sizes and the growing popularity of mobile devices mean these results are changing significantly.

Regardless of the varying heights, a consistent number of visitors—22%—scrolled to the bottom of the page, as shown in Figure 6-2.

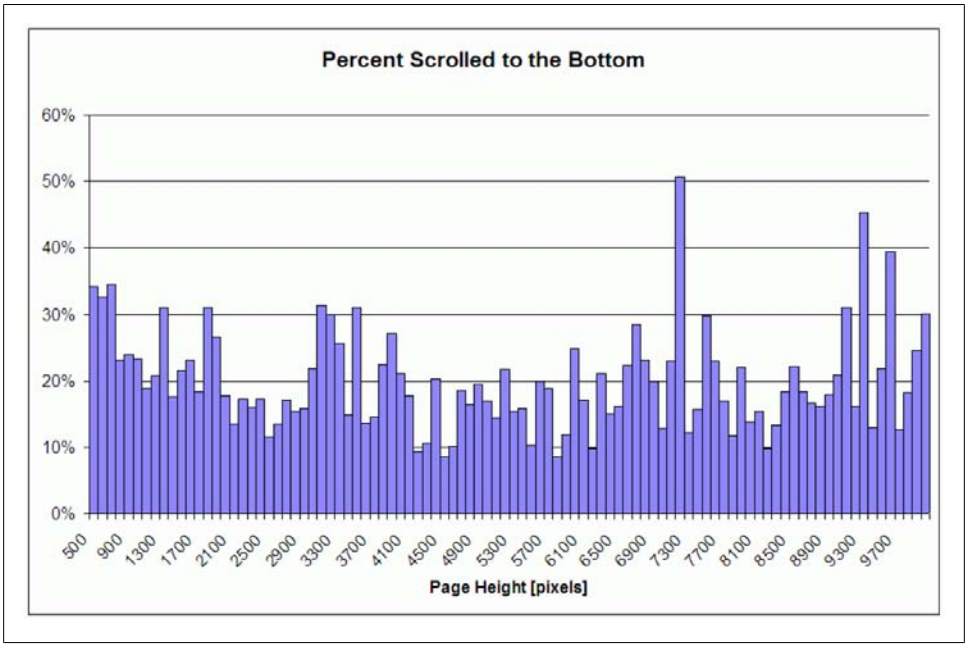


Figure 6-2. Percent of visits analyzed in the ClickTale study that made it to the bottom of the page

As the study says, "the same number of page viewers will tend to scroll halfway or three-quarters through a page, regardless of whether the page size is 5,000 pixels or 10,000 pixels" (*http://blog.clicktale.com/2007/12/04/clicktale-scrolling-research-report-v20-part-2-visitor-attention-and-web-page-exposure/*). Just to be clear: if you have content at the bottom of your page, 71% of visitors won't see it.

Paying attention

Tracking scrolling is only one measure of visitor attention. You also want to know how much time visitors spent on different parts of the page. Did they dwell on the uppermost portions of the page, or spend equal amounts of time throughout it?

To understand this, ClickTale plotted the number of seconds spent on various parts of the page for pages of different heights (shown in Figure 6-3).

The study showed that people spent most of their time in the top 800 pixels of a page regardless of its height, and that they spent slightly longer at the bottom of the page, possibly because they were interacting with navigational buttons or because they simply dragged the slider to the foot of the page.

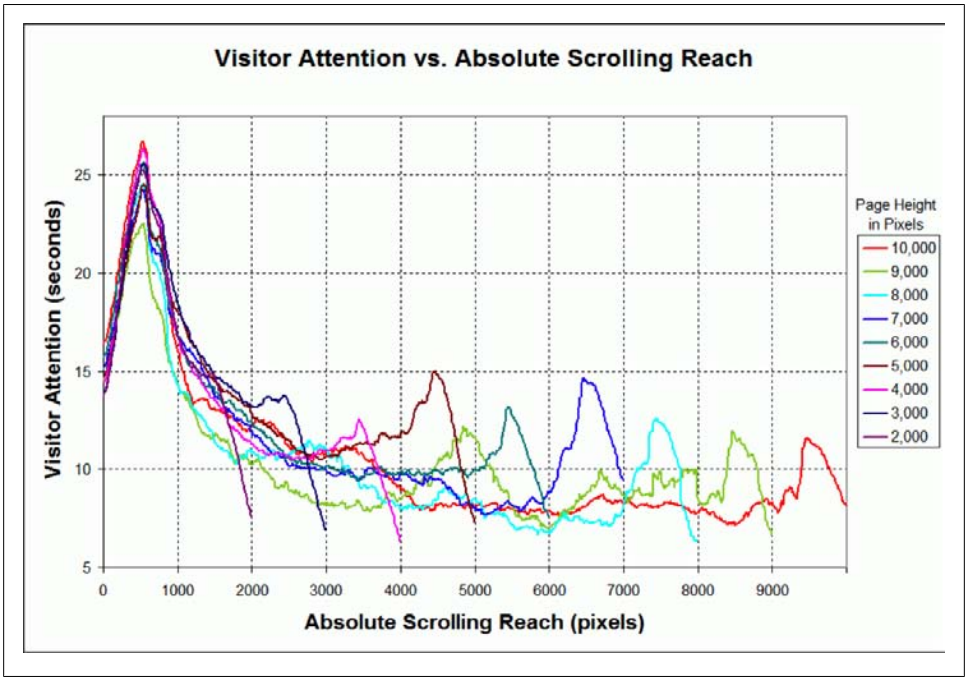


Figure 6-3. Number of seconds spent on parts of the page for various page heights

Your mileage will vary

This study doesn't segment scrolling behavior by page type. We would expect to see significantly different numbers if we were to analyze only content pages where visitors had a reason to review the entire page or only pages with a simple yes/no decision.

You'll also get very different results depending on whether the pages you're analyzing are part of a media site, a SaaS application, a transactional site, or a collaborative portal, and based on what your visitors are trying to accomplish.

Figure 6-4 shows an example of a report for a page, showing how many visitors scrolled how far down within the page.

Using this data, you'll know how far visitors are reading before going elsewhere. For media sites, this kind of WIA report will tell you how likely visitors are to see advertising embedded in the page. For collaborative sites, the report can show you how many visitors are making it to the comment threads below a post.

Proper Interactions: Click Heatmaps

On most websites, visitors have several ways to accomplish their goals. You may let a visitor click on a button or a text link that both link to the same page. Perhaps you have

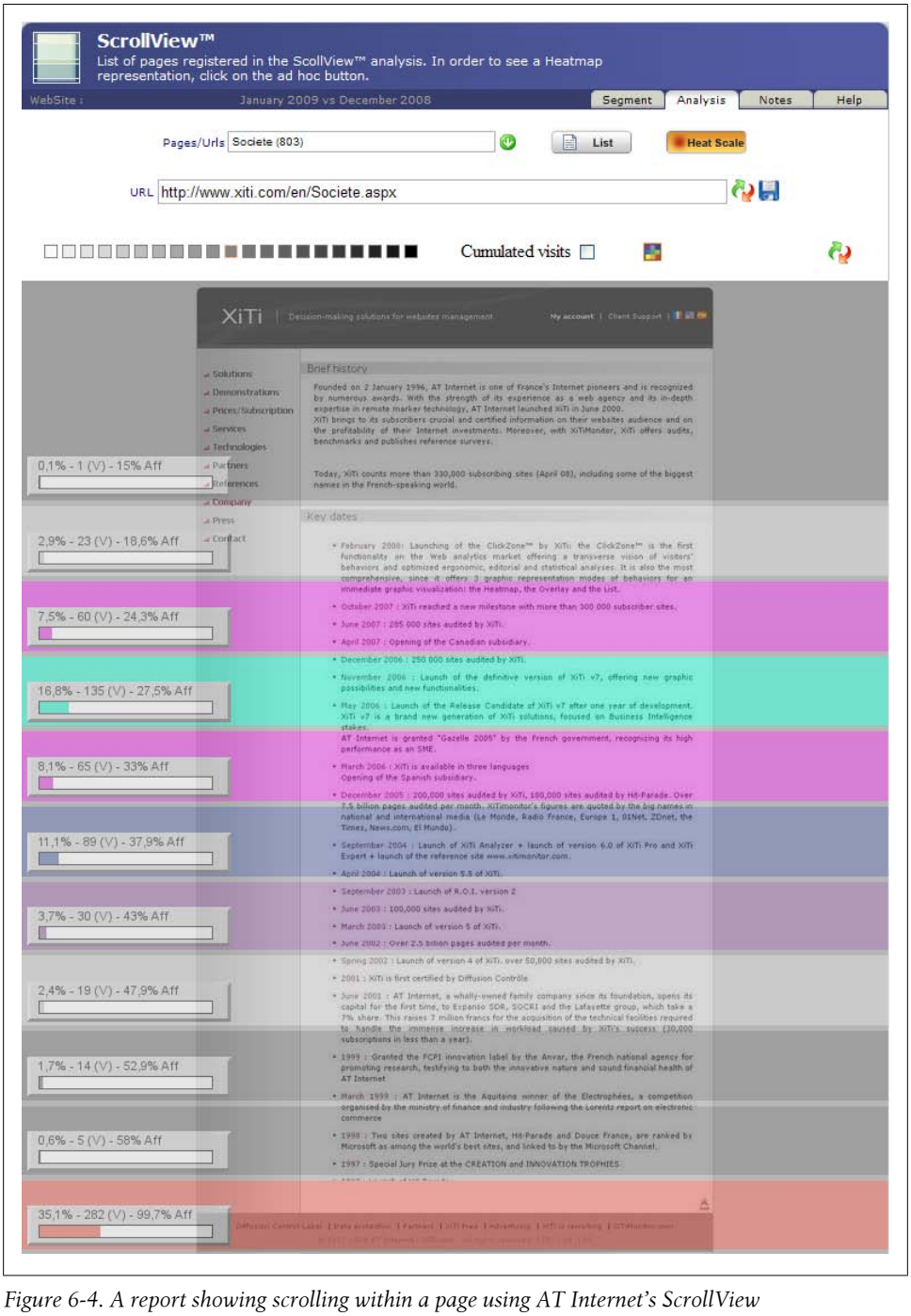


Figure 6-4. A report showing scrolling within a page using AT Internet's ScrollView

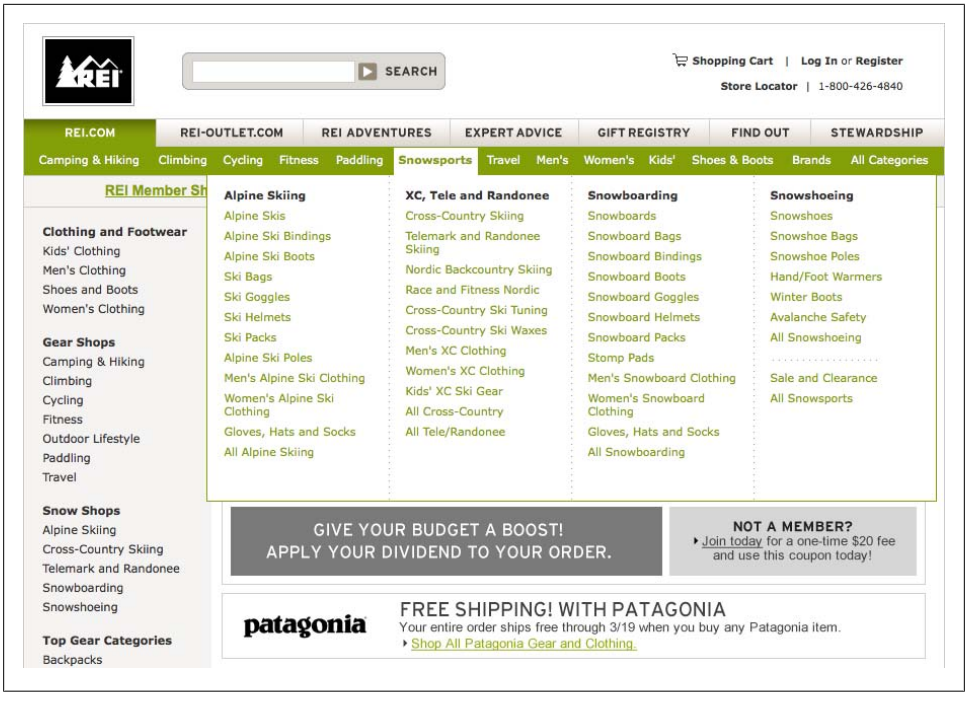


Figure 6-5. Visitors to REI.com have several ways to find a specific item, including a sidebar, specials, a top menu, and a search field

pagination controls at the top and bottom of a page. Or maybe you have top and sidebar menus. There's a tremendous amount of choice for visitors, as Figure 6-5 illustrates.

WIA tools can track the coordinates of each click, showing you what visitors clicked, and what they didn't. By segmenting these clicks, you can get a better understanding of which parts of your interface visitors prefer. You can also streamline the interface: if you discover that the majority of visitors rely on one particular navigational element, such as a top menu, you may be able to reclaim precious above-the-fold real estate by removing redundant or unused controls.

Usability and Affordance

Psychologist J.J. Gibson first coined the term "affordance" in 1977 to refer to things within one's environment that suggest courses of action. A door handle, for example, suggests that the door should be pulled—it "affords" pulling. More recently, the term has been applied to human interface design in general, and computer design in particular, with the fundamental belief that good design makes affordance obvious because it suggests the right actions to the visitor.

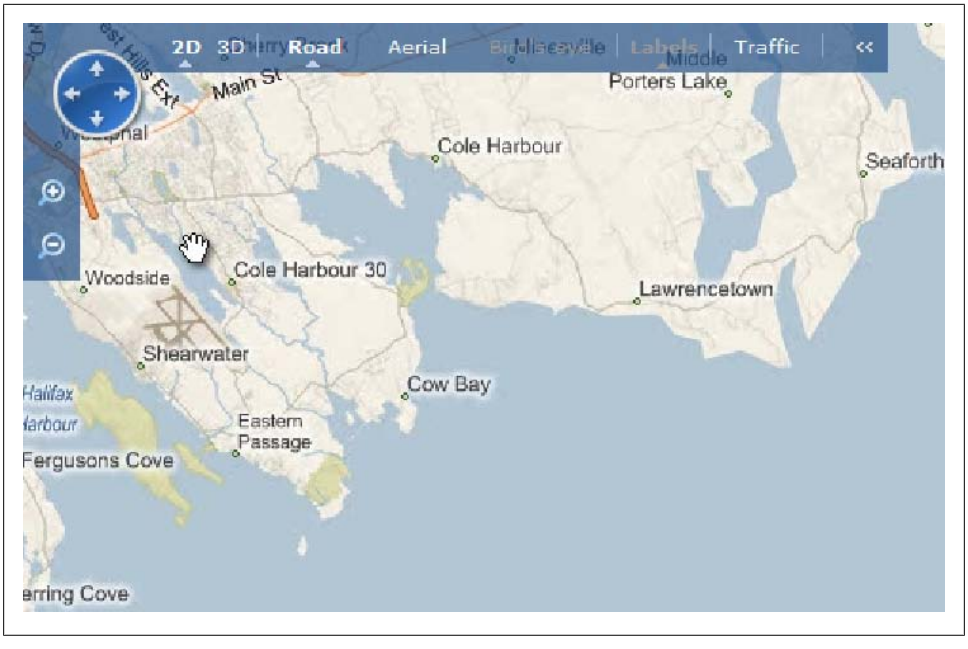


Figure 6-6. Microsoft's Live Maps changes the cursor to encourage certain visitor behaviors, such as dragging

Through years of web use, most consumers have become familiar with the basic building blocks of site interaction, such as text fields, scrollbars, and checkboxes. We recognize that we can expand a drop-down list or toggle radio buttons by clicking other nearby radio buttons. Put another way, a set of radio buttons *affords* us a choice.

On a website, affordances might include graphical elements that have underlying shadows, suggesting you can click them, or text boxes that appear recessed, suggesting they can contain information. More recently, we've added cursors and animations that reinforce a visitor's understanding of how to use the application, as shown in Figure 6-6.

When trying to understand whether a page is usable, we need to ask two questions. The first is whether an affordance exists: do we make it possible for visitors to do what they want to do? If we don't, it may be because the visitor is trying to do something we didn't envision in our design (which is a question better answered through VOC research). It may also be because the visitor doesn't understand the application, which means we need to explain how the site works.

The second question is one of perception: did the visitors notice the affordance we've created for them? Most developers find end visitor testing both frustrating and revealing because testers often can't see buttons that are right in front of their eyes.

Table 6-1 shows the four possible outcomes of these two questions. Problems with web interaction occur either because no affordance exists or because the visitor didn't perceive affordance.

Table 6-1. The four possible outcomes of interaction design

	Visitors perceive affordance	Visitors do not perceive affordance
Affordance exists	*Perceptible affordance* This is good design. The correct functions are present and visitors notice them.	*Hidden affordance* The function is present, but visitors don't notice it. It may be too small or hidden below the fold.
Affordance does not exist	*False affordance* Visitors believe that something is designed for interaction, but it's a red herring, such as an unclickable image that looks like a button.	*Correct rejection* This is good design: visitors are clear about what they can't do, so they're not interacting with the application in unanticipated ways.

In WIA, you will encounter both false affordance (in which your visitors are trying to do things you didn't intend them to) and hidden affordance (in which your visitors can't figure out how to do what you wanted them to).

In other words, we need to track the behavior of visitors, even when they're not using the application as intended, for example, when they click on a background they're not supposed to touch.

Analyzing Mouse Interactions

The most basic WIA report shows where visitors clicked, overlaid on the site, as shown in Figure 6-7.

Even these simple reports are revealing. You'll discover that visitors think graphical elements or backgrounds are actually clickable. Heatmaps like these will show you where visitors are clicking, even if they're not clicking on navigable items.

Segmenting clicks

If you're thinking like an analyst, you're probably eager to segment the data so you can act on it. One kind of segmentation is by click source. Segmented heatmaps color-code clicks according to visitor segments such as referring sites, keywords, browser types, and other information collected by the monitoring script or the analytics service, as shown in Figure 6-8. The segmentation is usually attached to page elements such as `div` tags, rather than just to links. Clicking a color-coded link shows more details about the volume of clicks the area received.

Segmentation of this kind shows you how your visitors' "baggage" affects their interactions. Visitors who come from a Google search, for example, may have different ways of interacting with your site than those who search on Yahoo!. Or users of a particular

Figure 6-7. A heatmap of clicks overlaid on a web page using XiTi Analyzer

browser may click in one place, while all others click elsewhere—a clue that the page is rendering differently in their browsers.

A good site is not only effective, allowing visitors to achieve goals without making mistakes or getting stuck, it's also efficient, letting them accomplish those tasks quickly. A wizard that helps first-time visitors complete a goal may frustrate "power users" seeking a faster way to achieve the same goals. Segmenting novices from seasoned users

Figure 6-8. Individual clicks segmented by referring URL in Crazy Egg

and examining their clicking behavior can reveal how more experienced visitors use the site differently.

In addition to segmenting by source, you should also segment by the goals you want visitors to achieve. Many analytics tools include click reports that show conversion rates by the elements of a page, as shown in Figure 6-9.

All of these aggregate views help you to understand what's most popular, which links don't lead to desired outcomes, and places where visitors are clicking mistakenly. In other words, they confirm or contest the design assumptions you made when you first built the website.

Data Input and Abandonment: Form Analysis

One of the most valuable, and least analyzed, components of a web application is its forms. A form is where your visitors provide the most detailed information about themselves. It's where they place orders, enter billing information, and leave messages. It's where the rubber meets the road. But companies spend far more time on site design than they do on analyzing form usability and abandonment.

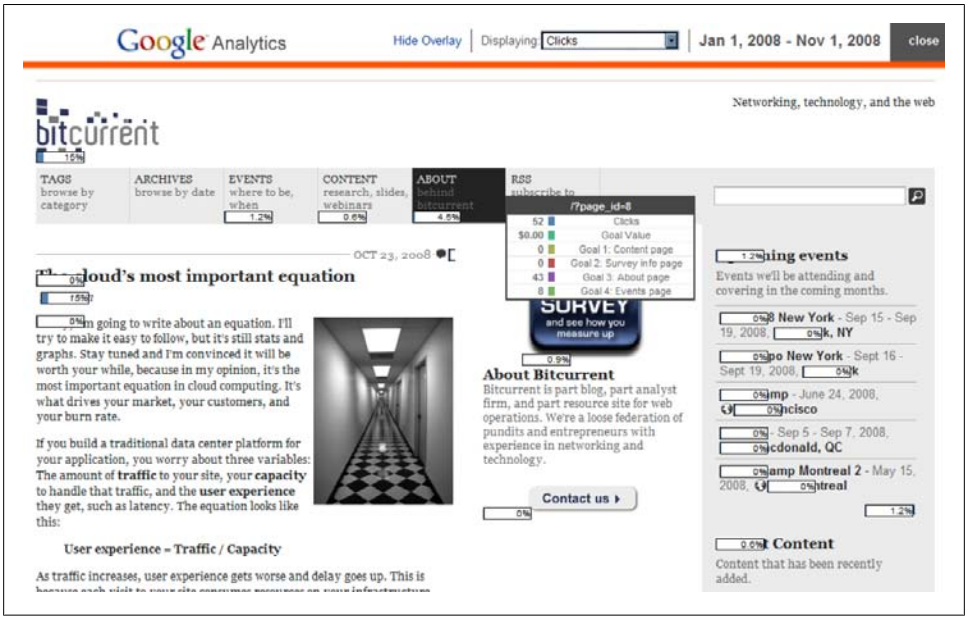


Figure 6-9. Aggregate clicks overlaid on a site, segmented by goal attainment, within Google Analytics

Each form represents a miniature conversion funnel with its own milestones, answering questions like:

- How many visitors saw the form correctly?
- How many visitors started to fill out the form?
- How far down the form did visitors get?
- How long did visitors linger at each field?
- What was the abandonment for each field?
- How many visitors submitted the form?
- How many visitors submitted the form, but missed required fields?
- How many optional fields did visitors skip?
- How many visitors successfully reached the following page?

Given how important form design is to web interaction, it's surprising how little work has been done on getting it right. Figure 6-10 shows an example of a funnel microconversion report.

One excellent resource on form design is Luke Wroblewski's *Web Form Design: Filling in the Blanks* (Rosenfeld Media) and the accompanying blog, "Functioning Form" (*http://www.lukew.com/ff/*).

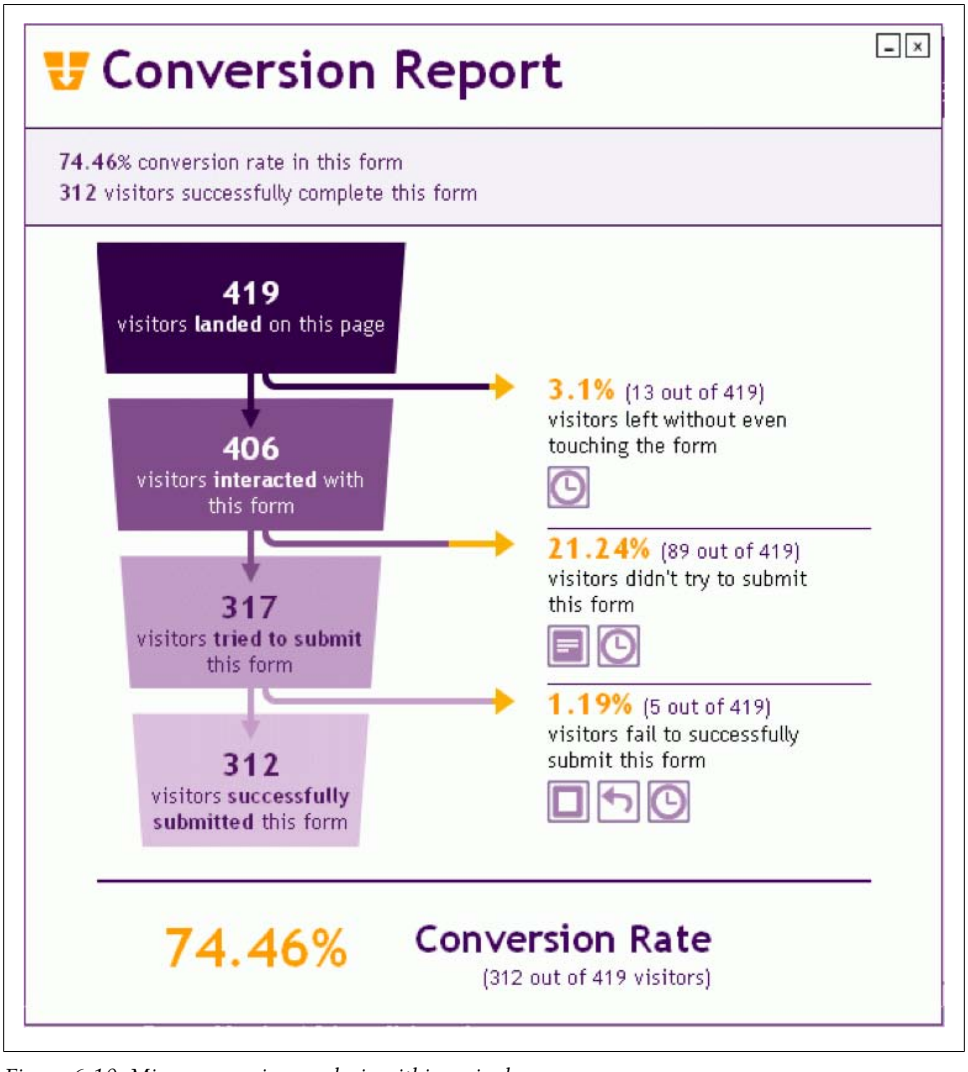


Figure 6-10. Microconversion analysis within a single page

We can peer deeper into the process than just abandonment, however. WIA lets us see how far in a form a visitor progressed, where she dropped out, and which fields she lingered on, as shown in Figure 6-11.

ClickTale has five reports related to form microconversion, described in the company's forum at *http://forum.clicktale.net/viewtopic.php?f=3&t=21&p=29&hilit=ssl+https - p29*.

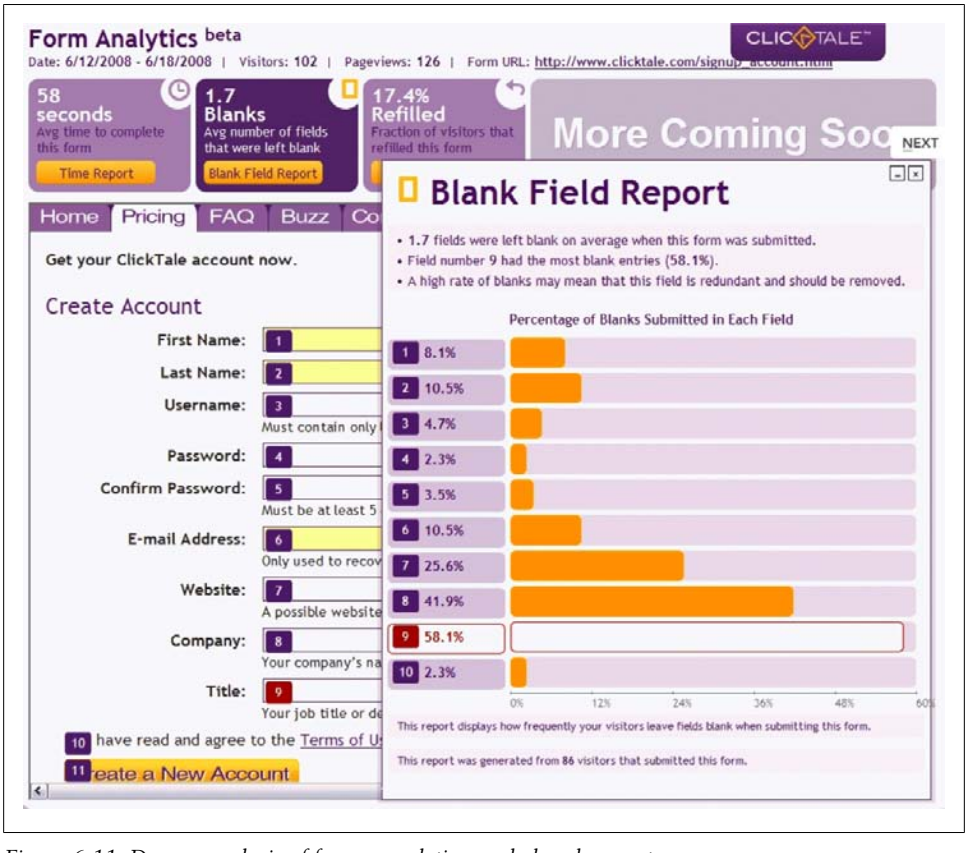


Figure 6-11. Deeper analysis of form completion and abandonment

Aggregation of in-page conversion is a relatively new field. While today's solutions are standalone offerings from ClickTale, FormAlive, and others, we expect to see this kind of analysis become an integral part of web analytics.

Individual Visits: Replay

We've looked at three kinds of analysis that can show whether your design assumptions are valid or wrong in the aggregate. There's another way to understand your visitors better: by watching them individually. Using either JavaScript or an inline device, you can record every page of a visit, then review what happened.

Replaying an individual visit can be informative, but it can also be a huge distraction. Before you look at a visit, you should have a particular question in mind, such as, "Why did the visitor act this way?" or "How did the visitor navigate the UI?"

Without a question in mind, watching visitors is just voyeurism. With the right question, and the ability to segment and search through visits, capturing entire visits and replaying pages as visitors interacted with them can be invaluable.

Stalking Efficiently: What You Replay Depends on the Problem You're Solving

Every replay of a visit should start with a question. Some important ones that replay can answer include:

Are my designs working for real visitors?
: This is perhaps the most basic reason for replaying a visit—to verify or disprove design decisions you've made. It's an extension of the usability testing done before a launch.

Why aren't conversions as good as they should be?
: By watching visitors who abandon a conversion process, you can see what it was about a particular step in the process that created problems.

Why is this visitor having issues?
: If you're trying to support a visitor through a helpdesk or call center, seeing the same screens that he is seeing makes it much easier to lend a hand, particularly when you can look at the pages that he saw before he picked up the phone and called you.

Why is this problem happening?
: Web applications are complex, and an error deep into a visit may have been caused by something the visitor did in a much earlier step. By replaying an entire visit, you can see the steps that caused the problem.

What steps are needed to test this part of the application?
: By capturing a real visit, you can provide test scripts to the testing or QA department. Once you've fixed a problem, the steps that caused it should become part of your regularly scheduled tests to make sure the problem doesn't creep back in.

How do I show visitors the problem is not my fault?
: Capturing and replaying visits makes it clear what happened, and why, so it's a great way to bring the blame game to a grinding halt. By replacing anecdotes with accountability, you prove what happened. Some replay solutions are even used as the basis for legal evidence.

We can answer each of these questions, but each requires a different approach.

Post-launch usability testing: Are my designs working?

It's always important to watch actual visitor behavior, no matter how much in-house usability testing you do. It's hard to find test users whose behavior actually matches

Figure 6-12. Search options in ClickTale

that of your target market, and even if you did, your results wouldn't be accurate simply because people behave differently when they know they're being watched.

For this kind of replay, you should start with the design in question. This may be the new interface you created or the page you've just changed. You'll probably want to query the WIA data for only visitors who visited the page you're interested in. While query setup varies from product to product, you'll likely be using filters like the ones shown in Figure 6-12.

Your query will result in a list of several visits that matched the criteria you provided, such as those shown in Figure 6-13. From this list, you can examine individual visits page by page.

The WIA solution will let you replay individual pages in the order the visitor saw them, as shown in Figure 6-14.

Depending on the solution, you may see additional information as part of the replayed visit:

- The WIA tool may show mouse movements as a part of the replay process.
- The visualization may include highlighting of UI elements with which the visitor interacted, such as buttons.
- The tool may allow you to switch between the rendered view of the page and the underlying page data (HTML, stylesheets, and so on).
- You may be able to interact with embedded elements, such as videos and JavaScript-based applications, or see how the visitor interacted with them.
- You may have access to performance information and errors that occurred as the page was loading.

These capabilities vary widely across WIA products, so you should decide whether you need them and dig into vendor capabilities when choosing the right WIA product.

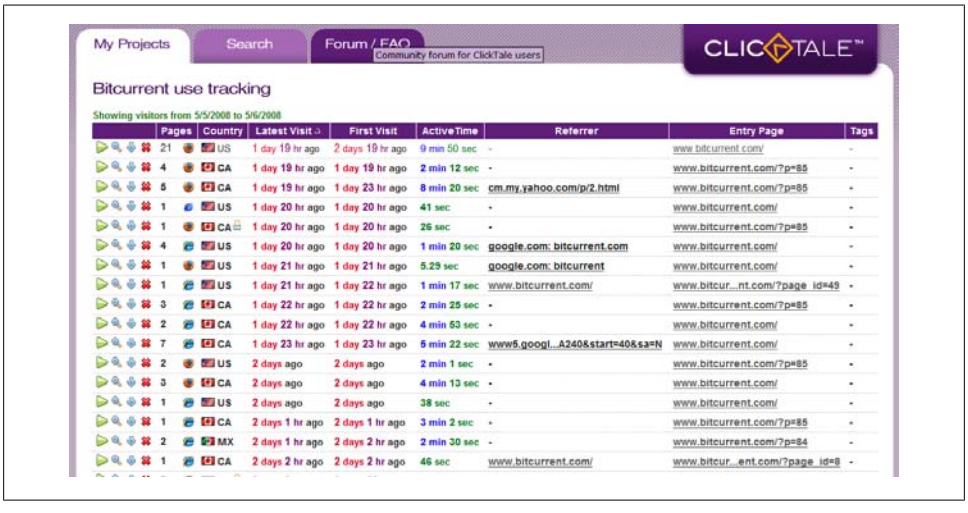


Figure 6-13. Browsing a list of recorded visits in ClickTale

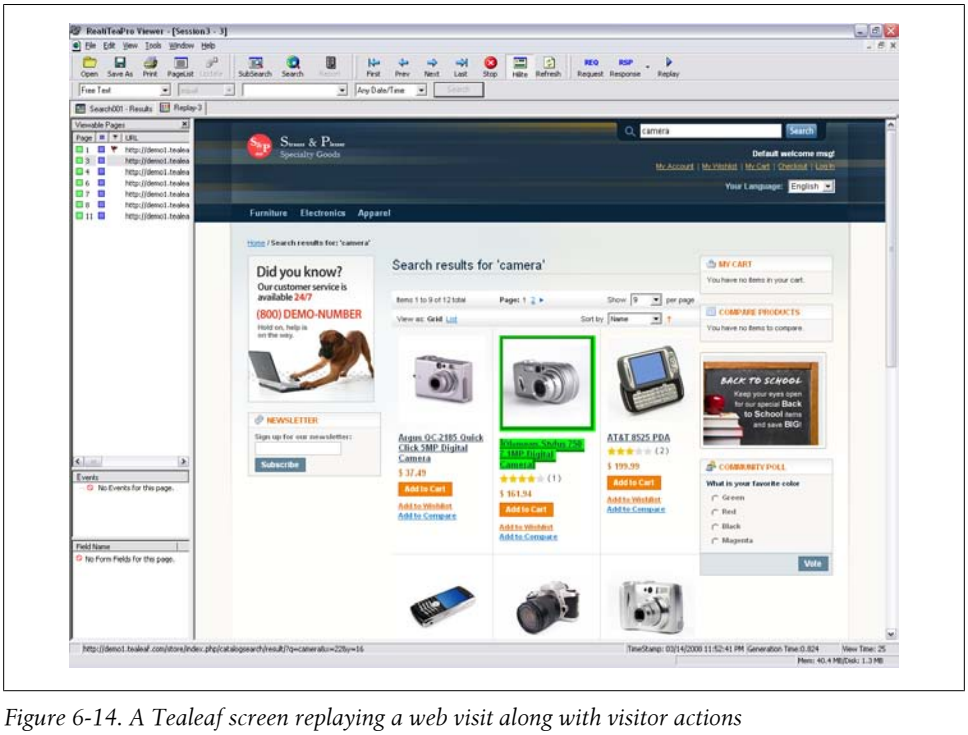


Figure 6-14. A Tealeaf screen replaying a web visit along with visitor actions

Conversion optimization: why aren't conversions as good as they should be?

When you launch a website, you expect visitors to follow certain paths through the site and ultimately to achieve various goals. If they aren't accomplishing those goals in sufficient numbers, you need to find out why.

Lots of things cause poor conversion. Bad offers or lousy messages are the big culprits, and you can address these kinds of issues with web analytics and split testing. Sometimes, however, application design is the problem.

Your conversion funnels will show you where in the conversion process visitors are abandoning things. Then it's a matter of finding visitors who started toward a goal and abandoned the process.

You should already be monitoring key steps in the conversion process to ensure visitors are seeing your content, clicking in the right places, and completing forms, as discussed earlier. To really understand what's going on, though, you should view visits that got stuck.

The Aha Moment: No Foreigners Allowed

A very large U.S.-based computer goods retailer started to receive orders from Canada. While American postal addresses contain a five-number zip code (for example, 01234), Canadian addresses have a six-character postal code (for example, H0H 0H0). The retailer was receiving orders from Canadians, but didn't know it because form field validation on the checkout page was refusing to accept the longer postal codes.

No amount of web analytics, synthetic testing, or real user monitoring (RUM) revealed the issue.

- Web analytics simply showed that visitors were abandoning the process, but since this had been happening for a long time, the company didn't notice an increase. Furthermore, the company didn't anticipate Canadian orders, so it was segmenting by U.S. city and not by country.
- Synthetic tests would have tried a five-number zip code, which would have worked.
- RUM data wouldn't have seen the form rejection on the wire, because form validation was rejecting the contents before they could be submitted via a POST.

By tagging visits that had a rejected form validation, the company identified a pattern of failed transactions that was strongly correlated with Canadian visitors. By replaying the contents of the rejected forms, the company quickly realized what was happening.

Sometimes, problems detected as part of usability monitoring stem from broader concerns—in this case, internationalization and an understanding of target markets.

Figure 6-15. Conversion funnel segments in Omniture's SiteCatalyst

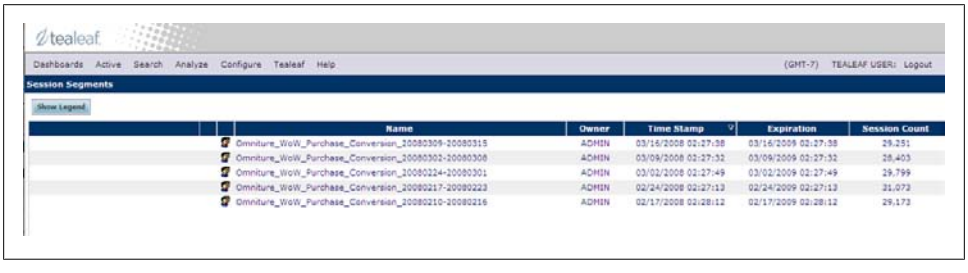


Figure 6-16. A list of captured visits in Tealeaf corresponding to the conversion segments in Figure 6-15; in this case, they're for World of Warcraft purchases

The best place to start is a conversion funnel. For example, in Figure 6-15 a segment of 29,251 visits reached the conversion stage of a sales funnel for an online store.

Let's say you're selling a *World of Warcraft* MMORPG upgrade, and you have a high rate of abandonment on the *lichking.php* page. You need to identify visits that made it to this stage in the process but didn't continue to the following step. There are three basic approaches to this:

The hard way
: Find all visits that included *lichking.php*, then go through them one by one until you find a visit that ended there.

The easier way
: Use a WIA tool that tags visits with metadata such as "successful conversion" when they're captured. Query for all pages that included *lichking.php* but did not convert.

The easiest way
: If your WIA solution is integrated with an analytics tool, you may be able to jump straight from the analytics funnel into the captured visits represented by that segment, as shown in Figure 6-16.

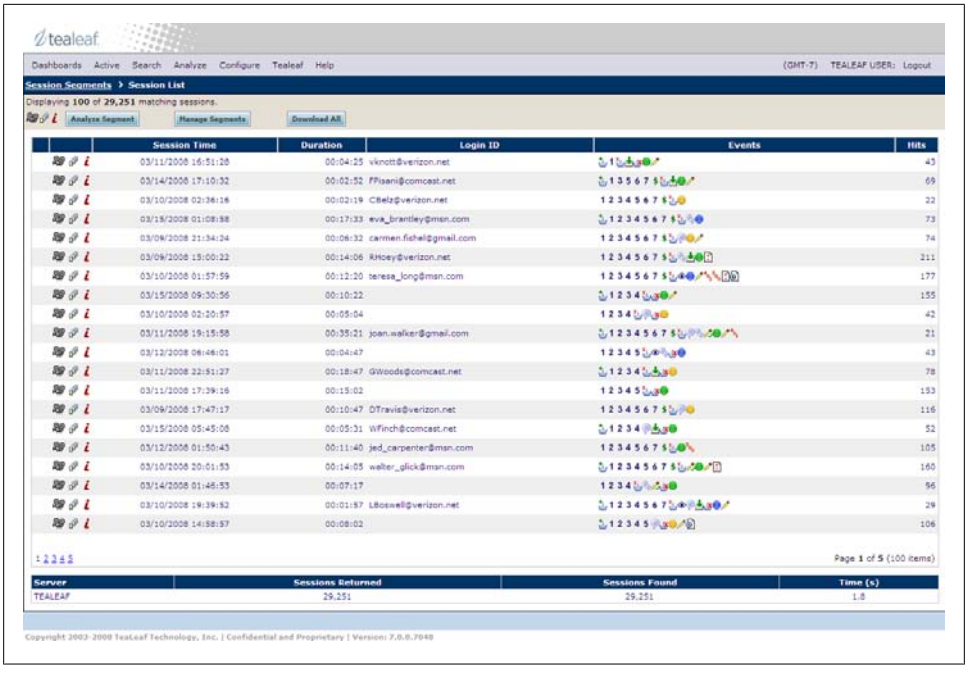


Figure 6-17. A display of captured visits, tagged with specific events or attributes, in Tealeaf

From this segment, you can then select an individual visit to review from a list (such as the one shown in Figure 6-17) and understand what happened.

One final note on conversion optimizations: when you're trying to improve conversions, don't just obsess over problem visits and abandonment. Look for patterns of success to learn what your "power users" do and to try and understand what works.

Helpdesk support: Why is this visitor having issues?

You can't help someone if you don't know what the problem is. When tied into call centers and helpdesks, WIA gives you a much better understanding of what transpired from the visitor's perspective. Visitors aren't necessarily experiencing errors. They may simply not understand the application, and may be typing information into the wrong fields. This is particularly true for consumer applications where the visitor has little training, may be new to the site, and may not be technically sophisticated.

If this is how you plan to use WIA, you'll need to extract information from each visit that helps you to uniquely identify each visitor. For example, you might index visits by the visitor's name, account number, the amount of money spent, the number of comments made, how many reports he generated, and so on.

You'll also want to tie the WIA system into whatever Customer Relationship Management (CRM) solution you're using so that helpdesk personnel have both the visitor's

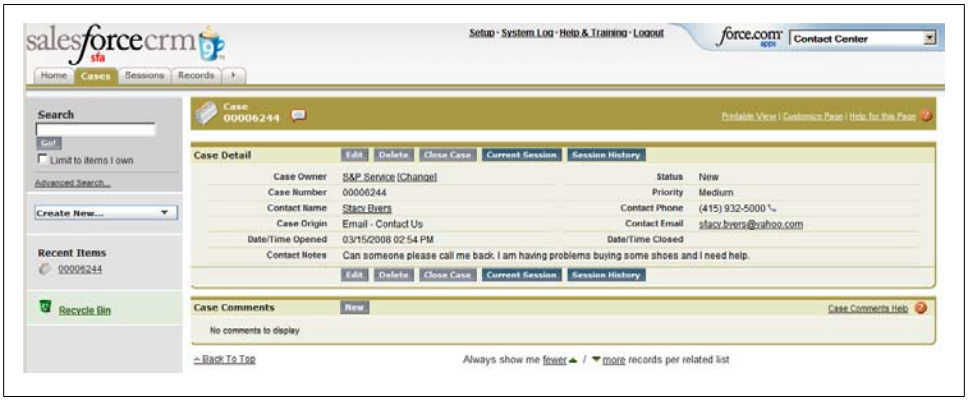


Figure 6-18. Tealeaf session history integrated inside a Salesforce.com record

experience and account history at their fingertips. Figure 6-18 shows an example of Tealeaf's replay integrated into a Salesforce.com CRM application.

The helpdesk staff member receives a trouble ticket that includes a link to her current session. This means she's able to contact the suffering customer with the context of the visit.

Incident diagnosis: Why is this problem happening?

While WIA for helpdesks can assist visitors with the application, sometimes it's not the visitor's fault at all, but rather an error with the application that you weren't able to anticipate or detect through monitoring. When this happens, you need to see the problem firsthand so you can diagnose it.

While you'll hear about many of these problems from the helpdesk, you can also find them yourself if you know where to look.

- When a form is rejected as incomplete or incorrect, make sure your WIA solution marks the page and the visit as one in which there were form problems. You may find that visitors are consistently entering data in the wrong field due to tab order or page layout.
- When a page is slow or has errors, check to see if visitors are unable to interact with the page properly. Your visitors may be trying to use a page that's still missing some of its buttons and formatting, leading to errors and frustration.
- If there's an expected sequence of events in a typical visit, you may be able to flag visits that don't follow that pattern of behavior, so that you can review them. For example, if you expect visitors to log in, then see a welcome page, any visits where that sequence of events didn't happen is worthy of your attention.

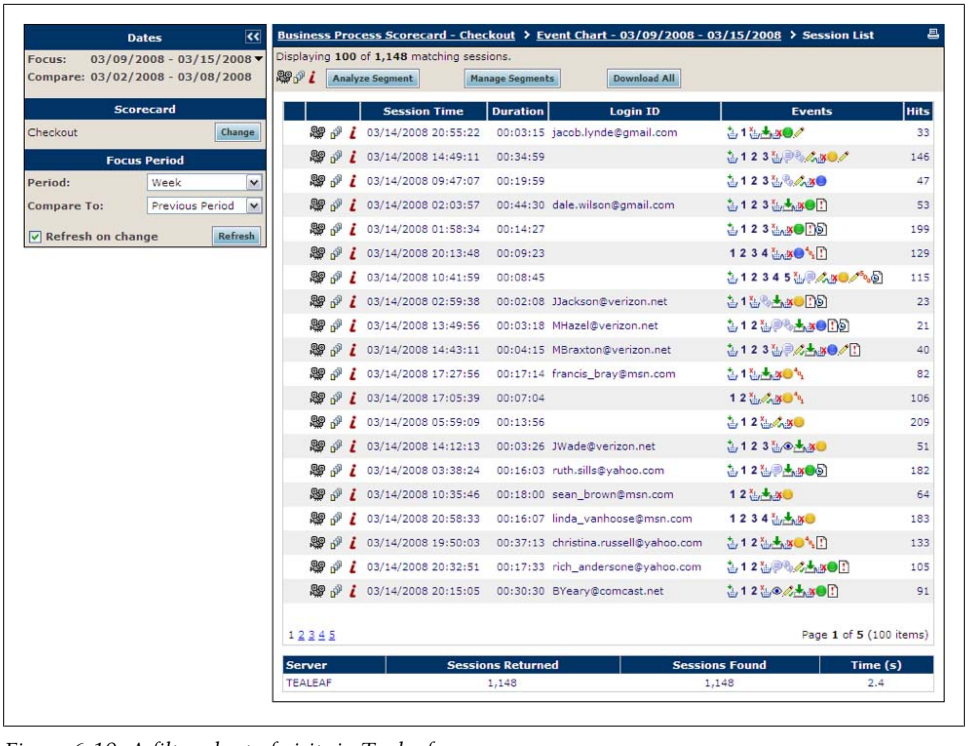


Figure 6-19. A filtered set of visits in Tealeaf

Once you've filtered the visits down to those in which the event occurred, browsing through them becomes much more manageable, as shown in Figure 6-19.

In many WIA tools, the list of visits will be decorated with icons (representing visitors who spent money, enrolled, or departed suddenly, for example) so that operators can quickly identify them. Decorating visits in this way is something we'll cover when we look at implementing WIA.

Test case creation: What steps are needed to test the app?

The teams that test web applications are constantly looking for use cases. Actual visits are a rich source of test case material, and you may be able to export the HTTP transactions that make up a visit so that testers can reuse those scripts for future testing.

This is particularly useful if you have a copy of a visit that broke something. It's a good idea to add such visits to the suite of regression tests that test teams run every time they release a new version of the application, to ensure that old problems haven't crept back in. Remember, however, that any scripts you generate need to be "neutered" so that they don't trigger actual transactions or contain personal information about the visitor whose visit was captured.

Dispute resolution: How do I prove it's not my fault?

There's no substitute for seeing what happened, particularly when you're trying to convince a department that the problem is its responsibility. Replacing anecdotes with facts is the fastest way to resolve a dispute, and WIA gives you the proof you need to back up your claims.

Dispute resolution may be more than just proving you're right. In 2006, the U.S. government expanded the rules for information discovery in civil litigation (*http://www.uscourts.gov/rules/congress0406.html*), making it essential for companies in some industries to maintain historical data, and also to delete it in a timely fashion when allowed to do so. (WIA data that is stored for dispute resolution may carry with it certain storage restrictions, and you may have to digitally sign the data or take other steps to prevent it from being modified by third parties once captured).

Retroactive Segmentation: Answering "What If?"

As a web analyst, you'll often want to ask "what if" questions. You'll think of new segments along which you'd like to analyze conversion and engagement, for example, "Did people who searched the blog before they posted a comment eventually sign up for the newsletter?"

To do this with web analytics, you'd need to reconfigure your analytics tool to track the new segment, then wait for more visits to arrive. Reconfiguration would involve creating a page tag for the search page and for the comment page. Finally, once you had a decent sample of visits, you'd analyze conversions according to the newly generated segments. This would be time-consuming, and with many analytics products only providing daily reports, it would take several days to get it right. The problem, in a nutshell, is that web analytics tools can't look backward in time at things you didn't tell them to segment.

On the other hand, if you've already captured every visit in order to replay it, you may be able to mine the captured visits to do *retroactive segmentation*—to slice up visits based on their attributes and analyze them in ways you didn't anticipate. Because you have all of the HTML, you can ask questions about page content. You can effectively create a segment from the recorded visits. You can mark all visits in which the text "Search Results" appears, and all the visits in which the text "Thanks for your comment!" appears—and now you have two new segments.

This is particularly useful when evaluating the impact of errors on conversions. If visitors get a page saying, "ODBC Error 1234," for example, performing a search for that string across all visits will show you how long the issue has been happening, and perhaps provide clues about what visitors did to cause the problem before seeing that page. You will not only have a better chance of fixing things, you will also know how common the problem is.

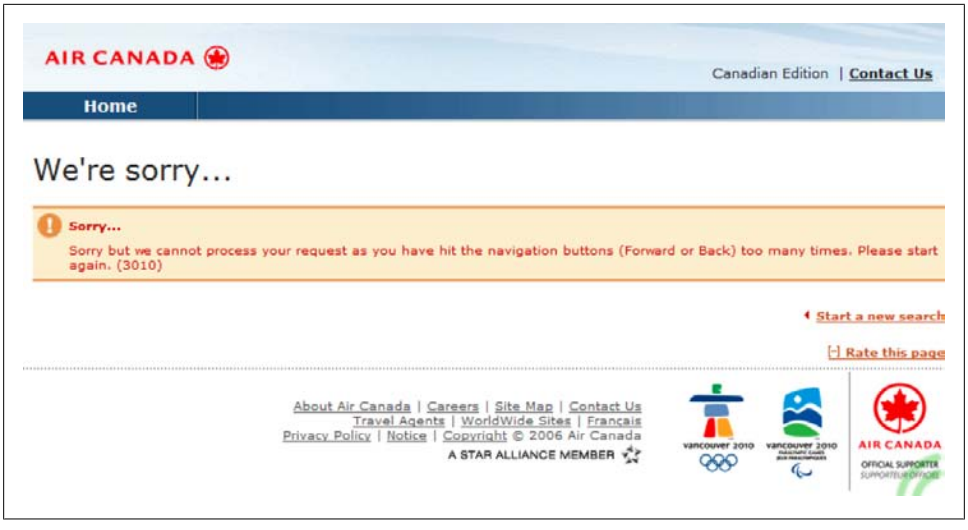


Figure 6-20. Something as simple as clicking the Back button on a travel site can break a visit

Consider the example in Figure 6-20. In this case, the visitor clicked the Back button once, entered new data, and clicked Submit.

The application broke. As an analyst, you want to know how often this happens to visitors, and how it affects conversion. With an analytics tool, you'd create a custom page tag for this message, wait a while to collect some new visits that had this tag in them, then segment conversions along that dimension.

By searching for the string "you have hit the navigation buttons," you're effectively constructing a new segment. You can then compare this segment to normal behavior and see how it differs. Because you're creating that segment from stored visits, there's no need to wait for new visits to come in. Figure 6-21 shows a comparison of two segments of captured visits.

Open text search like this consumes a great deal of storage, and carries with it privacy concerns, but it's valuable because we seldom know beforehand that issues like this are going to happen.

Implementing WIA

Now that you know which questions you want to answer with WIA, it's time to implement it. Which tools you use will depend on what kinds of interaction you want to watch, and how much you need to drill down into individual visits.

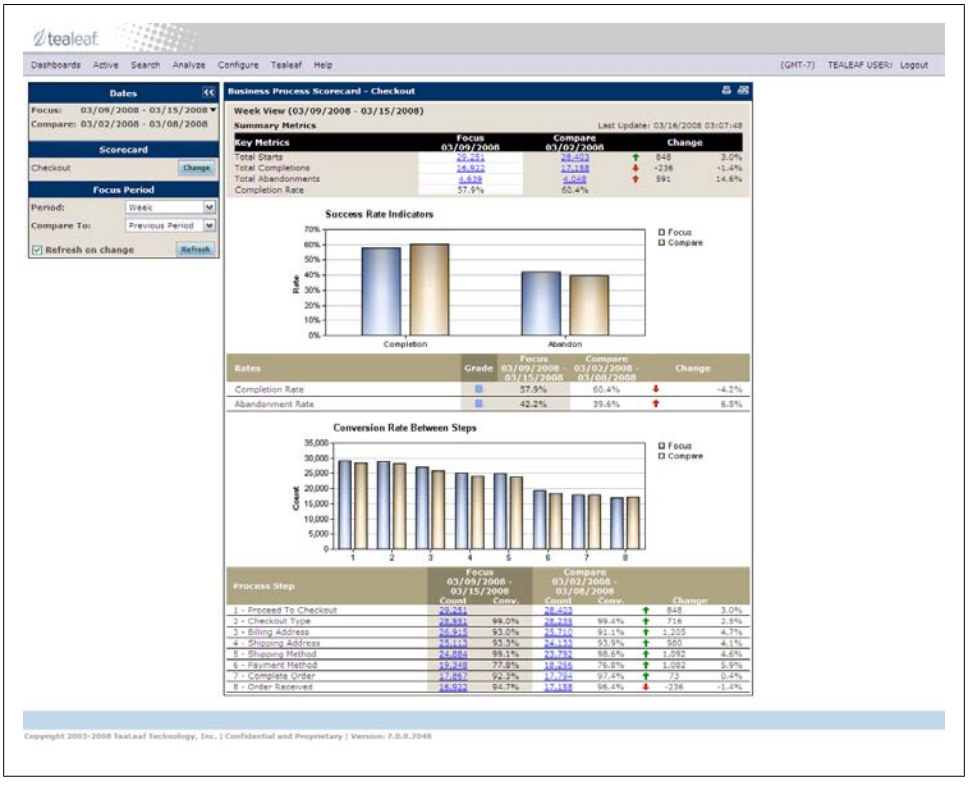


Figure 6-21. Comparing conversion rates across two segments in Tealeaf

Knowing Your Design Assumptions

Your first step is to map out your application and identify all of the assumptions you've made. There will be lots of them, so focus on these three:

Those that drive your business
: These may include steps in a funnel, an invitation cycle, publishing content, or making a purchase. This is where you can maximize the effectiveness of your site and optimize conversions.

Those that drive support and helpdesk calls
: Which pages make subscribers call you? Which pages give visitors the most ways to do the same things? Where do people usually get stuck? This is how you'll address usability and find problems you haven't identified.

Those that drive productivity

: If there's an action, such as upvoting a topic, adding a comment, updating a record, or browsing through a catalog, in which visitors spend most of their time, analyze it. This is your best chance to improve the efficiency of the application.

Once you've chosen the parts of your site to instrument, start collecting a baseline. Soon, you'll know what "normal" is. You'll have a pattern of clicks, or know which parts of a form make people leave, or know how far visitors scroll, and you can start testing variants on those forms much as you would for page analytics.

Deciding What to Capture

WIA collection happens through client-side JavaScript, server-side instrumentation, or a combination of the two.

If you want to capture things that happen on a browser, like scrolling, partial form completion, and clicks, you'll need to insert JavaScript snippets into the pages you want to monitor. Be sure to baseline website performance before and after inserting these scripts so you know what kind of performance price you're paying for this visibility.

To capture actual visits, you can either use server-side collection (eavesdropping on the conversation between the web server and the browser) or JavaScript. Server-side collection is more complete and provides greater forensic detail, but also costs more. Client-side collection is easier to implement, but is limited in what it can easily record about visits.

If you're planning to review individual visits, be sure you can flip between aggregate and individual views of visitor interactions. The usefulness of a WIA solution depends considerably on your ability to query for specific visits and to analyze across a segment of visits to find patterns within them.

Instrumenting and Collecting Interactions

You capture interactions in much the same way you collect other analytics data: a script on the page records information, such as form completion, click location, or scrolling, and sends it to the service along with something to identify the unique visit. The service does the rest, rendering visualizations and reports. Your work will involve putting the scripts on pages and adjusting those scripts to define what is captured. You'll define the start and end of an "experiment" period to coincide with changes to design and layout.

When it comes to recording entire visits, however, more work is required. Capture and replay of individual visits happens in one of two ways, depending on the WIA solution.

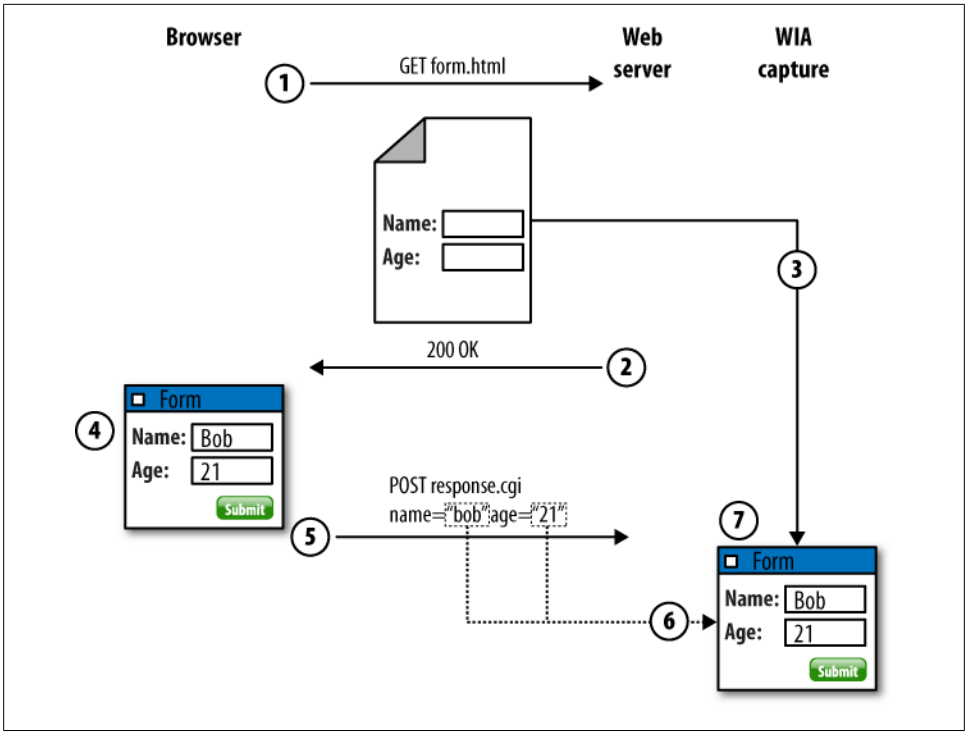


Figure 6-22. How inline capture works in solutions such as Tealeaf

Inline WIA

The first approach to visit capture, employed by WIA products that rely on inline capture of requests and responses, is illustrated in Figure 6-22.

Here's what happens:

1. The browser requests a page from the server.
2. The server responds with the page.
3. The WIA solution captures the HTML of the response, as well as headers and other information such as timing.
4. If the page requires user interaction (such as a form), the visitor carries out the required action.
5. The interaction (in this case, a completed form) is sent back to the server.
6. The WIA solution captures the response (in this case, what the visitor typed into the form).
7. To replay the visit, the system reassembles the original page (*form.html*) and the user input (from *response.cgi*) into a single display.

Capturing user interactions, such as mouse movements, requires additional JavaScript on the client, which is then captured by the WIA solution in the form of small requests similar to those used in an analytics solution. You don't *need* these scripts to see what visitors entered in a form, but if you're concerned about in-page actions or mouse movements, you may need them. If your WIA solution is trying to capture page load time from clients, JavaScript may also be able to augment performance measurements that the inline device collected.

Client-side WIA

The second approach, used by WIA solutions that collect visitor interactions from JavaScript, takes a different approach. The JavaScript on the page tells the service when the visitor has retrieved a page, at which point the service can get its own copy. The JavaScript then records a variety of details about the visit, which may include scrolling behavior, mouse movements, form completions, and clicks (Figure 6-23).

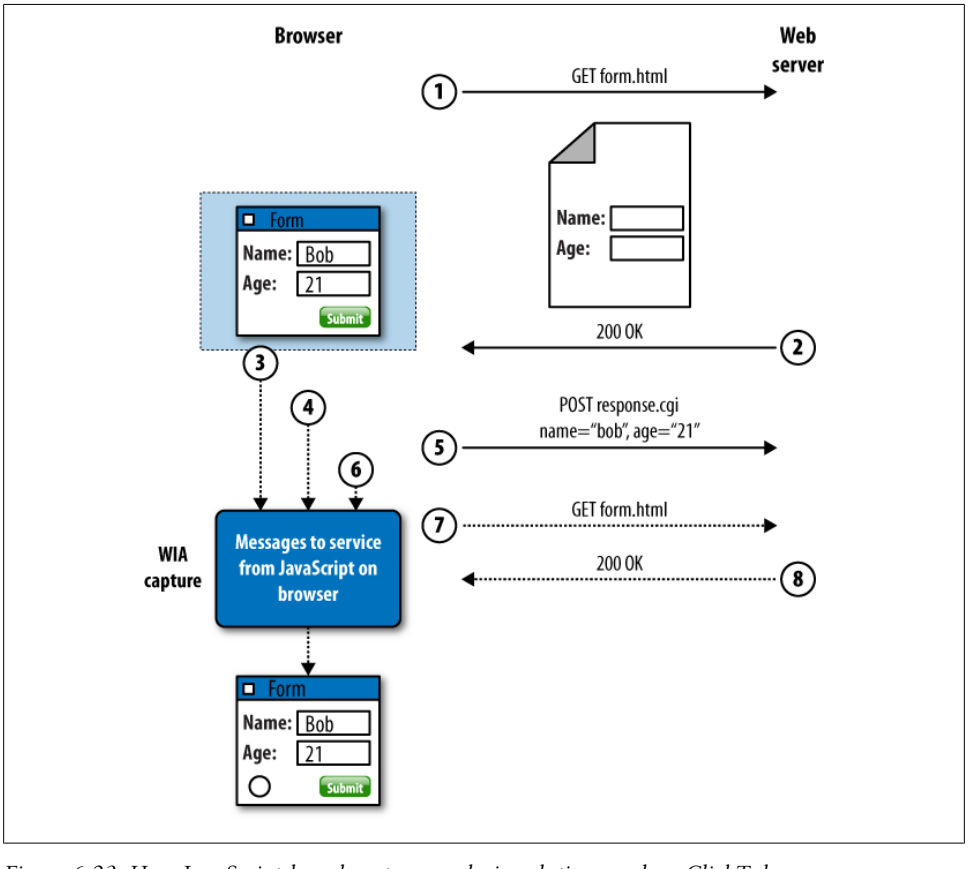


Figure 6-23. How JavaScript-based capture works in solutions such as ClickTale

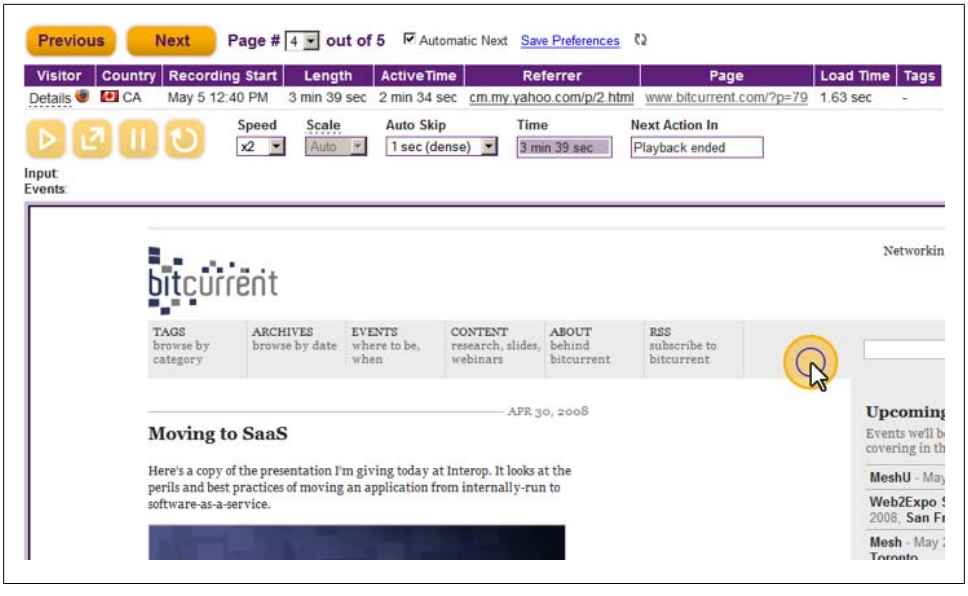


Figure 6-24. Replaying mouse movements in ClickTale

Because the browser is communicating with both the website and the WIA capture service, the steps are somewhat more complex:

1. The browser requests a page from the website.
2. The web server delivers the page.
3. JavaScript on the page tells the WIA capture service that a page (in this case, form.html) has been requested.
4. The visitor moves her mouse, scrolls, and clicks, all of which may be recorded by the JavaScript on the page and sent to the WIA capture service.
5. The visitor completes her interaction with the website, which may include the transmission of data (in this example, form information as part of an HTTP POST).
6. The JavaScript on the page passes this data back to the WIA capture service.
7. During this time, the WIA service retrieves a second copy of the website and stores it.
8. The web server returns a copy of the same page the visitor received.
9. When an analyst wants to review the replayed page, the WIA service shows its copy of the page, mouse movements, and visitor data.

If you're using client-side scripts to capture mouse movements and keystrokes, you can use playback controls to see where the visitor moved and what she typed, as shown in Figure 6-24.

When replaying WIA recordings that include mouse and keystroke data, you'll see the mouse moving around the page, as well as any scrolling the visitor performed. Note, however, that you may not be able to replay actions when the visitor interacted with some site elements. When a browser component had control of the mouse and keyboard, for example, you may not be able to see what the visitor was doing.

Service versus software

Hosted services simply can't give you the complete visibility of inline or server-side equipment, but they're much easier to deploy (and usually cheaper). It's up to you to consider cost, data privacy, and the need for visibility. If you want to use a service, you probably won't be able to capture every component of every visit, so you'll be using the tool more for behavioral analysis and less for diagnostics, retroactive segmentation, and evidence gathering.

Sampling and completeness

Capturing all these visits takes computing power, bandwidth, and storage. Most services limit the number of visits or clicks that they will store. Server-side WIA tools are limited by storage capacity, but all that traffic consumes a lot of storage.

To reduce the amount of data they need to store, these solutions can compress HTML files. They can also store just one copy of frequently requested content, such as stylesheets, JavaScript, and images.

To do this, a tool will look at each HTTP request to see if it has already stored each component on the page. If so, there's no need to store it again. On the other hand, if the requests are for a new object (such as an updated image or a new stylesheet), the system stores a copy. When it's time to assemble the page for replay, the tool simply uses the object that was current at the time the visit occurred.

Filtering and content reduction

Once you deploy a WIA solution, you'll quickly realize that you need to filter some of what's collected, either to reduce the total volume of data or to comply with security and privacy standards. Filters tell the system what to exclude and what to keep.

Often, you'll want to exclude a specific group from collection. You may, for example, only want to record new visitors to the site, or those who come from a Google organic search. Most WIA services allow you to customize collection rules using JavaScript within the pages themselves. For example, you could parse the DOM of the browser and examine the referring URL or check for the presence of a returning visitor cookie. Or you could configure inline server-side tools to check the contents of a page before deciding whether to store it, with similar effect.

Another common form of filtering is to limit the analysis to only a part of the site, such as a checkout process, or only to visits in which an event occurs. Again, using JavaScript,

Figure 6-25. A ClickTale report segmenting occurrences by tags

you can parse for certain conditions on the client. Or you can simply put the collection scripts only on the parts of the site that you want to monitor, which will speed up the time it takes unmonitored pages to load.

Augmenting visits with extraction and tagging

As we've seen, being able to query visits to find the interesting ones is important when replaying visits. Many attributes of a visit, such as browser type, IP address, or time that the visit happened, are automatically collected when the WIA tool records the visit.

Replay tools become far more useful when you add business context to them. Tagging a visit with the visitor's account number means the helpdesk can find it quickly. Flagging visitors who made it to the checkout process or who wound up paying for a product allows you to segment by those dimensions later.

Some client-side WIA tools let you add custom tags to the data they collect. For example, in ClickTale the following HTML marks the page with the `LoginForm Submitted` tag and appends the user's name:

```
<form ... onsubmit="ClickTaleTag("LoginFormSubmitted:" + UserName); ..." ... >
```

Later, you can run an analysis of those tags across captured visits, as shown in Figure 6-25. You can also use the tags as the start of a query for specific visits you want to replay.

You should always tag error conditions, such as a form that the visitor didn't complete correctly or a warning about navigation. That way, when you're analyzing the WIA data you can quickly find visits that encountered those problems and see what caused visitors to get stuck.

For inline, server-side collection, the collecting device identifies information within the HTML (such as the title tag of a page or a visitor's account number) and stores it as metadata about the page. Similarly, you can use certain URLs (such as *thanks.jsp*) to flag a milestone in a visit, such as payment completion. All of this makes captured pages easier to segment and recognize during analysis.

Tying WIA to other sources

If you've found a visit that's interesting, you probably want to share it with others. One way to do this is to forward a link to the WIA system, assuming other members of your team have access. You may also be able to export the data as a video file; a series of screenshots; a test script suitable for QA; a list of page, object, and error records that developers can analyze; or even a raw packet trace that can be examined in a sniffer.

Issues and Concerns

WIA lets us peer deep within the pages of a visit to get a feel for how visitors interacted with components below the page level. It's a technology that must be used judiciously and in the context of the rest of your web monitoring for its results to be meaningful. It also has some important limitations.

What if the Page Changes?

When analyzing visitor interactions, remember that measurements are only valid for a particular page design. You're trying to test a particular page layout to evaluate scrolling, click locations, and form completion. If the page changes in the middle of that test, the results won't make sense.

Some WIA monitoring tools capture the page when you first set up monitoring, and use this as the background when displaying their results. Other tools capture the page at the time the report is generated, and associate results (such as a count of clicks) with specific page elements, so if you delete a button on a page, the click count for that button won't show up in reports.

To avoid this kind of confusion, don't change a page while you're testing it. Treat each WIA analysis as a test with a beginning and end, then make a change, run another analysis, and compare the results.

Visitor Actions WIA Can't See

Client-side WIA captures what happens when a user interacts with the browser, recording such events as changes in dialog box focus, mouse clicks, and keystrokes. Users may do things that have little to do with your site, but that affect how they use the application and your ability to collect data. For example:

- They can copy and paste text, save files, print content, or otherwise cause interactions between the browser and the host operating system.
- They can launch a new window or new browser tab.
- They can close the window and abandon their visits—sometimes by accident.
- They can bring another application to the forefront and interact with it.

If a visitor pastes text into a field from a clipboard, the WIA scripts may not capture that text without considerable additional effort, because there are no "keydown" events for scripts to capture. Similarly, text submitted through autocompletion, in which the browser fills in forms on behalf of the visitor, may not be stored.

When another desktop application has focus, meaning it's "on top," receiving keystrokes and mouse events, mouse and keyboard activity isn't available to the browser and whatever monitoring system you've got installed. This can happen when the visitor is interacting with another window that is covering the current window, when the visitor is browsing through a right-click menu, when the web page is minimized or in the background, or when the visitor interacts with certain Java and RIA plug-in applications or the operating system itself.

In these cases, you'll often know something's happening only because of extended periods of inactivity. Patterns of inactivity that always happen in the same place may indicate a usability issue. For example, visitors may be printing the page, and you could provide them with a Print button at that point in their visit. Or they may be going to fetch a credit card because they didn't know one was needed.

Dynamic Naming and Page Context

If your pages aren't named intuitively, as is the case in Figure 6-26, it may be difficult to understand what visitors were doing on the site.

Page tagging and the extraction of things like title tags can help with this problem, but you'll need to work with developers to ensure URLs are meaningful. Incidentally, this will also help your search engine rankings, since search engines favor sites with descriptive URLs.

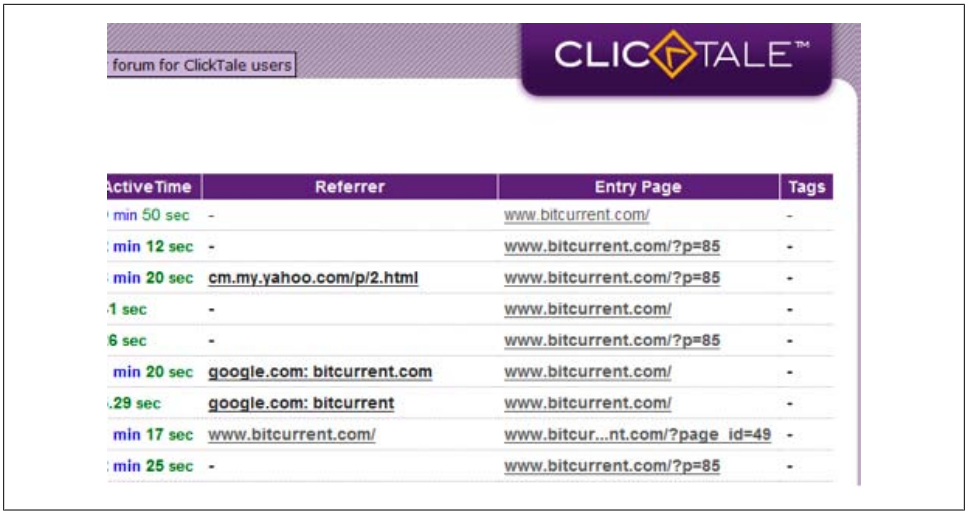


Figure 6-26. Dynamically generated page names don't give you clues about the purpose or content of the page

Browser Rendering Issues

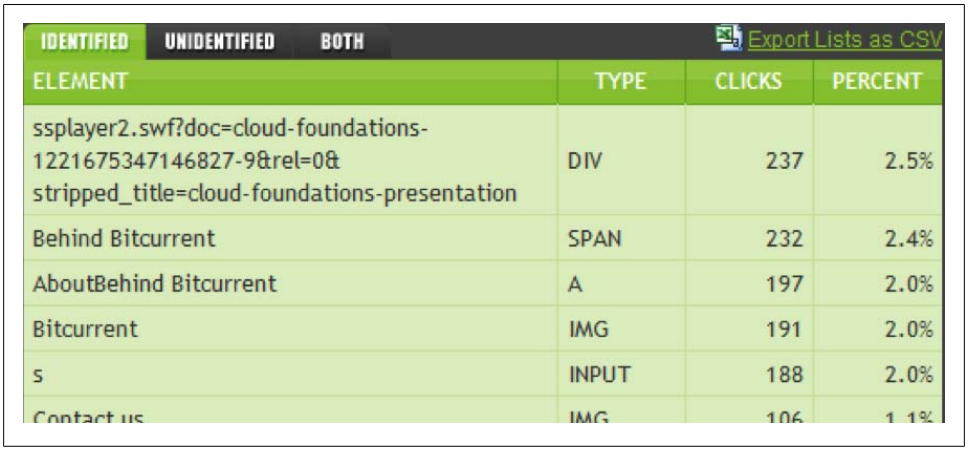
You won't always see what your visitor saw when you replay a visit. Differences in browsers, embedded RIA applications, and other factors mean that for many sites, you'll only have an approximation of the user's visit.

WIA solutions that rely on JavaScript to capture pages are less faithful to the end user experience than inline capture devices, in part because of the way in which they collect data. There are many components of a replayed page, such as embedded components, HTTP POST messages, SSL-encrypted content, and client-triggered JavaScript, that require considerable additional effort on the part of site operators and developers to get working properly in a hosted replay solution.

Different Clicks Have Different Meanings

WIA tools can overcount clicks if they don't understand the context of a page. Consider a report that shows how many mouse clicks each object on the page received in Figure 6-27.

The clear leader in terms of clicks was a `div` tag containing a Flash object. What's not readily apparent from such a report is that the Flash object was a slideshow player. As a result, those clicks were from visitors paginating through the slides in the object, not interacting with the site. Knowing this, you can disregard these outliers. Without application context, however, you might conclude that this object was popular rather than just interactive.



IDENTIFIED | UNIDENTIFIED | BOTH — Export Lists as CSV

ELEMENT	TYPE	CLICKS	PERCENT
ssplayer2.swf?doc=cloud-foundations-1221675347146827-9&rel=0&stripped_title=cloud-foundations-presentation	DIV	237	2.5%
Behind Bitcurrent	SPAN	232	2.4%
AboutBehind Bitcurrent	A	197	2.0%
Bitcurrent	IMG	191	2.0%
s	INPUT	188	2.0%
Contact us	IMG	106	1.1%

Figure 6-27. Embedded RIA objects receive more clicks because they're interactive, not because they are a popular part of the site

The Impact of WIA Capture on Performance

The verbosity of a collection tool depends on the granularity of the data you're capturing. Passive inline collection doesn't introduce delay, but any JavaScript embedded in the browser to capture mouse and keystroke interactions requires an additional GET from the visitor.

For mouse recorders, client-side JavaScript may be transmitting mouse movements at regular intervals as a compressed string. For click-placement recorders, visitors will generate a message to the WIA service each time they click. In fact, if you examine pages that use WIA you'll see the browser's status bar showing that it is communicating with these systems (Figure 6-28).

Vendors claim that their scripts consume no more than 2 Kbps of upstream bandwidth, which is negligible for many broadband visitors. For hosted replay services, there's also the impact of the secondary hit on the server to consider.

Playback, Page Neutering, and Plug-in Components

Replayed pages must be *neutered* so that the act of viewing them doesn't inadvertently trigger real actions. We want to sandbox the replay of the visit from any real-world outcomes so that we don't buy a pair of shoes or send an email message every time we replay the visit.

Since pages are made of many components, you may also have problems replaying visits that included third-party content, such as a map, or a third-party stock quote, many of which require a developer API key.

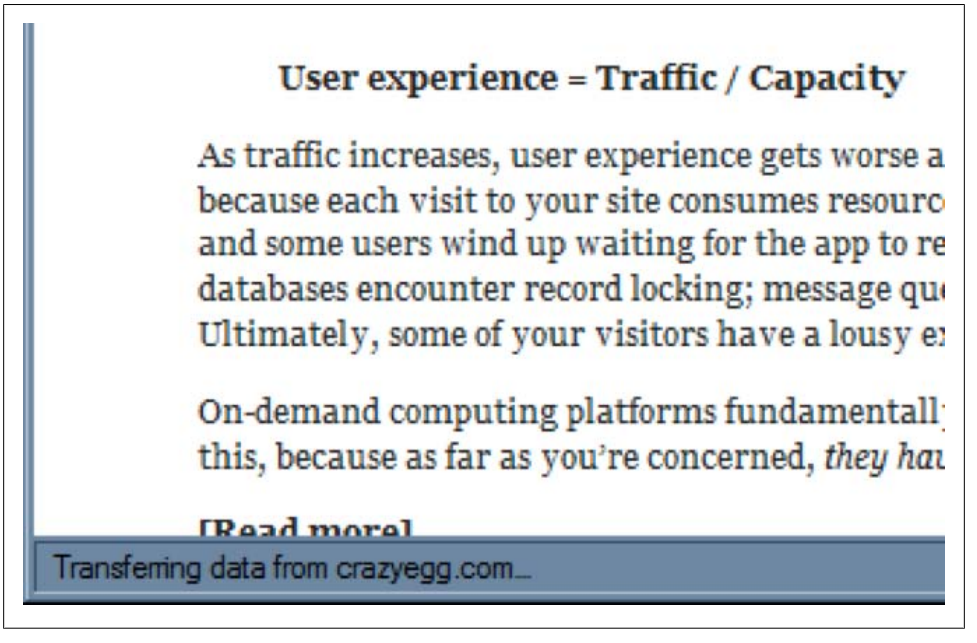


Figure 6-28. Ongoing communication between the browser and the WIA service after the page has loaded

Here's why: when a visitor to your site loads a page that includes a customized Google Map, your browser makes a request to Google. That request includes your URL (as the referring page) and the API key you've received from Google. Google issues these keys for several reasons: to prevent excessive use, to authenticate requests, and so on. When you replay the page from a third party service, the requesting URL (of the service) and the API key (from your site) won't match, so embedded third-party components may behave strangely or not work at all.

Privacy

Of all the technologies we've looked at so far, WIA is the most invasive and risky. Analytics only looks at visits in the aggregate, and RUM is primarily concerned with performance and errors. With WIA's visit capture, you're seeing what visitors saw and typed, usually without them expecting their visits to be seen by someone else.

Depending on the solution you're using, you may be decrypting and capturing a copy of a visitor's entire page, complete with personally identifiable information and credit card numbers. You're also able to uniquely identify individual visitors by the data they enter, further raising the risk of fraud and leakage.

The consequences of privacy violation may be subtler, though. Consider a visitor looking at pages about cancer treatment. Your record of his visit could have a material effect

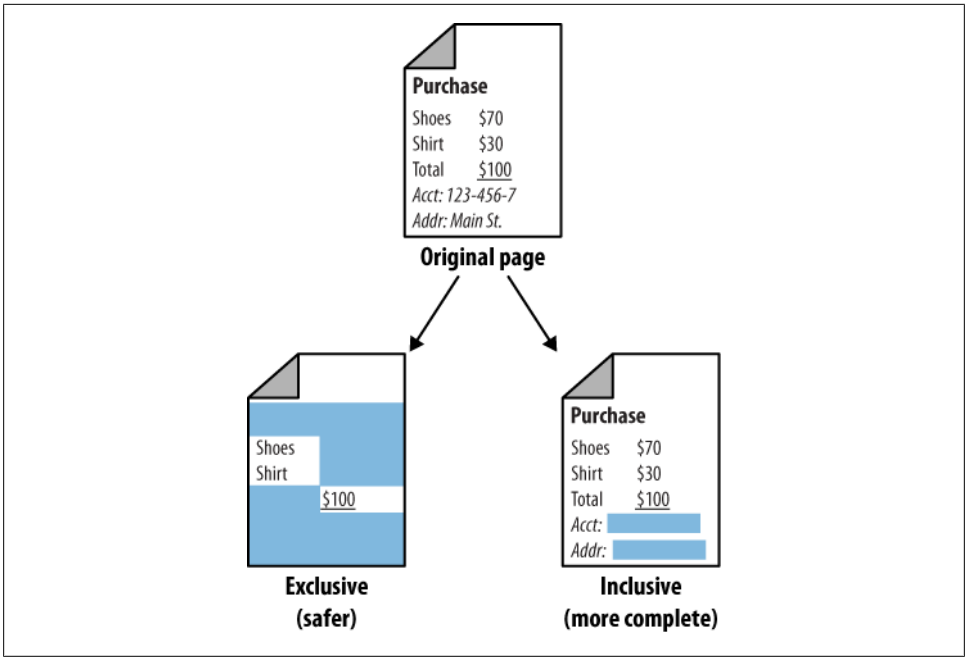


Figure 6-29. Two capture strategies collect different amounts of personally identifiable information from visitors

on his insurance premiums simply because of the pages' URLs. You need to take steps to protect visitor data, control who can access it, and delete it in a timely fashion when it's no longer needed.

Data collection: Inclusive and exclusive

Configure WIA products to capture only what you need. Unfortunately, since you're trying to understand a visitor's behavior, what you need is often the entire visit. You can't just record certain strings within the page—you need the whole page, including images and stylesheets, to see what they saw.

Because of this, WIA tools are usually configured to delete certain strings and even look for common patterns such as the four-sets-of-four-digits of credit cards, and to remove them automatically before displaying them.

A capture strategy that includes all data by default is more promiscuous—it will collect everything it isn't told to ignore, ensuring that you have maximum data for analyzing a visitor, but exposing you to the risk of privacy violations. By contrast, a capture strategy that excludes everything except for what it's told to collect won't capture the full visit, and will limit your ability to segment and review what happened. Figure 6-29 shows these two approaches.

You may want to avoid recording certain form fields, such as credit card numbers. Most WIA services won't record form data that's hidden in the first place (such as passwords), but you can also tag form elements with special strings to tell the collection system to ignore them. With ClickTale, for example, the following HTML tells the system that the form field "Acct" should not be stored by marking it as sensitive.

```
<input id="Acct" type="text" class="ClickTaleSensitive">
```

Inline devices let you define which POST parameters to collect or discard, or to configure regular expressions to extract or remove only certain portions of a page.

Remember, however, that one popular use for WIA is dispute resolution. You may be required to protect historical data without modifying it. In this case, you have to store all of the data collected in a digitally signed, tamper-evident format; you can't scrub the data when it's captured the way you could for RUM data, so your WIA tool needs to hide sensitive data when it's viewed by an operator and delete it when its storage period has expired.

Keystroke capture is an ethical hot potato

What about something a visitor types before she abandons a form?

A visitor might enter data such as her email address into a form on your website, then change her mind about submitting it. Traditional web analysis would just show that the visitor abandoned the visit on that page. On the other hand, a WIA script that's sending back keystrokes and mouse movements might capture a partially completed form. To whom does that data belong? Can you add that visitor to your mailing list? Put another way, *does data collection begin at the keystroke or the Submit button?*

Depending on the vendor, your collection scripts may trap every keystroke or every form field change. If you collect this data and the visitor leaves the site without clicking Submit, you now know things the visitor didn't necessarily want you to.

ClickTale's CEO explains their collection policy as follows:

> While ClickTale does record all keystrokes, we protect the anonymity of visitors who start to fill out online forms and do not submit them. We intentionally mask these visitors' keystrokes since they have not implicitly agreed to share their information with the website.

As consumers become more aware of the ways in which we monitor them, we expect a backlash against this sort of data collection, but given the importance of understanding conversion and abandonment, microconversion analysis will become increasingly popular, and web operators will find themselves employing and defending collection methods that may ultimately pose a privacy risk to the companies for which they work.

Web Interaction Analytics Maturity Model

As with web analytics and EUEM, companies undergo several phases of maturity as they start to embrace WIA technologies. What starts out as aggregate "where did they click" visualization quickly becomes segmentation by internal outcome, external source, usability, and learning curve analysis.

Maturity level	Level 1	Level 2	Level 3	Level 4	Level 5
Focus	Technology: Make sure things are alive	Local site: Make sure people on my site do what I want them to	Visitor acquisition: Make sure the Internet sends people to my site	Systematic engagement: Make sure my relationship with my visitors and the Internet continues to grow	Web strategy: Make sure my business is aligned with the Internet age
Who?	Operations	Merchandising manager	Campaign manager/SEO	Product manager	CEO/GM
WIA	Click diagrams showing "hot" areas on key pages	Segmentation of visitor actions (scroll, drag, click) by outcome (purchase, abandonment, enrollment)	Segmentation by traffic source (organic search, campaign) and A/B comparison; visitor replay	Learning curve analysis; comparison of first-time versus experienced users; automated A/B testing of usability	Product specialization according to usability and user groups; usability as a component of employee performance