A Practical Guide to Measuring Usability

72 Answers to the Most Common Questions about Quantifying the Usability of Websites and Software

By Jeff Sauro

jeff@measuringusability.com || Twitter: @MsrUsability

Measuring Usability LLC
761, Garfield St
Denver CO 80206 USA.
jeff@measuringusability.com

A Measuring Usability LLC Publication

ISBN: 1453806563

Table of Contents

Acknowledgements

I would like to thank James R Lewis, PhD and Joseph Dumas, PhD for their thorough comments and suggestions on earlier drafts of this document. I am also grateful for the comments and suggestions on the content and organization from Nigel Bevan and John Romadka.

Testing

1. Can you measure usability?

When you think of the ease of use, usability, learnability and intuitiveness of an interface, you might think these aspects are warm and fuzzy and can't be measured. Although usability certainly reflects aspects of our experience that are more difficult to quantify than, say, taking a temperature, usability can be measured. The ISO standard of usability (ISO 9241 pt. 11) defines usability as the intersection between effectiveness, efficiency and satisfaction in a context of use. You could say the standard defines usability as the ability to get things done fast and be happy about it.

Usability is typically measured by having users attempt realistic tasks on an interface. You can watch users informally, remotely, or in a lab. The key idea behind usability is actually seeing what happens when users try to complete tasks. Can they complete the task? Does it take them too long? Do they make mistakes? What problems do they have? What do they think of the application after having used it? Do they love it, hate it or are they indifferent?

Contrast this with asking users: Would you use the product? What would you do with it? Do you like it? There is a place for surveying wants and needs, and there is some overlap in the data, but it is different from usability testing. When you observe users attempting

tasks, you gather a wealth of data on actual usage, including the type of problems users have that they cannot articulate or tend to forget when it comes time to provide feedback.

2. What is the difference between a formative and summative usability test?

There are generally two types of tests: finding and fixing usability problems (formative tests) and describing the usability of an application from average task times, completion rates and other metrics. The terms come from education where they are used in the same way except to describe student learning (formative—providing immediate feedback to improve leaning vs. summative—evaluating what was learned).

People typically think formative means a qualitative test and summative means a quantitative test. While that might be a convention, I hope to change the view that formative evaluations don't involve quantitative measurements. Just because the goal is to find and fix as many problems does not mean you cannot quantify the problems, which users encountered them and use statistical methods to know what the probability a user will encounter a problem.

What's more, measuring completion rates, tasks times and errors help identify and qualify usability problems. As an example, see Task Times in Formative Usability tests: http://www.measuringusability.com/formative-time.php

In a recent survey I sent to the subscribers of MeasuringUsability.com there was a 2 to 1 ratio between the frequency of formative and summative tests. On average, respondents reporting conducting 13.8 formative usability tests over a 2-year period and 6.1 summative tests during the same time.

In my experience no usability test is strictly formative or summative. Even when I've conducted more summative-like benchmarking tests, I still record problems users have in an effort to get them fixed. During more formative-like evaluations I record the same metrics as I do in a summative-like test and use it as a comparison point for future testing iterations.

3. What is a quantitative usability test?

The majority of usability testing involves having users attempt tasks and finding and fixing as many problems discovered during testing as time and funds allow (called formative evaluations). This process in turn should make an application more usable. Less frequently, measurements are made of the usability of a system to establish a benchmark of usability (called summative evaluations). On average there are twice as many formative as summative ones (see the section: "What is the difference between a formative and summative usability test?").

The terms “Summative” usability tests and “quantitative: usability tests are often used interchangeably by usability engineers because this is often the only time quantitative measurements are taken. For that reason, it is difficult to conduct a summative usability test and not call it quantitative. A formative test can and should also be quantitative.

Since usability testing is often thought of as describing problems and at best counting the number of problems (a qualitative activity) people don’t see a need to count and measure things. When there are many problems to fix this non-numeric approach to usability testing might suffice. When the low-hanging problem fruit has been picked you can no longer rely on intuition to discover problems. Instead you need to systematically record problems and use other metrics such as task times, completion rates, errors and satisfaction questionnaires to both diagnose and qualify usability problems. With these numeric quantities you can statistically support your claims even at very small sample sizes (See the Analysis Section below).

With a qualitative formative test you describe a problem and perhaps report that 3 of 5 users encountered it in the test. With a quantitative formative approach you describe the problems, report how many and who encountered it, measuring how long tasks takes to complete, what percent complete and the number and types of errors which were caused by the UI problems.

Using confidence intervals you then qualify how likely problems are to occur, the most likely percent of users that can complete a task and the range for the task time and satisfaction scores. Ideally you can also compare the metrics using external benchmarks (from a prior version, an industry standard or customer requirement). Even if you don't have an external benchmark, by quantifying the usability you will have a benchmark on the application tested to use in future tests.

A quantitative usability test is then a usability test (both formative and summative) with systematic recordings of events by using a variety of metrics and describing those events using numbers.

4. What are the advantages of a Quantitative Usability Test?

If you can't measure it you can't manage it. Business decisions are largely based on numbers. If you can't quantify usability problems, user performance and the benefits from usability activities someone else will (e.g. finance or product development).

Imagine a marketing department asking for more money to conduct a direct-mail campaign and their only justification was that marketing is a critical business advantage. Now contrast that with an argument that showed that in a previous direct-mail campaign the response rate of 3% was more than twice the industry average and was achieved from

testing a sample of the campaign on customers. The test led to a more effective message and ultimately in an additional $100k in sales.

If a usability manager asked for money to conduct a usability test based only on the argument that usability is critical for business then you're making the same argument. Contrast that with an argument that showed the last usability test reduced transaction time by 50% and reduced check-out abandonment rate by 20%. It was achieved from fixing problems with button labeling and placement and led to an additional $100k in sales over a year.

Quantifying usability means making better decisions about usability data. By quantifying the reduction in problems, the increases in efficiency and satisfaction will ultimately lead to better business decisions.

5. What are the basic steps for conducting a quantitative test?

1. **Determine what you want to test**: What parts of the application and what functions.
2. **Why are you testing**?
    a. Do you want a benchmark of usability and performance?
    b. Are you trying to find and fix as many usability problems as possible?
    c. Are you comparing an application with a competing product?
3. **Write a minimum of 3-5 task** scenarios that have users attempting realistic tasks in which they interact with the parts of the application you want to test.
4. **Recruit a set of users** to test at a minimum aim for between 10 and 20 in total. For goal 2b, test a subset of your users as part of an iterative testing approach (see section below "Do I only need to test with 5 users?"). For goals 2a and 2c, you'll have a margin of error of around 20% for task times and completion rates for a sample size of 20. At a sample if 10 it will be close to 30%.
5. **Test your users**; plan on 1 to 2 hours per user, knowing some complex tasks can take longer.
6. **Collect as many metrics as possible**--they are free to collect and provide a wealth of information. At a minimum, collect task times, completion rates and UI Problems. You should also get at least some measure of satisfaction (task level or test level). I usually collect both and errors.
7. **Code your data** and analyze it.

8. **Generate confidence intervals** around your means and proportions.
9. **Summarize the results** in graphs and a **report or briefing**.

6. How long does it take to conduct a quantitative usability test?

In my experience, it takes a minimum of a month from start to finish. This assumes you are testing between 10 and 20 users and each testing session lasts between 1 and 2 hours.

1. **1 week to recruit**—emailing, calling and scheduling.
2. **1 week to test** –you have to have your users packed in. At most one facilitator can test 4 users in a 9-5 workday for a total of 20 users. You need some bathroom breaks, lunch and the inevitable run-over time. You'll be exhausted at the end of the day.
3. **1 week to analyze the data**: If you took good notes and caught most things, you'll still need some time to clean up your data, review video files if necessary and put the data in the right format. You then need to start computing the averages, counts and confidence intervals which will go into the report.
4. **1 week to write a report or prepare a PowerPoint briefing**: You incorporate your findings from the analysis into the report. You will typically have to revisit the analysis where you notice gaps or want to explore things in more details. The last 2 weeks are often a mix of analyzing, reporting and communicating.

7. How long should the test session last?

There always seems to be too many things you need to, want to or are told you need to test. Unfortunately the attention span of a user isn't unlimited. In my experience, you don't want a test session to last more than 2 hours and most should be done is less than 1.5 hours. The sweet spot between 1 and 2 hours allows you enough time to test between 5 and 15 tasks, depending on how long they are. It also allows you enough time to sign and read an informed consent, fill out pre-test material and answer post-test questions. You should plan on leaving at least 10 minutes for getting the user into the lab, thanking and debriefing and a few minutes for late-arrivals. There is a certain amount of overhead in recruiting and getting users in a lab so I have a hard time having tests last less than say 45 minutes. If I only have a few tasks and it only takes about 30 minutes to complete, I usually add things like task repeats, new questionnaires or slight variations on other tasks. Consequently, my usability test sessions almost never last less than an hour. Most users will plan on spending at least that much time. They scheduled it in their day and if they are getting paid probably don't mind spending the hour telling you how to fix the software! If you're at a conference or other public place then you'll be competing with a lot of distractions. In those cases—get the user in and out as quickly as possible.

8. How do I select the users for testing?

One of the hallmarks of usability testing is having actual users attempting realistic tasks with your application. You should identify the profile of your user population. Are they

accountants, small business owners or women under the age of 30? Be as specific as possible and use this profile or "persona" to help recruit from this target user population.

Getting the right contact information and scheduling users to come in does take effort, but bringing in just anyone to test can be a problem. Because you will use the data from a sample of users to extrapolate to the entire population of users, it is critical that you test users that reflect the whole population. The more specific your application is to a domain or industry, the more important it is that your sample is representative. Ideally you can select your users in some semi-random way—that's not systematically biased to say just happy customers. It is more important to get representative users than to select users at random.

9. What are common usability metrics?

Although there is an international standard for measuring usability (ISO 9241), the standard leaves open the questions of how to measure effectiveness, efficiency and satisfaction. Metrics should be collected for each user then summarized as averages across all users (See the Reporting section below). The following metrics are the most common ways of measuring usability:

1. **Usability Problems (UI Problems) encountered** (with or without severity ratings): Describe the problem and note both how many and which users encountered it. Knowing the probability a user will encounter a problem at each phase of development can become a key metric for measuring usability activity impact and ROI. Knowing which user encountered it allows you to better predict sample sizes, problem discovery rates and what problems are found by only a single user (See the section below "How do you measure UI problems?" for more on this fundamental usability metric.)
2. **Time on Task**: How long it takes a user to complete a task in seconds and or minutes. Start task times when users finish reading task scenarios and end the time when users have finished all actions (including reviewing).
3. **Completion Rates**: A simple metric that tells a lot and is easy to understand. Typically measured as a binary value (1= passed the task and 0= failed the task).

Some also code it using grades of completion (25, 50, 75, 100) but I prefer keeping it as a simple pass or fail.

4. **Errors**: Any unintended action, slip, mistake or omission a user makes while attempting a task. Record each instance of an error along with a description. For example, "user entered last name in the first name field". You can later add severity ratings to errors or classify them into categories. Errors provide excellent diagnostic information and, if possible, should be mapped to UI problems.
5. **Post-Task Satisfaction Ratings**: Asking a user immediately after completing a task how usable they though it was. Keep the questions to 1 to 5 short numeric questions and optionally collect comments. See the ASQ for an open source example of a post-task questionnaire and the questionnaires examined in Sauro & Dumas (2009).
6. **Post-Test Satisfaction Ratings**: Ask between 10 and 50 questions about the user's overall impression of the usability and learnability of the software. Post-test questionnaires provide a view of the perceived usability and are less affected by the task performance then post-task questionnaires. It is best to use a standardized questionnaire which has been shown to be reliable and valid. For example, the System Usability Scale (SUS) is a popular free 10-item questionnaire.
7. **Clicks / Pageviews** : For websites and web-applications, these fundamental tracking metrics might be the only thing you have access to without conducting your own studies. Clicks have been shown to correlate highly with time-on-task.

Both these measures can be used for measuring conversion rates or purchases which are fundamental to measuring the effectiveness of e-commerce systems.

8. **Comments**: You need a way to collect the "why" behind the numbers. I provide open-ended questions with my post-test questionnaire and post-task questionnaires. These comments, while more difficult to quantify, provide valuable insight into users' thoughts and can make great quotes for a report.

10. How do I record the measurements?

You can be as low or as high tech as you want. All you really need is a stop-watch and a piece of paper. Set your paper up in a grid with places for times, errors, completion rates and comments. I use an Excel sheet and a custom built web-application with some macros that will record the time and allow me to enter comments and quickly export the data. Some software like TechSmith's Morae include time recordings but, in my experience, it adds too much time to review the recordings, identify the start and stop points then export the times. I typically use these applications as a way to record the usability session, and then return to them when I need to review a task if I missed something or for highlight clips.

11. Do you sit in the room with the user or behind a one-way mirror?

In usability testing you want to create a realistic testing environment. Since users don't typically have someone watching their every move, any observation will be artificial and affect task performance. If I'm testing with access to a lab with a one-way mirror I sit on the other side and speak to the user over a microphone. From the control room I'm able to scribble notes, type and move between different applications without disturbing the user. Such control rooms also allow you to have other observe the test (such as product management or marketing). Sure, users still know they're being watched when you're just a voice from behind the mirror, but it minimizes some of the distractions.

If I don't have access to a lab, such as when I'm testing at a conference, then I'll sit as far away as possible from the user and watch their actions on a separate monitor. Inevitably a user will hear or see me taking notes and say something like "oops, did I do something wrong?"

12. Do you compensate users for their time?

Unless the users are employees of your own company, you should expect to pay them for their time. I typically see compensation rates from $50 to $200 for a usability testing session lasting between 1 and 2 hours. It depends on how long the session lasts and how hard it is to recruit and bring in actual users. Compensation can also be in the form of discounted or free products from your company. If you have a limited budget you can consider conducting a raffle or drawing for a prize (check local laws to be sure this is legal where you test).

Sometimes you have no budget and rely on volunteers, which is fine since most users who have the time and use the product a lot don't mind sharing their opinions and experience if they think it will make a difference. The tradeoff is between cost and the likelihood that the participant we agree to participate and actually show up.

I've done a fair amount of testing at company conferences where it is often our only time to get access to international users in person. In those situations I usually provide a smaller token for their time, such as a $25 Amazon gift card, a chance to win an iPod or tchotchke.

13. Do I need to conduct a pilot test?

It doesn't take much to discover problems with your tasks, application or testing setup. Everything might make sense to you, but the test can fall apart with some small oversights. In my experience having just one user (preferably a patient one) attempt the tasks will allow you to iron out many of the problems.

If you can't pilot test a user at least have someone else read through your tasks and try them yourself with cameras rolling in the lab. If you detect problems in your tasks and test when you have a full day of testing, it can be difficult to make changes in time.

14. Do you ask users to think aloud or have them complete the tasks silently?

While it might seem like having users think aloud while they complete a task will increase the time on task—the literature is actually mixed. Some studies report no difference, some show longer task times for thinking aloud and others report faster task times for thinking aloud. It has been hypothesized that the reason users can perform faster while they think aloud is because this vocalization allows them to think more clearly through their approach for completing the task (it's like explaining a problem to someone when the process of explaining helps solve it).

The key issue is whether you probe (ask for follow-up explanations after comments). Probing will increase the task time as users take longer to explain what they meant unless you are able to subtract the probing time from the total task-time. If you need to probe, it doesn't necessarily render the task time metric useless. If you consistently probe users while they think aloud at different stages of testing then task time can still be used to compare the designs iteratively. There's a tradeoff between the information gains of probing (keeping in mind that people don't always know why they do things) and the increase in variability of time measurements, which is partly dependent on whether the study is primarily formative or summative.

Whether you ask users to think-aloud largely depends on the goals of the study (more formative vs. summative) and the phase in the development of the interface. If your primary reason for testing is to discover as many problems as possible, then having the user think aloud will help uncover issues more easily. This is especially true if you are testing on an early phase prototype.

If your primary motivation is to generate a benchmark of performance or if the interface is in a later (or released) phase of production, then it may be better to not ask the users to think aloud.

Most of my testing is on late phase or realized applications so I ask my users to complete the task as quickly as they can without rushing. I don't say anything about thinking out

loud. Some users will express their thoughts as they complete tasks and others are silent. When I see a problem that I want to know more about, I wait until after the task is complete and ask the user to elaborate more.

15. Is there a difference in data from unattended versus lab-based tests?

The ability to quickly collect data from users by using the Internet to test websites or applications is generally a good thing. You can collect data from a much larger sample of users by having users test themselves. With a larger sample size your metrics will be more precise estimates of actual user performance.

The research is still coming out on how data differ between unattended and lab-based moderated testing. I recently helped organize and participate in the Comparative Usability Evaluation (CUE-8) of quantitative usability data. About half the teams used moderated testing and the other half used remote unmoderated testing. Many variables affected the outcome of the data, most notably the experience of the team (some teams had never conducted a quantitative usability test). For that reason, it was difficult to make judgments about the similarity of remote vs. local testing (since differences in results might be entirely due to differences in testing approaches).

The major problem with remote testing is you tend to get a lot of bad data—task times that are impossibly short (1-10 seconds) and others that are way too long. Since the users aren't directly observed, you have to decide which times to "throw-out," and this inserts a fair amount of subjectivity. Although you can collect task times automatically, you have to add questions at the end of each task to determine if a user completed the task. For example, you would ask what the location of the nearest rental car facility is to an address (something that has an objective right answer). Without these questions and a fair amount of diligence, you can easily get bad data (as many of the teams who had little experience conducting remote tests did).

I was, however, convinced that you can get valid results with the right approach to unattended testing. I compared the completion rates, task times and SUS scores I obtained from 12 users to another team who tested over 300 users using a remote testing approach. While I cannot reveal the team members on this team, I can say that they have a lot of experience conducting lab and remote tests. The figures below show comparisons of my metrics and theirs. The large overlap in the confidence intervals by task show a surprising amount of agreement. It is even more surprising considering I tested only 4% of the number of users as this team. The System Usability Scales (SUS) scores administered after the tests were within 2% of each other with scores of 79.5 and 78 (79.5-78)/79.5 =1.89%.

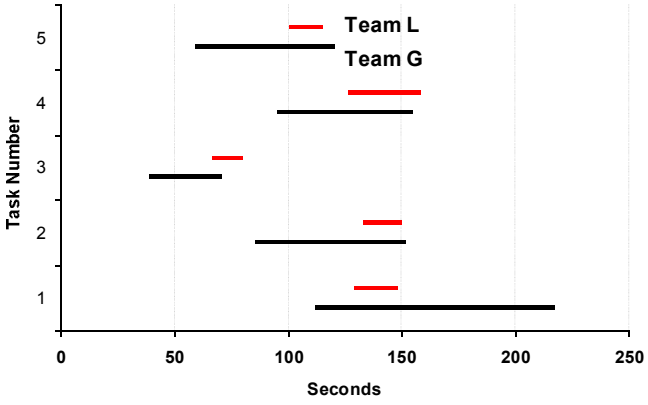

Figure 1: Comparison of 95% Confidence Intervals for Task Times for Team G and Team L. The large overlap in the confidence intervals shows similar results despite using a remote vs. lab based testing approach.

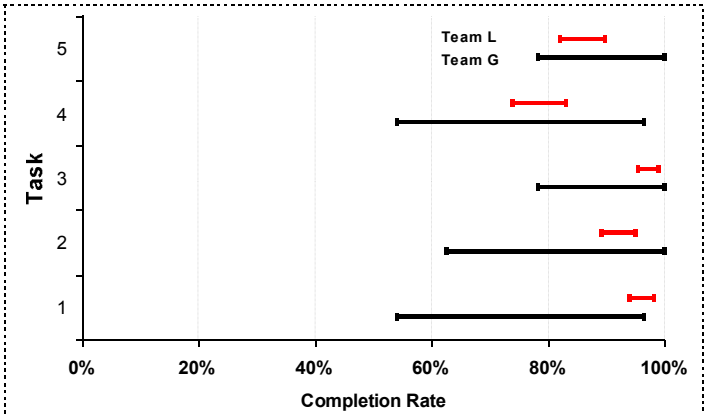

Figure 2: Comparison of 95% Confidence Intervals for Completion Rates for Team G and Team L. The large overlap in the confidence intervals shows similar results for completion rates despite using an unattended vs. attended testing approach.

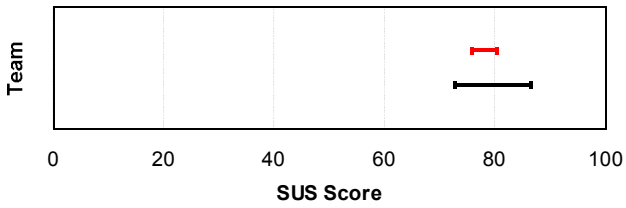

Figure 3: Comparison of 95% Confidence Intervals for S for Team G and Team L The large overlap in the confidence intervals shows similar results SUS scores despite using an unattended vs. attended testing approach.

Team L took about half the time as Team G (21 vs. 40) hours but provided data on 312 vs. 12 users. These results show that small sample sizes from attended testing still lead to similar conclusions as unattended testing. The difference is you have less uncertainly and it can be done in about half the time.

16. Is user testing the only way to measure usability?

Expert Reviews & Heuristic Evaluations

A usability engineer needs many tools. There isn't always time or resources available to test users. In such cases, expert reviews or heuristic evaluations (HE) can be used (which are technically different but everyone seems to think of them as the same thing). An expert-review is a more generic term for an inspection of a user interface by someone trained in usability (and ideally the application domain). A heuristic evaluation has one or more experts follow a list of rules (heuristics) to compare against the interface (e.g. "The system should speak the users' language…").

Having someone with an understanding of usability principles review an application is better than nothing. In my experience, though, requests to have a "usability expert" review an interface come too late to do anything and are seen as some sort of certification process.

The major problem with expert reviews and heuristic evaluations is that they are perceived as opinions, often with little data to justify them. The method of HE is best used when at least three experts can evaluate an interface. In my experience, it is usually easier to find three users than three experts who can conduct an expert review. The result is that usually just one expert reviews an application (and often without the heuristics).

There have been numerous studies debating whether experts agree on the problems found in an interface and whether such data is better than that obtained from usability testing. For example, is it really a problem if an expert says it is but a user didn't have a problem with it in a usability test?

At a recent conference for usability professionals (UPA 2009) the efficacy of Heuristic Evaluation was discussed. On the panel were the two practitioners who get credit for creating the Heuristic Evaluation method, Rolf Molich and Jakob Nielsen. Although Rolf, Jakob and the other panel members stated that HE has some value, Rolf's view after twenty years sums it up well: "Heuristic Evaluation is 99% bad."

My recommendation: When you have no time and no budget and absolutely cannot test any users, then an expert review is better than nothing. Also if you are in an organization with several other usability specialists, sometimes a HE can be done quickly. Ideally you would conduct some sort of expert review **prior** to user-testing. Just the process of preparing tasks for a review or potential usability test might capture obvious problems any developer would agree needs to be fixed without any user validation. What's more, conducting an expert review prior to testing allows you to determine whether the problems you found in the review are actually encountered by any users. This recommendation would only apply to formative testing as an expert review cannot generate reliable estimates of user performance. For that we turn to Keystroke Level Modeling.

Keystroke Level Modeling: Measuring Productivity

It is often the case that you want to estimate how long it will take experienced users to complete a task. Bringing users into a lab is expensive and time consuming. What's more, you need to have some application to test users on and you need to have users (this is a problem if it is a brand new application). Instead of empirically measuring users attempting tasks on an interface, an analytic method called Keystroke Level Modeling has been shown to estimate the task time of skilled users to within around 10% (recall that the average margin of error in a usability test is in excess of 30% when there are less than 10 users). KLM was developed in the late 1970's and 1980's at Xerox PARC and Carnegie Mellon and was described in the seminal book Psychology of Human Computer Interaction (Card, Moran, & Newell, 1983).

In KLM, you estimate the task time by assigning each sequence of operations (keystroke, pointing, clicking and thinking) millisecond-level times that have been experimentally refined over the decades. In my experience, the method works quite well for tasks that are done repeatedly (for example, call-center agents who are repeatedly performing the same tasks).

For more information on KLM, see the recent paper Sauro, J. (2009) "Composite Operators for Keystroke Level Modeling." in *Proceedings of the Human Computer Interaction International Conference (HCII 2009), San Diego CA, USA* available for download here: http://www.measuringusability.com/papers/Sauro_HCII2009.pdf

www.MeasuringUsability.com

Tasks

17. How do you define the task-scenarios?

A task is made up of the steps a user has to perform to accomplish a goal. A task-scenario describes what the test user is trying to achieve by providing some context and the necessary details to accomplish the goal. I use the terms interchangeably in this document. Writing good task scenarios is a bit of an art, but there are good task scenarios and bad task scenarios. There are sections of books dedicated to crafting the right task scenario. A good task scenario:

1. **Has a bit of context** to get the user thinking as if they were actually needing to perform the task. For example, "you will be attending a conference in Boston in July and need to rent a car."
2. **Provides specific details the user needs to know or enter**: dates, locations, number of items, model numbers etc. Don't be so vague in your task that users have to guess what you want them to do. For example, "you need to rent a mid-sized car on July 21st at 10am and return it on July 23rd at noon from Boston's Logan Airport."
3. **Doesn't tell the user where to click and what to do**: While providing specific details is important, don't walk the users through every step. Leading a user too much will provide biased and less useful results. For example, instead of saying "click on the small check box at the bottom of the screen to add GPS" just say "add GPS to your rental car."
4. **Has a correct solution**: If you ask a user to find a rental car location nearest to a hotel address, there should be a correct choice. This makes the task more

straightforward for the user and allows you to more easily know if a task was or wasn't successfully completed.

5. **Don't make the tasks dependent**: It is important to alternate the presentation order of tasks as there is a significant learning effect that happens (See the section "Should all the users get the tasks in the same or different order?". If your tasks have dependencies (e.g., create a file in one task then delete the same file in another task) then you will have problems with counterbalancing. Do your best to avoid dependencies (e.g. have the user delete another file). This isn't always possible if you're testing, say, an installation process.

It takes some practice in balancing not leading users through a task on one hand and not making the task too difficult on the other.

18. What are some examples of good task-scenarios?

Example 1 : Update a saved expense report[This is a task statement. The scenario follows.]

Earlier you started to create an expense report but ran out of time. You saved your work with the description "UPA Conference" You want to add an expense for sending a FedEx package to the report.

Step 1. Find the report and open it to update it

Step 2. Add the following expense:

Amount:$9.00 cash

Expense Type: Miscellaneous

Justification: "Sent FedEx Package"

Step 3. Review the report and submit it for approval

Step 4. Return to the Expense Report home page

Example 2 : Find the nearest rental car location

Assume that you are staying at the Hilton Hotel in downtown Portland, Oregon, USA. Find the address of the nearest Budget rental office.

The Hilton Hotel address is 921 SW Sixth Avenue, Portland, Oregon, United States 97204

19. How do I select the tasks for testing?

Usability testing is about watching users do tasks which should simulating actual usage. An initial usability test should focus on core functionality—areas that will see the most usage. You may be able to get this data from server logs, support calls, surveying the users, or transaction data. Sometimes a program or marketing manager will know what users typically do. If you are tasked with testing a specific piece of functionality, then naturally your test would need to focus on that area.

For example, as part of a recent test of a rental car website, the core tasks we identified were: renting a car, renting a car with additional options, finding the address of a rental location and finding the opening hours of a rental location on a Tuesday morning. For an expense reporting application, we had users create an expense report, create an expense report with trip mileage, update an existing report and check to see if an expense report was paid.

Sometimes you have to test more than the core (even if it is the initial usability test). As a balance you can have some core tasks tested first and if there is still time during a test session, test a set of peripheral tasks that include the fringe functionality.

20. If users select their own tasks, can I still measure usability?

There may be times when you want to see what users do on their own. You might have a few predefined tasks and some open-ended tasks where users attempt to complete their own tasks. This might be especially helpful in the early stages where you want to discover as many usability problems as possible. If you have **no** predefined tasks, then you will have too much variability to make sense of task times. Completion rates, errors and satisfaction scores would still be valid (the task times would be of interest but not averaged across users).

In fact, I recently had 137 users answer a questionnaire after they attempted to make a purchase on a website for an item **they** selected. They reported whether they were able to complete the task and answered several questions about the perceived ease of use of the site. This wasn't a conventional usability test, but still measured some aspects of ease of use. I almost always use pre-defined tasks in my usability tests.

21. How long should a task scenario last?

You want enough interaction to look for usability problems but not so long that a user forgets what their task is and gets tired. A broad rule of thumb is that effective tasks last between 30 seconds and 7 minutes. A task should be a meaningful unit of work Some don't even take that long and others might take much longer (such as a product installation task).

For example, adding a contact to a list-manager might only take 10 seconds whereas configuring the user permissions on a database might take 10 minutes. Often the task length is decided based on the total amount of time you have with the user (see test length above). One complication is that you don't often know how long users will take to perform a task. You can try the task out yourself and double or triple the time as a guide. In many cases a few problems and tasks can take a lot longer than even this buffer. When that happens, you may have to stop the task before the user has finished it.

For example, if you have only 1 hour with a user, you need to leave time for introductions, a little down-time between tasks, answering post-test questions and debriefing the user. These steps can easily take 10 minutes, leaving you with 50 minutes for the task-testing portion.

22. How many tasks should I have the users attempt?

This is largely a factor of how much time you have with each user, but definitely plan on testing more than one task. If a typical task lasts around 5 minutes (including the time it takes a user to read the task scenario and ask questions) then you'll find it hard to get more than 10 into an hour of test time. Of the 120 or so usability tests I have in a large dataset, 75% have fewer than 12 tasks and almost all have between 5 and 15 tasks.

23. Should all the users get the tasks in the same or different order?

The short answer is to not present them in the same order. When users sit down in a usability lab and start the testing scenarios they are still getting acquainted with the setup of the application, the different data and the people staring at them behind the one-way mirror. The first few tasks a user attempts are heavily influenced by the unfamiliarity effect. As users get more comfortable and "warm-up," their task performance improves (fewer errors and shorter task times). The consequence is that the first few tasks are penalized by the longer times and higher errors. In many cases the effect is so large it will cause you to think some tasks are exhibiting poor usability when in fact they are just artifacts of the warm-up period.

To offset this penalization you can alternate the order of the tasks the users receive (called counterbalancing). Counterbalancing task order is like randomization, except you are more methodical in how you alternate the order (true randomness can still result in some tasks always being presented earlier). A counterbalanced design is one in which each task is presented first, second, third etc. If you have 5 tasks then you need 5, 10, 15 or multiples of 5 users to be balanced. In practice, what matters is that the same tasks aren't always presented first. Even if you can even break the tasks into half or thirds and alternative those sections you'll offset much of the nuisance learning effects. See the section below on how to counterbalance.

24. How do I counterbalance the order of the tasks to avoid a learning effect?

Counterbalancing, or alternating the presentation order of your tasks is an essential step to minimize the learning effect that happens as users get warmed up during a usability test. The easiest way to counterbalance is to use a Latin square. Simply take the number of tasks you have by the number of users and create a grid. Then alternate the starting task by moving it to the end.

User	First Task	Second Task	Third Task	Four Task	Fifth Task
1	A	B	C	D	E
2	B	C	D	E	A
3	C	D	E	A	B
4	D	E	A	B	C
5	E	A	B	C	D
6	A	B	C	D	E
7	B	C	D	E	A
8	C	D	E	A	B
9	D	E	A	B	C
10	E	A	B	C	D

Table 1: Example of counterbalancing tasks using a Latin Square design. The Letter represent the actual tasks and are alternated to minimize nuisance learning effects.

This approach should eliminate the bulk of order effects from a usability test, there will still be some sequence effects (e.g. task D almost always follows C) but will remove much of the unwanted learning effects. To remove both sequence and order effects requires full counterbalancing—and more tasks. Full counterbalancing – using all possible orders of tasks to control not only for position but also for all sequential effects, requires n! permutations, where n is the number of tasks.

If for some reason you cannot alternate your tasks (for example the task is an installation task that all users must perform first), then give your users one or two warm-up tasks to offset the learning effects.

25. Should users repeat tasks or only attempt them once?

Time is a scarce commodity during a usability test and you're usually trying to test too many tasks already. For that reason, users typically only attempt tasks once. With only one task attempt you have a measure of **initial** ease of use, as opposed to a more sustained metric of ease of use. To complicate matters, users are usually a bit out of their element when they're in the lab (or online for an unmoderated test) and you get more error in your measurements. If you're primarily interested in "walk-up-and-use" features of an interface, then once might be enough. If you're just collecting UI problems (I recommend you also collect task-time, completion rates and satisfaction scores in a formative study) then you're probably less likely to find these after users get some familiarity with the interface. Task time will be heavily influenced by the unfamiliarity of the lab environment, tasks and people behind one-way mirrors.

If you have time to repeat a task at least once, it has been my experience that the second measure is more stable (has a smaller standard deviation) and is probably a more accurate measure of how long it will take a user to complete a task. In addition, the difference between the first and the second task attempt can be used as a measure of learnability (see sections on learnability).

When repeating tasks stagger the order so the same task isn't repeated immediately. For example, if you have five tasks and want to repeat them once, have the user attempt them

as 1, 2, 3, 4, 5, 1, 2, 3, 4, 5 instead of 1,1,2,2,3,3,4,4,5,5. On the repeated task change the information slightly so the user can't just mindless enter information. Change the numbers, names or other details slightly but keep them at the same length so you can compare the times between iterations.

Sample Size

26. Do I only need to test with 5 users?

One question I get a lot (citing the article by Jakob Nielsen at www.useit.com/alertbox/20000319.html) is, "I've heard you only need to test with 5 users, is that true?"

There are a lot of strong opinions about the magic number 5 in usability testing and much has been written about it. As you can imagine there isn't a fixed number of users that will always be the right number (us quantitative folks love to say that) but testing with five users may be all you need for discovering problems in an interface.

The five user number referenced in the article comes from the number of users you would need to detect approximately 80% of the problems in an interface, **given that the probability a user would encounter a problem is about 31%** (See Lewis 2006). Most people either leave off the last part or are not sure what it means. This does not apply to all testing situations such as comparing two products or when trying to get a precise measure of task times or completion rates (see the sections below on "What is the minimum sample size I should plan for" and "How do I determine my sample size").

For example, let's assume that there is a problem with the login function of a web-application. When users attempt to login they accidentally reset their password and it requires a call to tech support to fix. Let's imagine that this affects 31% of the users of a

website—which is quite a lot. So the question becomes, if a problem occurs this frequently (affects this many users) how many users do I need to observe to have an 80% chance of seeing it in the lab? You might be tempted to think you only need 3 to see it once, but chance fluctuations don't quite work that way at small sample sizes.

To find out you use the binomial probability function given the probability of a problem is .31. Of course not all problems affect 31% of users. In fact, in released software or websites, the likelihood of encountering a problem might be closer to 10% or 5%. When problems are much less likely to be "detected" by users interacting with the software, you need more users to test to have a decent chance of observing them in a lab. For example, given a problem only affecting 10% of users, you need to plan on testing 15 to have an 80% chance of seeing it once.

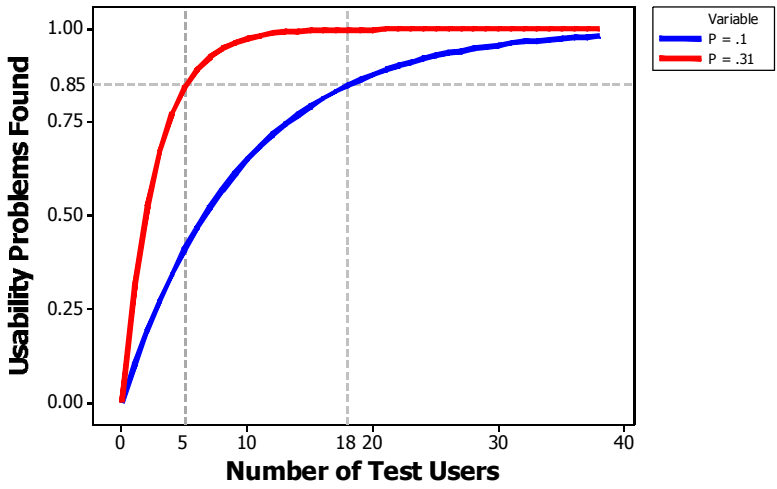

Figure 4: Difference in sample sizes needed to have an 85% chance of detecting a problem that affects 10% of users vs. 31% of users,

Of course you don't know what the probability a user will encounter a problem. In fact, you often don't even know if there is a problem—if you did you'd fix it! As a strategy, pick some percentage of problem occurrence, say 20%, and likelihood of discovery, say 80%, which would mean you'd need to plan on testing 7 users. After testing 7 users, you'd know you've seen most of the problems that affect 20% or more of the users. If you need to be surer of the findings, then increase the likelihood of discovery, for example, to 95%. Doing so would increase your required sample size to 13.

The best strategy is to bring in some set of users, find the problems they have, fix those problems, then bring in another set of users as part of an iterative design and test strategy. In the end, although you're never testing more than 5 users at a time, in total you might test 15 or 20 users. In fact, this is what Nielsen recommends in his article, not just testing 5 users in total.

You can use the online calculator here to find the needed sample size:

http://www.measuringusability.com/problem_discovery.php

To see how this is calculated see the article at:

http://www.measuringusability.com/qualitative_sidebar.htm

And Jim Lewis has a comprehensive and accessible article called: "Sample sizes for usability tests: mostly math, not magic" which can be found at the ACM Portal Site (Lewis 2006).

27. What is the minimum sample size I should plan for?

For finding and fixing problems (so called formative testing) the best strategy is to test just a few users (like 5), fix the problems they encounter then in the next iteration test another 5. For this type of testing goal, testing five users in three or four iterations (15-20

users total) is reasonable knowing you're detecting most problems that occur most frequently.

When you are trying to describe the usability of an application by describing how long tasks take and how many users complete a task (so called summative testing), the five user sample size doesn't apply. In these situations you need to compute your sample size based on your goal of either comparing applications or finding a precise measure. Both of which require a sample size calculation.

Despite all the sophisticated methods for finding the right sample size, people still want some minimum number they can budget and plan for. If I had to give you one number as a minimum to plan on when getting an estimate of precision, I'd say 10 users. There are basically three reasons why 10 is a good **minimum** sample size.

1. It has been shown that you can get reliable results using the SUS questionnaire with between 10-12 users.
2. You can have one task failure (9/10) and still be 90% confident at least 70% of the users can complete the task.
3. There's something psychologically better when you test double-digits (10 vs. 9) for product teams, marketing managers or anyone who's taken a statistics class and bristles at the thought of small samples.

See Tullis and Stetson (2004) for more information.

The calculations for showing at least 70% of users can complete a task if 9/10 did in a sample can be found in the Statistics Packages for Usability Testing

- Basic : http://www.measuringusability.com/products/statsPak
- Expanded: http://www.measuringusability.com/products/expandedStats

This section provided a minimum sample size. It is possible to generate a more precise number by using one of the sample size and power calculators from above and the method explained in the next section "How do I determine my sample size?"

28. How do I determine my sample size?

Your sample size depends on a number of factors. When you hear you only need to test with 5 users, this applies to finding usability problems in an interface given a problem affects about 31% of your users. While the approach for determining 5 users for frequent problems is statistically valid, it's not what marketers, survey designers or statisticians use when computing sample size. To determine your sample size in this more traditional sense you need to answer two questions.

1. Are you comparing products?

2. How confident do you need to be?

The first factor is whether you are comparing products or just testing to get an idea of the performance (as measured in task times and completion rates). When you compare two or more products you need to take into account the following:

1. **Effect Size** : The amount of the difference that is meaningful
2. **Confidence Level** : How confident you need to be you're not saying there is a difference in products when one doesn't really exist.
3. **Power:** How confident you need to be to not say there is NOT a difference when one doesn't exist.

The formula and nuance of computing sample sizes for comparing products is complex and will be covered in a subsequent report. In the interim you can use the Expanded Stats calculator which will calculate the sample size based on some common defaults. http://www.measuringusability.com/products/expandedStats.

For stand-alone usability tests (no comparisons), your sample size is determined based on how precise you want your estimates of the true completion rate and true average task time. Essentially you want to find out what sample size you need to have a desired margin of error. The margin of error is just half the width of a confidence interval. In my experience most teams have an idea of approximately how many users they can test (some can only test up to 15; others are willing to go to 40 or 50).

So the first thing you need to do is identify your desired margin of error. For example, is 10% precision enough or is 5% necessary? One way to find out is to have a discussion with the stakeholders of a test and use the following approach:

"If I told you the average time to complete a task was 100 seconds, would we think differently about the usability of the task if in fact it really took 110 seconds? How about if we think it takes 100 seconds but it really takes 120 seconds or 130 seconds?"

These just reflect margin of errors of 10, 20 and 30%. The point at which you or the stakeholders say there would be a perceived difference in usability is where you can set your desired margin of error. So if you, or a product team, decide that anything under +/- 20% you can live with, then you determine your sample size based on this margin of error. The Expanded Usability Statistics package will calculate the sample size needed based on your desired margin of error.
http://www.measuringusability.com/products/expandedStats

You can also use the 20/20 rule of sample sizes which is based on data from over 120 usability tests. At a sample size of 20 users, you will typically have a margin of error of approximately 20%. If you want half that margin of error you need to quadruple your sample size. So to obtain a margin of error of 10% you'd need to plan on testing around 80 users. The reason is the relationship between the uncertainly in our estimate is

reduced by the square root of the sample size. See http://www.measuringusability.com/test-margin.php for more.

Here is an approximate guide for how large your margin of error will be around task-times and completion rates:

Sample Size	Margin of Error (+/-)
5	58%
10	36%
15	28%
20	20%

Table 2: Approximate margin of error for task times based on usability data from 120 tests.

A sample size calculator to compute the desired margin of error for a single usability test or for comparing two products is available in the Expanded Statistics Package for Usability Testing http://www.measuringusability.com/products/expandedStats

Metrics

29. How do you measure UI problems?

It doesn't take much effort to list the problems you find users encountering. These problem lists can be sent directly to development or entered into bug-tracking systems. In addition to identifying a description of the problem for a development team, it is good to also describe any errors users encountered which you attribute to the UI problem. It is also important to track which users encounter which problem. This way you are able to both estimate the frequency of problems and identify problems which only 1 user encountered. This information can be used to determine the average probability of encountering a UI problem, which in turn can be used to calculate the improved usability of designs, ROI for usability dollars and sample size calculations for planning tests. In addition to recording the description of the problem and severity, you should also create a user by problem matrix such as the following:

	User 1	User 2	User 3	User 4	User 5	User 6	Total	
Problem 1	X	X			X	X	4	.67
Problem 2	X						1	.167
Problem 3	X	X	X	X	X	X	6	1
Problem 4				X	X		2	.33
Problem 5					X		1	.167
Total	3	2	1	2	4	2	**14**	**p = .47**

Table 3: Example UI Problem Matrix. The X's represent users that encountered a problem. For example, User 4 encountered problems 3 and 4.

Given 14 total problems and 2 unique problems the probability a user will encounter a problem with this interface is 47%.

Technical note: The 47% almost certainly overestimates the actual problem discovery rate and requires some adjustments. It is a complicated and controversial topic and the interested reader can see (Lewis 2006).

30. If 1 user encounters a problem will it impact 1% or 20% of all users?

Let's imagine you are testing five users as part of an iterative testing approach to find and fix problems. During the test only one user encounters a particular problem. To fix this particular problem might take a lot of effort. A question that comes up is how we know whether this problem is more likely to affect 1 out of 5 users (20%) or 1 out of 100 users (1%).

There is uncertainly with small sample sizes, but for this particular question we can use probability and statistics to help decide. Doing so is what makes a usability evaluation quantitative. First, a 95% Confidence Interval around this problem tells us we can be 95% sure the problem will affect between 2% and 64% of users (see the sections on Computing Confidence Intervals). So right away we know that there's less than a 5%

chance the problem affects 1% or less (since 1% it is lower than the 2% boundary of the 95% confidence interval).

We can use the Binomial Probability Density function to compare the probability 1 out of 5 users encountering a problem comes from actual occurrences in the user-population of 20% versus 1%. The probability the actual occurrence of the problem is 20% is about 41%. The probability the actual occurrence is 1% is 4.8%. Therefore, it is about 8.5 times more likely the problem affects 20% of the users than 1% of users. You can use the Excel formula =BINOMDIST(1,5,0.2,FALSE) and =BINOMDIST(1,5,0.01,FALSE) to get these values.

Without doing any math you can think of it conceptually. With small sample sizes around 5 users, you are likely to observe frequently occurring problems and miss the infrequent ones. If the problem didn't occur frequently then you probably wouldn't have seen it in a test. Of course there is a chance that you just happened to see a rare problem with five users, but it is not very likely.

31. How do you determine task success or failure?

Prior to testing your users you should identify the success criteria for a task – what fields need to be filled out, what information must be contained, what steps are essential. If any of the critical steps for task success are left off then the user fails the task.

For example, a task might ask a user to rent a mid-size car from Boston on July 15th 2010 at 11am for 2 days. If the user enters the wrong pick-up day then this is a task failure. If the user rents a compact car it is a task failure. If the user adds the Loss Damage Waiver Protection even though it wasn't requested then this is not a task failure.

32. How do you code task completion rates?

Code task success as a 1 for successfully completing a task and a 0 for unsuccessfully completing it. You can then compute the percent who completed the task e.g., 9/10 = 90% and compute a confidence interval around the proportion.

Some researchers code task completion using partial completion rates reflecting partial task success (e.g. 25%, 50%, 75%). I prefer to keep task success as a more intuitive pass and fail to reduce some of the ambiguity in assigning a task as say 50% or 75% complete. Set your task success criteria ahead of time. If the user completes all required parts of the task criteria then it is a task success. If users leave off non-critical information but still

meet all required criteria, then code these omissions as errors (I also advocate recording errors see below).

33. When do you start and stop the time?

You want to have a task time that is as close as possible to what it might take a user in real life. The lab setting and test scenarios create a number of artificial nuisances. Wait for the user to read the task scenario and understand what they need to do. I start the time as soon as the user "orients" toward the screen. By "orients" I mean the user looks at the screen and begins to take an action. I don't record the time it takes users to read the task, since this is an artificial element of the testing situation. When tasks are long and less familiar to a general audience, it matters less when you start the time, but be consistent. On the other hand, when tasks are shorter and have meaningful benchmarks, every second counts. For example, can users rent a car in less than 60 seconds? If it takes a user 20 seconds to read a task scenario, then this would eat into your test criteria if you counted it. Would the product team or client think that legitimate (especially if you come to them and say the average time is 75 seconds?).

Ideally your task-scenario ends with a clear ending point (user clicks the submit button) which removes much of the subjectivity in determining when the task ends. While it

might make sense to have the user tell you when they're finished, in my experience they forget to tell you for many tasks.

34. What is an acceptable task completion rate?

I get asked a lot what minimum percent of users should be able to complete a task. The unfortunate answer is that it depends (you saw that coming). If the costs for task failure are high (loss of money, loss of life, abandoning the website) then you need to aim for100%. If the consequences are less impactful, then you can relax the rate a bit. I've seen walk-up-and-use consumer web-applications with a 70% completion rate. That is, if there is evidence from testing that fewer than 70% can complete a task without training on their first attempt, the rate is unacceptable. If your users complete tasks daily on an application, then this level of completion is simply unacceptable and something closer to 100% should be expected. As a rough guide you can use the following table (based on 1200 task completions from over 120 usability tests) to see where your task completion rate falls.

Percentile	Completion Rate
98	100%
80	97%
70	90%
60	83%
50	78%
40	67%
30	56%
20	42%
10	22%

Table 4 : Percent of tasks (Percentile) and their corresponding completion rates. For example, about half the task have greater than a 78% completion rate.

The table above shows there is a heavy skew toward high completion rates. A full 20% of the sample tasks have a 100% completion rate (that's why the second row of the table above shows 80%). This is likely a consequence of the dataset which contains many fully released products. For example, if you observe a 90% completion rate for a task, then that task is at the 70th percentile—meaning it has a higher completion rate than 70% of the tasks in the dataset. A completion rate below 56% would put the task below the 30th percentile and have one of the lower task completion rates.

35. What is an error?

When users interact with an application, mistakes happen and unintended actions are inevitable. It is important to have some idea about what these are, how frequently they occur, and how severe their impact is. At the very least, just record what unintended action occurred (e.g., user entered the client's whole name in the first name field). Many errors are preventable through better designs. Some errors are inevitably going to happen because we humans just make mistakes. Even these are important to see. A usable interface is one that, to as great an extent as possible, prevents errors and, when errors occur, helps users recover from them with as little pain as possible.

36. Do I need to record errors?

Errors provide great diagnostic information and explain longer task times and lower completion rates. One drawback to errors is they can require more effort to collect and observers will not always agree on whether an event really was an error (and agree even less on what the correct severity is). In my experience even simple recordings of errors provide insights into the effects of UI problems. Errors also can explain much of the causes for longer task times. Without a record of errors, you just see a longer task time and wonder why a user might have taken longer. In fact, errors correlate highly with task completion time and account for around 25% of the differences in user times (see Sauro & Lewis 2009).

Errors also allow you to capture those omissions and commissions which almost lead to task failures. They are especially important when you code task completion rates as binary (which I recommend).

37. How do you record errors?

The simplest thing to do is just note that a user committed some unintended action and a brief description of what it was. Note each time the unintended action occurs, even if a user commits the same error multiple times. For example, something like "user forgot to select GPS navigation checkbox before submitting". When you analyze error data you will then have a count of these unintended actions per task per user.

38. How do you analyze errors?

Errors, unlike task completion rates, can occur more than once per user per task. This can complicate the analysis since you cannot easily compute a proportion like in task completion rates. For example if a user committed 3 errors for 1 task you cannot just divide 3/1. The simplest thing to do is treat errors as binary and code the raw error counts as either 1's (user committed at least 1 error) or 0's (user committed no errors). This loses some information but for many tasks and applications which don't see many errors, this may be sufficient.

An alternative approach which retains all the information is to convert errors into a proportion based on the opportunity for errors. An opportunity for an error is a technique I borrowed from Six Sigma's opportunity for a defect. The idea is that some tasks, especially those that are longer or more complex, will have more opportunities for users to make mistakes. See also HEP Human Error Probability for a similar idea Park (1997). For example, withdrawing money out of an ATM will have fewer error opportunities than submitting an expense report with 4 receipts and mileage. You identify the places in an interface where users can make mistakes and divide the total number of errors across all users by the total number of opportunities. For example, if a task has 5 opportunities for an error and 10 users attempt the task there are 50 opportunities. If you observe 5 errors across the users the error rate is 5/50 = 10%. See the section on defining error opportunities.

39. What is an acceptable average number of errors per task?

Just like task completion rates, the number of errors that is considered acceptable depends on the context. If loss of life or lots of money is at risk, then very few errors should be tolerable. Errors are inevitable as humans make mistakes in interacting with interfaces (the H in the HCI). I have a large dataset of usability data from dozens of companies and over 120 usability tests. 719 tasks contain error data. The dataset contains a combination

of consumer and business software, with mostly productivity tasks such as entering an expense report, adding a contact, dialing a phone number on a cell-phone. Below is a distribution of the average number of errors per task.

Percentile	Avg. # of Errors Per Task
90-99	0
80	.18
70	.31
60	.48
50	.66
40	.93
30	1.2
20	1.6
0-10	2.4+

Table 5: Distribution of raw errors from 719 usability tasks. For example, a task with an average of .18 errors is better than 80% of all the tasks in the dataset.

For example, only 10% of all tasks observed have no errors (making them tied for between the 90-99th percentile) and half the tasks have on average .66 errors per task. If you observed 12 total errors in one task from 10 users you would have 1.2 average errors and this would put the task at the 30th percentile. That means only about 30% of tasks

have more errors and 70% have fewer. So when you compute the average number of errors and want to know if it is too high, use the above chart and determine what percentile is too high. At the very least you want to be in the top half (i.e. less than .66 errors per task). Anything more than 2.4 errors per task is both unusable and unusual.

When you communicate the results in a report you can use this table to state that the average number of errors is either above or below average or cite the approximate percentile—for example the average number of errors of .31 for the task is better than 70% of tasks from a large database of usability data.

40. Do I need to convert errors into an error rate?

You will need to convert errors into an error rate if you want to combine the errors, along with task completion rates, time and satisfaction into an overall usability score. Since the error rate is a percentage (out of total error opportunities) you can combine it with the other percentages to generate a single usability metric (see the section below on "Is there a single usability score that I can report"). Average errors by themselves are fine to report, especially when put into a context of how many errors are too many. In the absence of data for the particular task or domain, using the table from the section on what an acceptable number of errors can help provide context.

41. How do you define an error opportunity?

Errors can be analyzed by counting the incidences of unintended actions. To measure how error prone a task is, divide the total number of errors across users by the total number of opportunities for an error. An error opportunity is a designation, borrowed from six-sigma, which identifies parts of the interaction where mistakes can happen. For example, if a user needs to enter an email address, enter a password and click a checkbox to agree to terms and conditions, then there are three opportunities for errors. Error opportunities are not paths through an interface or steps, just places where things can go wrong. Multiple types of errors can occur in the same error opportunity (e.g., forgetting to enter a password or entering a last name in the password field). There is some subjectivity when defining error opportunities because it isn't always clear when one opportunity ends or begins. If you define error opportunities too granularly, then you can deflate the true nature of the occurrence of errors. Conversely, if you define your error opportunities too broadly you can inflate the amount of error in your task. If possible, have two evaluators independently identify the opportunities for errors by task and reconcile any differences. If you're ambitious, compute some measure of agreement (e.g Kappa) before reconciling the differences so the level of disagreement can be seen for tasks. When someone questions the number of opportunities and you have a high level of agreement between independent evaluators, this can help justify your selection.

Once the opportunities have been identified, divide the total number of errors per task from all users by the total opportunities per error for all users. For example, if there were 3 errors observed from 5 users and 6 opportunities for error, you can use the proportion (3*5)/(6*5) = .10 (or report as a percentage).

42. What is an acceptable minimal error rate per task?

If we convert raw errors to a percentage by dividing them by the total number of error opportunities, we can generate an error rate. As with most usability metrics, it is difficult to define a benchmark that is applicable to all contexts (or even many contexts). Like task completion rates, you want to have fewer errors and a lower error rate when the consequences of these errors are high (loss of money, loss of life). Errors are inevitable so a certain percentage are going to occur just due to the fallible nature of human interactions.

For a rough guide on error rates, of the tasks I've analyzed with errors and error opportunities defined, the following distribution of error rates can be seen:

Percentile	Error Rate
99%	9%
75%	13%
50%	18%
25%	31%

Table 6: Distribution of errors from 50 usability tasks. For example a task with a 13% error rate is better than 75% of the tasks in the dataset.

For example, if your total number of error opportunities is 50 (10 tasks and 5 users) and you observed five total errors, then your error rate is 5/50 10% and would fall between the 50th and 75th percentile of tasks. That means the amount of errors relative to opportunities is better than 50-75% of the tasks in this dataset. The dataset contains a combination of consumer and business software, with mostly productivity tasks such as entering an expense report, adding a contact, dialing a phone number on a cell-phone. These should be used as only a rough guide as they come from a dataset of 50 tasks.

43. How long should a task take?

Once you've measured task time you will inevitably want to know if your average time is good enough or taking the users too long. Unlike completion rates and errors, task times are difficult to compare across tasks. Thirty seconds might be fast for one task but too long for another. To determine how long your task should be taking you should look for external benchmarks and criteria that are meaningful to a user. Such criteria are often referred to as specification limits. Some tasks have more intuitive specification limits. For example, it probably should take less than 1 minute for someone to withdrawal money from an ATM. Specification limits can be set from:

- previous versions of the application
- competing products
- an industry standard
- some multiple of a skilled users time

Any specification is better than none and will go a long way to providing context to a task time. You can then generate a confidence interval around the average time. If the upper boundary of the confidence interval is lower than your task time than you can conclude with statistical confidence that you've met or exceeded the specification limit.

For more ideas on setting specification limits for task times see Sauro & Kindlund 2005 "How long Should a Task Take? Identifying Specification Limits for Task Times in Usability Tests"

44. Is there a single usability score that I can report?

Product teams, management and those unfamiliar with usability will typically ask for an overall usability metric—some defining measure that encapsulates what we mean when we say usability. Often such metrics get used as part of a dashboard on department performance. There is no usability thermometer; instead usability is a combination of several metrics. Once you collect your metrics (task time, completion rates, satisfaction scores, errors etc) you run into the problem of how to summarize multiple metrics. To compound things, the metrics are on different scales.

It has been shown that there is evidence for an underlying construct of usability and the different metrics collected during a usability test correlate. Therefore, combining the metrics into an overall average will provide a description of usability.

There are different approaches you can take to combining metrics into a single score. The approach I recommend is to standardize each metric into a z-score, then convert that z-score to a percentage and take the average of those percentages as a singe usability score.

When you average, time, errors, task-level satisfaction and completion rates into a single score it is called SUM (Single Usability Metric). See the section below for more information on SUM.

45. What is SUM?

SUM stands for Single Usability Metric. SUM is an average of the four common metrics taken during a usability test: completion rates, time, errors and satisfaction scores. These raw metrics are standardized by converting them into percentages (from z-scores for satisfaction and time data) and averaged together. The benefit of SUM, or other aggregate scores, is to provide a summary picture of usability. Since there is no single usability thermometer that can tell us how usable an application is, we rely on these core metrics to describe the effectiveness, efficiency and satisfaction with an interface. These measures correlate at the task level and provide evidence of an underlying construct of usability (see Sauro & Lewis 2009). There are times when it is easier to describe the usability of a system or task by combining the metrics into SUM, for example, when comparing competing products.

SUM and other composite scores do not replace the underlying metrics, which are still available for inspection and interpretation. Instead they are like an abstract is to a full research paper. The abstract does not replace the full paper, so you do not actually lose information from summarizing. See

www.MeasuringUsability.com

http://www.measuringusability.com/SUM/index.htm for a free downloadable excel file and scorecard which will compute SUM from raw usability data.

46. How do you measure learnability?

The terms learnability and usability often get used together and interchangeably. Often usability is defined as some intersection between efficiency, effectiveness and satisfaction. See the earlier sections on how to measure usability. People often say they want to measure learnability, but like usability, there is no learnability thermometer you can use to asses how easy an interface is to learn. Instead, learnability can be thought of as usability over time. That is, you measure learnability using the same metrics you do for usability but you need at a minimum 2 points in time to compare the differences.

The most common way to measure learnability is to use time-on-task and have users repeat tasks either during the same testing session or at some point in the future. The latter is a longitudinal usability study.

When I've measured learnability, I've had users attempt tasks repeatedly in the same testing session. At a minimum you need to have users attempt tasks twice, but ideally, three to five times are better. Changing some minor details of a task (such as the names, dates or numbers) helps ensure users aren't just mindlessly completing a task from memory (See the section Should users repeat tasks or only attempt them once?).

With each task attempt, users usually complete the tasks faster, with the biggest improvement between the initial and second task attempt. Improvement in performance then asymptotes out after so many attempts (approaches no additional improvement). Graphing the performance over time provides the famous learning curve (which is an inverse logarithmic relationship -- see also http://en.wikipedia.org/wiki/Learning_curve).

As with usability, it is difficult to know if an application is "learnable" enough without comparing it to something else. For that reason, every time I measure learnability it is in the context of a comparative usability study. You are then able to graph the learning curves for two applications and see which application has a steeper curve (better learning). You can also see if one application is not as quick to learn, but shows the best performance after, say, three trials. In practice, time is tight in usability tests so I often only have time to measure two task attempts.

As with usability metrics, there are statistical ways to determine if one application is more learnable (e.g., comparing the slopes of the learning curves). Below is an example of the same task attempted twice on two competing products.

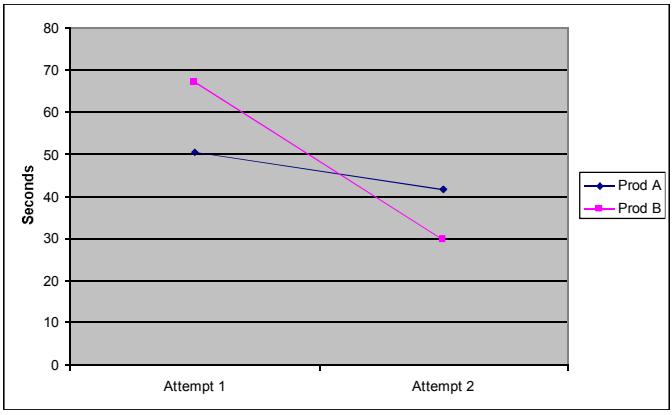

Figure 5 : Learning curves for two products. The crossing lines show Product B to have a better than expected learning curve.

You can see that Product A took less time on the initial task attempt but Product B took less time on the second attempt. With only two task attempts it is inconclusive whether users will take less time after *many* task attempts on A or B. Product A is likely faster on the initial use and Product B might be faster after some initial exposure. You can see why it is valuable to have more than two attempts to measure learnability.

47. Should you assist users during a usability test?

We try to create a realistic scenario in a lab to mimic real-life use. Since users won't have omni-present facilitators available to help them should they get stuck, any intervention

makes the data less realistic. With that said, testing a user in a lab is not natural and never fully will be. Inevitably a user will misinterpret a task or get stuck going down a wrong path and they're unlikely to get out. When these or other situations occur the test facilitator needs to intervene to either end the task or get the user back on track. Since the days of Stanley Milgram's shock trials, all researchers are more sensitive (and often legally bound) to limit the discomfort of the test participant. Pragmatically, it doesn't make much sense to burn precious time and video with users on activities that don't tell you much. When such interventions occur they should be noted as an assist. The major reason to assist a user during a task is so you can still have them attempt other aspects of a task. This is especially the case when the primary testing goal is finding and fixing problems.

48. If you assist a user during a task, is it a task failure?

You should do your best to limit your assistance with users but these interventions inevitably occur. When they do, record it as an assist. I have a category in my data-logging sheets for assists and I note if I assist and how many times I assist a user during a task. As a general rule, if I assist, it is a task failure, unless the assist was short and due to something like a technical problem with the software.

Recording assists and task completions as separate metrics allows you to report assisted and unassisted task completion rates. In my experience it is best to report just one

completion rate, and that's the unassisted rate. You then code any task success attributable to your intervention as a failure. If you're finding yourself assisting more than a couple times during a test session then you might consider adjusting your testing setup to minimize such interventions.

Questionnaires

49. Do I administer a questionnaire after each task or at the end of the test?

If possible, you should do both. Research has shown that the questions asked immediately after a task measure something a bit different than the questions asked after the entire testing session (see Sauro & Lewis 2009). There is an overlap in what these measure, but the metrics you get immediately after a task correlate more highly with task performance. Essentially a user attempts a task and they have no problem rating the experience on a Likert scale. If they had problems they rate a low score, if it was easy they typically rate high scores.

Research has shown that there isn't a perfect correlation between satisfaction and performance, with around 14% of users rating high satisfaction scores after failing a task (see http://www.measuringusability.com/failed-sat.php for more).

Post-task ratings of usability allow you to quickly identify problem areas in an interface. With post-test questions, users tend to respond about their experience with the application as a whole. When you get low ratings on these questionnaires it is difficult for you or a product team to know what to do to fix it. Some questionnaires have sub-categories like "messages and errors," but it still doesn't tell you what in the application should be fixed—just overall impressions. While the usability test will affect users' responses, it

isn't clear to what degree the experiences and tasks encountered during the test affect the response.

In spite of their limitations, I find that it is important to get impressions of the usability of an application by asking post-test questions and post-task questions. The post-test data is helpful in identifying perception problems and can be used for marketing. Post-task questions allow you to zero-in on problems very quickly. If a task had low satisfaction ratings, then the functionality encountered during this task should be addressed. Usually the reason for not collecting post-task questionnaires is that there is a perception that they take too long to administer. Time is already tight during a usability test, so this concern is understandable. However, it has been shown that just asking one question about the task experience can yield reliable results (see Sauro & Dumas 2009). A single question only adds mere seconds to each task and a few minutes to the whole usability test. The data you get is well worth it.

See the section on post-task questionnaires and post-test questionnaires for recommendations.

50. What post-test satisfaction questionnaires do you recommend?

If you have a budget of at least $1200 then there is some benefit in using one of the commercially available questionnaires such as SUMI for standard software applications

and WAMMI for website (see http://sumi.ucc.ie/ for more information). The benefit of using one of these commercial packages is that they come with a norm-referenced database of scores based on other software applications or websites. The score they generate is then based on the relative standing of your application (or website) relative to a database of the other hundred or thousand tested. You will then have scores that will tell you, "your application is in the top 10% of all applications tested".

If you don't have enough in a budget, most people use SUS (see Brooke 1996) or PSSUQ (see Lewis 2002) for both regular software and websites. The SUS (System Usability Scale) has been around since 1986 and was designed for use with software applications. Because it wasn't designed specifically for websites, there is some controversy over whether SUS provides a meaningful metric for websites.

Both SUS and PSSUQ are well designed questionnaires (meaning they have desirable statistical properties) but lack the norms to tell you how to interpret your score. PSSUQ does contain some basic norms. If you have an internal set of scores on SUS or PSSUQ, then you can benchmark against those. Otherwise it is hard to interpret your score. See the section on acceptable SUS scores.

51. What is an acceptable SUS Score?

The advantage of commercially available standardized questionnaires such as SUMI, QUIS and WAMMI is that you get scores relative to other applications and websites tested. Questionnaires like SUS and PSSUQ, while psychometrically valid, lack such normative databases. In the last few years, a number of publicly available sources of data have been made available to help interpret raw SUS scores. Tom Tullis (Tullis & Albert 2009) found that the average score of many websites and applications is 66. Sauro and Lewis (2009) found an average score of 62 for business based applications. Bangor et al., 2008 found an average score of 70, which varies a bit by the application tested—including websites, cell-phones and IVRs.

As a rough guide, this tells you that a score below about 66 (a weighted average of the three sources cited above, sd=19) suggests the application is below average in perceived usability, and a score above 66 is above average.

The following table provides approximate percentiles to help interpret your score:

Percentile	Raw SUS Score
90	91
75	80
50	66
25	53
10	41

Table 7: Distribution of SUS scores from hundreds of usability tests. For example a raw SUS score of 80 would be better than 75% of all applications tested.

For example, an average SUS score from say 14 users for an application that is below a 41 would be among the worst applications tested, with over 90% scoring higher. Conversely, an SUS score above a 91 is among the best applications and websites tested (top 10%). *See the Guide to Using SUS on the Measuring Usability website for more information on using and interpreting SUS (Coming 2010).*

52. What post-task satisfaction questionnaires do you recommend?

Post-test satisfaction questionnaires tell you overall sentiments about an application but can be difficult to help diagnose problems. Collecting a user's sentiment immediately after a task provides more specific diagnostic information. The 3-question After Scenario Questionnaire "ASQ" developed by Jim Lewis (see Lewis 1991) is short, reliable and used the most. The three questions are:

1. Overall, I am satisfied with the ease of completing the tasks in this scenario.
2. Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario.
3. Overall, I am satisfied with the support information (on-line help, messages documentation) when completing the tasks?

Each question is rated on a 7-point scale where 1 is "strongly disagree" and 7 is "strongly agree." I often drop the third question about help and documentation when testing self-service applications that have no documentation help or error messages. Of course this means I can no longer compare 2-question results from one test to 3-question results from another test without dropping the 3rd question from all datasets.

Note: Based on the evidence currently available, Jim Lewis recently sent me a personal communication telling me that he plans to stop using the ASQ and to switch to the single 7-point Likert item (see below).

In general, asking more questions will reduce errors in your measurement. Time is often limited, however, so every second matters. So if you only have time to ask a single question, then research has shown that two single-item questions work as well as the ASQ. They are the SMEQ and a 7-point Likert question.

1. The **Subjective Mental Effort Questionnaire** "SMEQ" is an interval scaled single question. Users mark a line on a piece of paper or use a web-version to indicate how much mental effort a task took (see the figure below).

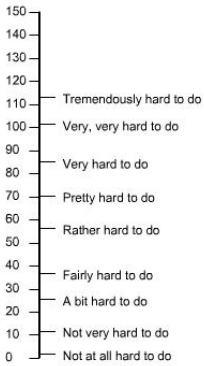

Figure 6: The Subjective Mental Effort Questionnaire (SMEQ)

2. **Single Question Likert**: Asking the simple question: "Overall, this task was:" with a 7-point scale from very easy to very difficult provides results as good as the ASQ and almost as good as the SMEQ (see the figure below).

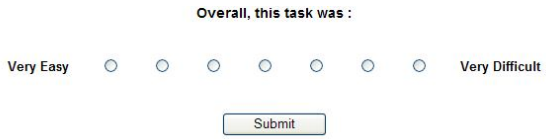

Figure 7: The Single Question Likert

While the SMEQ is a bit more sensitive than the 7-point Likert, it requires more administrative overhead. You have to either physically measure all the responses using a ruler or you need access to a Flash Application which allows users to electronically draw a line. Conversely, with the single question Likert, you can easily and quickly collect responses electronically or verbally.

For more information on both SMEQ and the single question Likert see:

Sauro, J. & Dumas J. (2009) "Comparison of Three One-Question, Post-Task Usability Questionnaires." in *Proceedings of the Conference in Human Factors in Computing*

Systems (CHI 2009) Boston, MA, available for download here:

http://www.measuringusability.com/papers/Sauro_Dumas_CHI2009.pdf

For more information on the ASQ see Lewis, J. R. (1993).

53. Can I use my own questionnaire or should I use a standardized one?

You can use your own questionnaire and many researchers do. Often companies have their own internal questions. An analysis of several internal "homegrown" questionnaires shows that they are generally less reliable than standardized ones (such as SUS, SUMI and PSSUQ) see Sauro & Lewis (2009) and Hornbæk & Law (2007). With lower reliability there is more error in your measurement, so, for example, your scores will fluctuate more from errors in interpretation than other tests with higher reliability.

If your company uses their own questionnaire and you have data collected, then you should continue using it. Having existing data allows you to compare current and future results to baseline data. One of the biggest problems with homegrown questionnaires is that you don't know what a good score is. With legacy data you can determine if your current designs are better than past ones, or at least better than the average scores you've seen. There might be more error from poorly worded questions or bad scale design, but

that is offset by having baseline data. What's more, the research from Sauro & Lewis 2009 on homegrown questionnaire reliability suggests that while they are not as reliable as the publically available standardizes ones, most are good enough to continue using.

If you don't have too many internal questions to ask users, consider phasing in one of the publically available standardized questionnaires.

54. How many scale points should a response item have: 3, 5, 7 or more?

When you have multiple questions that are rolled into one score the number of scale steps matters less. For example, the SUS has 5-point scales, the PSSUQ has 7-point scales and the SUMI has 3-point scales. However, the SUS, PSSUQ and SUMI have 10, 16 and 50 questions, providing many points of discrimination (50 for the SUS, 112 for the PSSUQ, and 150 for the SUMI).

When there are fewer questions from which to choose (say, under 10), the number of scale steps matters more. It is especially important with only one question, which is why you often see single-item questionnaires having 11-point scales. In general, with more scale steps, users can provide finer discrimination on the attribute being measured (e.g.,

satisfaction, usability). Having at least 7-scale steps should be considered the minimum number if you are asking one or a few questions.

If you have only a single question, then using up to 11 points will provide the best balance of allowing the user choice without confusion. Research into scale steps shows there is a diminishing return after 11 scale steps and especially after 20. For more information on scale steps see Nunnally (1978).

55. Does it matter if the number of scale steps is odd or even?

When you design your own rating scale, I often get asked whether the number of scale steps should be even or odd. With an odd number of scale steps (or points) there is a neutral point, with an even number there isn't. I encourage you to use an odd number. Although it might be nice to push some users off their neutral fence, forcing users into an "agree" or "disagree" position will likely just introduce more measurement error. The major standardized questionnaires all have odd numbers of scales steps (SUS, SUMI, PSSUQ, QUIS and ASQ).

56. Should I alternate positive and negative statements in my questionnaire?

You might have noticed that when you ask multiple questions, some users will rush through, not read the questions and just choose all 5's. This has a tendency to over-inflate the internal reliability, but as you can imagine decreases the validity of the results.

Some questionnaires, like the System Usability Scale and SUMI, alternate the wording of items so some are positive and some are negative. Consider questions 2 and 3 from SUS:

2. I found the system unnecessarily complex.
3. I thought the system was easy to use.

These questions are asking about the perceived usability, with one a negative statement and the other a positive statement. The idea is to force users to at least read the question and respond accordingly. It also allows you to provide different ways of asking the questions that might be easier to ask in the negative without requiring users to respond to double-negatives.

Some users may have a tendency to agree more quickly to statements than disagree. A disadvantage to alternating questions is that you now increase the chances of mistakes from forgetting to rate in the negative. I've seen users repeatedly make mistakes, some they catch, and others I catch from them intending to respond positively to a question that

was reversed. The increase in the cognitive load (having to remember if 5 and agree is a good or bad thing) also will increase the time it takes to respond. A few users have expressed dislike for having to go back and forth between positive and negative and even asked if I was intentionally trying to trick them.

The research is mixed on whether the gain from alternating positive and negative questions overcomes the increase in cognitive load and response mistakes. Unless you have good reason I recommend keeping your items all pointing in the same direction.

For more information see the excellent discussion of this issue in Lewis, J. R. (2002). Psychometric evaluation of the PSSUQ using data from five years of usability studies. International Journal of Human-Computer Interaction, 14, 463-488, available for download here:

http://drjim.0catch.com/PsychometricEvaluationOfThePSSUQ.pdf

57. Can I use statistical tests to analyze questionnaire data?

If you've taken a statistics class you might recall the classification of data into the hierarchy of nominal, ordinal, interval and ratio. This classification was devised in the 1940s to bring more scientific rigor to the practice of behavioral measurement.

- **Nominal data** are categories like pass and fail (completion rates) or types of errors
- **Ordinal data** are data that have an ordering – for example, the order in which horses finish in a race (with no information about the time it took).
- **Interval data** have an order and the distance between each value is equal—examples are things like weight, height and temperature measured in Fahrenheit or Celsius scales.
- **Ratio data** is just like interval except that there is a natural 0 value—for example task-time or temperature measured in the Kelvin scale.

With the publication of the classification came guidance on what statistical tests you can perform with each type of measurement. For example, it was proposed that data has to be at least interval scaled to compute a mean, standard deviation and all the popular statistical tests like the t-tests and ANOVA. So where do rating scale data from questionnaires fit into the hierarchy? Technically these are only ordinal, since we don't know if the distance between a 3 and a 4 is the same as between a 4 and a 5.

A number of publications have shown problems with this rigid hierarchy, and almost all behavioral researchers have no problem using rating scales (often called Likert-type data) in standard statistical tests. The one important thing to remember is not to make interval statements on ordinal data. Just because one satisfaction rating scale average is a 4 and the other is a 2, doesn't mean the users are exactly twice as satisfied. They might be, but it is unclear if a 4 really does translate into twice the level of satisfaction as a 2. You can, however, claim that the users who averaged 4 were consistently more satisfied than those who averaged 2, if the results were supported by a statistical test.

Analysis

58. Why & how do I compute a confidence interval around a completion rate?

The completion rate is a fundamental metric of usability. Just because you observe 9/10 users complete a task does not mean exactly 90% of all users will. To express the precision of your estimate you should compute a confidence interval. For small samples (less than about 150), the best formula to use to compute the interval is the Adjusted-Wald confidence interval. It uses the common statistical formula found in most introduction text books (called the Wald Formula named after Abraham Wald) but is "adjusted" in that it adds half of the squared Z-critical value to the numerator and the entire squared critical value to the denominator before computing the interval i.e $(x+z^2/2)/(n+z^2)$.

For example, for 95% confidence it is like adding two successes and two failures to whatever you observe. For the 95% confidence level you would use the Z-critical value of 1.96 (or if you're computing it by hand on the back of a napkin, use the approximate value of 2). If you observe 9 out of 10 users completing a task, this formula computes the proportion as $(9 + (1.96^2/2))/(10 + (1.96^2)) = 10.9/13.8$, which is approximately 11/14 or .789.

You then use this adjustment in the standard "Wald" formula presented in statistics text books:

$$\hat{p} \pm z_{(1-\frac{\alpha}{2})}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

- Where p is the adjustment – in the example above, it's .789.
- n is the adjusted sample size – in the example above, it's 13.8.
- $Z_{1-a/2}$ is the critical value from the normal distribution for the confidence level. For 95% confidence, it is 1.96.

.789 +/- 1.96* SQRT [(.789*.211)/13.8] = .215

.215 is the margin of error. Thus, for this example, the 95% confidence interval is.789-.215 = .574 to .789+.215 = 1.004. Because a proportion cannot be greater than 1, just cut off anything above 1, so the confidence interval ranges from .574-1.000.

If we were to test the entire population, we would expect the true completion rate to fall somewhere in the range of 57.4% and 100%.

For more information on the background of the formula and a free calculator, see: http://www.measuringusability.com/wald.htm

59. How do I compute a confidence interval around an average task time?

Knowing how long it takes users to complete a task can be a valuable metric for measuring the productivity of an application. Just because a sample of 10 users takes 100 seconds to complete a task does not mean the average of all users will be this long. Individual differences in task completion times are very large, so providing the precision of your estimate of the true task completion time is essential.

To complicate matters, task time data is positively skewed from some users taking much longer than others to complete a task. Confidence intervals assume the data to be approximately normally distributed, so this skew will affect the accuracy of the confidence interval. This isn't the case with task completion rates because there is a different formula for binary data (1 and 0).

To correct for the skew, transform the raw data using the natural logarithm (see the section "How should I report average task time?"). You can get the log of a task time by using the excel function =ln(). For example, a raw task time of 230 seconds becomes 5.438. Compute the confidence intervals using the t-distribution on the transformed data, then transform back into the original scale for reporting. Use the following formula to use t to compute confidence interval:

$$\bar{x} \pm \frac{s}{\sqrt{n}} * t$$

- Where x-bar is the mean of the log-task times
- The fraction s/SQRT(n) is the standard error of the mean.
- s is the standard deviation of the log task times
- n is the sample size
- t is the critical value from the t-distribution for the number of degrees of freedom and desired confidence level.

For example, here are the raw task times for a recent task on a rental car website:

215, 131, 260, 171, 187, 147, 74, 170, 131, 165, 347, 90.

First, we convert the raw times into log times using the Excel function =ln() to get the transformed times:

5.371, 4.875, 5.561, 5.142, 5.231, 4.990, 4.304, 5.136, 4.875, 5.106, 5.849, 4.500

The mean (x-bar) is 5.078, which can be found using the Excel function =AVERAGE(). If we transform this value back using the function =EXP(5.078), we have the geometric

mean of 174 seconds (see the section on reporting the average task time using the geometric mean).

Using the Excel formula =STDEV(), we determine that the standard deviation of the log times (s) is .423.

The sample size (n) is 12

The critical t-value for 95% confidence and 11 (n-1) degrees of freedom can be found using the Excel function =TINV(.05, 11). That critical value is 2.201.

Next, we get the standard error s/SQRT(n) = .423/SQRT(12) = .122. We multiply this times the critical t-value = .122*2.201 = .269, which is the margin of error. We add and subtract the margin of error to the mean to get the confidence interval = 5.078-.269 and 5.078+.269 = 4.809 to 5.347. Because it is difficult to interpret log time, we transform back to the original scale using the function =EXP() and report the 95% confidence interval as ranging between 123 and 210 seconds, with an average time of 174 seconds.

These calculations can be made using the Usability Statistics Packages available here:

http://www.measuringusability.com/products/statsPak (Basic)

http://www.measuringusability.com/products/expandedStats (Expanded)

60. How do I compute a confidence interval around a mean satisfaction score?

All averages from a sample are estimates of the true unknown average. It is important to understand and show the precision of the estimate using a confidence interval. To compute a confidence interval around the average satisfaction score, we use the following t-confidence interval formula (same as the one used for task times).

$$\bar{x} \pm \frac{s}{\sqrt{n}} * t$$

- Where x-bar is the mean of the satisfaction scores
- The fraction s/SQRT(n) is the standard error of the mean.
- s is the standard deviation of the satisfaction scores
- n is the sample size
- t is the critical value from the t-distribution for the number of degrees of freedom and confidence level.

For example, take the following 12 SUS scores from a recent usability test:

70.0, 90.0, 85.0, 80.0, 82.5, 92.5, 87.5, 80.0, 67.5, 62.5, 65.0, 92.5

The mean (x-bar) is 79.58 – in Excel, use =AVERAGE().

The standard deviation of the scores (s) is 10.81 – in Excel, use =STDEV().

The sample size (n) is 12.

The critical t-value is 2.201 – in Excel, use =TINV(.05, 11), which is for a 95% level of confidence and n-1 degrees of freedom.

Doing the math, we get the standard error s/SQRT(n) = 10.81/SQRT(12) = 3.12. We multiply this times the critical t-value = 3.12*2.201 = 6.87 which is the margin of error. We add and subtract the margin of error to the mean to get the confidence interval = 79.58-6.87 and 79.85+6.87. In summary, the mean is 79.58, with a 95% confidence interval ranging from 72.715 to 86.452.

These calculations can be made using the Usability Statistics Packages available here:
http://www.measuringusability.com/products/statsPak (Basic)
http://www.measuringusability.com/products/expandedStats (Expanded)

61. How do I compute a confidence interval around a UI Problem?

A usability problem that is coded as binary (a user encountered it or they didn't) can be modeled just like a completion rate. For example, if 2 out of 6 users encounter a problem in an application, an Adjusted-Wald Confidence interval tells us that we can be 95% confident between 9.3% and 70.4% of users will likely encounter the problem. For

computational details see the section: "How do I compute a confidence interval around a completion rate?" For small samples, the width of a confidence interval will be wide (in this case the width is over 60 percentage points). To reduce the uncertainly you can explore the option of conducting an unmoderated usability test or comparing the completion rate against a criteria (See the sections "Is there a difference in usability data from remote unmoderated tests and lab-based tests?" and Will at least 70% (or another percent) of all users complete a task?" respectively).

The confidence interval can also be made using the Usability Statistics Packages available here:

http://www.measuringusability.com/products/statsPak (Basic)

http://www.measuringusability.com/products/expandedStats (Expanded)

62. Should I put the confidence intervals in a report?

Once you compute a confidence interval, and especially on a small sample size, you will see very quickly that there is a lot of uncertainly in an estimate of a population mean time or completion rate. Showing a large amount of uncertainly might be the message you want to convey. In other cases you don't want to undermine your testing efforts by having product teams thinking there is so much uncertainty in an estimate that the conclusions are meaningless.

I put my confidence intervals in the same table where I summarize the means and standard deviations. You can also insert your confidence intervals into an appendix, which is a compromise approach. This way it doesn't overwhelm those who aren't used to seeing uncertainly quantified and it's there for those who want to see the boundaries of your intervals. I find a lot of success by graphically representing a confidence interval with a short explanation in an appendix. For example, see the confidence interval graphs in the section on the difference in data from remote unattended tests versus local lab-based tests (See the section "Is there a difference in usability data from remote unmoderated tests and lab-based tests?").

Another approach is to rephrase the information contained in a confidence interval in your report. For example, if four out of five users complete a task (80% completion rate) then the 95% confidence interval is between 36% and 98%. In your report you can say there's less than a 5% chance that less than 36 percent of users will be able to complete the task. Or more generally, it is unlikely that less than 36 percent of users will be able to complete the task.

63. What confidence level do I use?

Use 95% or 90% confidence levels. When in doubt use a 95% confidence level. The 95% confidence level is a matter of convention and is generally necessary for publication. In

applied settings it might be a bit too strict of a criterion and will generate intervals that are too wide. When that happens, many teams prefer the use of 90% confidence levels. These are a good compromise in applied settings between not generating too wide or too narrow of an interval. With that said, I find many readers wonder why 90% and not 95% since it is so prevalent. The other benefit of using a 90% confidence level is that when you want to make a 1-sided statement about a 2-sided confidence interval, you can make it at 95% confidence. For example, the 90% confidence interval around a 9/10 completion rate is between 64% and 99%. That means there is about a 5% chance the true completion rate is below 64% and a 5% chance the true completion rate is above 99% (5%+5%=10%). That's the way two-sided confidence intervals work—they put ½ the probability above and below the boundaries. Well, for things like completion rates, we're usually only concerned that a certain minimum can complete a task (e.g., 70%). From the 2-sided 90% confidence interval (64%-99%), we can then say we are 95% confidence the true completion rate is ABOVE 64%, versus saying we can be 95% confidence it is between these numbers.

I typically use 95% confidence levels for no other reason than it is the default level on the statistical packages I use. Using another confidence level would mean I'd have to remember to always adjust the level prior to making all the pretty graphs. I suppose defaulting to 95% is good practice so we're erring on the side of caution.

Why not an 80% confidence level? One reason is that there's rarely much practical difference between the size of 80% and 90% intervals, and rarely much difference between 90 and 95% -- which is a reasonable argument in favor of 95% -- in addition to it being the general stats package default.

Whatever you pick you should be consistent so you're not capitalizing on chance and your readers don't think you're trying to fool them. If you're not sure what the difference is between a confidence level and confidence interval, see the brief tutorial online: http://www.usablestats.com/tutorials/CI

64. Do my data need to be normally distributed?

If you've taken a statistics class (or know something about statistics) you probably have heard that you'll have problems if your data do not follow a bell-shaped (normal) distribution. The reason is that the math behind things like confidence intervals and most statistical tests are based on the assumption that your data are approximately normally distributed. Problems of non-normality are generally not that severe for most statistical tests, and there are ways of dealing with it. Because confidence intervals are covered here, I'll address that concern specifically. For completion rates around a binary task completion metric (1 or 0), we don't need to worry about normality since the adjusted Wald confidence interval doesn't expect normal data. It expects binary data, and that's

what we have (see the section on computing a confidence interval for task completion rates).

For confidence intervals around task times, we do need roughly normally distributed data. Since task times are often positively skewed from longer task times, there is a bit of an issue. To correct for it, transform the raw data using the natural logarithm, compute the confidence intervals, then transform the data back into the original scale (See the section "How do I compute a confidence interval around an average task time?"). Satisfaction data is usually roughly normal, so the confidence interval formulas don't need adjusting.

Another thing to keep in mind is that if you compute a confidence interval on data that are not normal, the consequence is that when you say you're 95% sure the true mean time is between, for example, 100 and 150 seconds, in reality the true probability of it being in that range might be closer to 85% or 75%, depending on how non-normal the data are. So you're safe if someone says "you can't use statistics or confidence intervals with non-normal data"—just use the adjustment for task-time confidence intervals described in this report.

65. What percent of all users can complete a task based on my sample?

When dealing with samples (and especially small samples) there is a lot of uncertainty in our estimate of the completion rate. Just because we observe 7 out of 10 users complete a task does NOT mean that 70% of all the users (the thousands or millions of them that use the software) will complete the task. If we were to observe another sample of 10 users, the completion rate will fluctuate due to chance variations.

Even with small samples, we can confidently conclude a completion rate exceeds a certain threshold. Instead of using the average, we use the lower boundary of a confidence interval. By computing an adjusted Wald confidence interval (see http://www.measuringusability.com/wald.htm for a calculator and formula), we can see with 7 out of 10 completing a task, we can be 95% sure the actual completion rate is between 39% and 90%. You can see the uncertainly I'm talking about—50 percentage points. If we observe 9 out of 10 we can be 95% confident the completion rate is between 57% and 99%, which is better. Using the lower boundary of the confidence interval allows you to be sure you are not overstating the estimate of the completion rate. But as you can see, the lower boundary is quite different from the impression that the raw number gives. Saying "9 out of 10 completed the task" sounds quite different from "I am 95% sure that the completion rate is at least 57%." This may give the developers the impression you are being unfair to them. You will have to decide which to report. If you

would like to know the probability a completion rate exceeds a threshold see the section "Will at least 70% (or another percent) of all users complete a task?"

66. Will at least 70% (or another percent) of all users complete a task?

Confidence intervals will show you the most likely range for the completion rate for all your users. For example, if 10 out of 12 users complete a task you have an observed 83% completion rate. If you were to test all your users, the completion rate will almost certainly not equal 83%. The best estimate is generated using a confidence interval. In this case, you can be 95% confident the actual completion rate is between 54% and 97%. If you wanted to know what the probability at least 70% can complete the task, you can use a statistical test called the 1-proportion test. There is a free calculator available at http://www.measuringusability.com/onep.php . We can then see that the probability that an observed 83% completion rate comes from a population greater than 70% is **83.11%.**

Depending on the context, an 83.11% probability might be compelling enough evidence to conclude the completion rate exceeds 70%. While 95% is used as a default level of confidence, especially when publishing research, in applied settings one balances the consequences for being wrong and the time and money it would take to be more confident. In many contexts where there is not loss of life or loss of money, a probability

in excess of 80% is compelling enough to move on to other areas which need improvements in usability.

A downloadable calculator for comparing a completion rate to a criterion is available in the Statistics Packages for Usability Testing

- Basic : http://www.measuringusability.com/products/statsPak
- Expanded: http://www.measuringusability.com/products/expandedStats

67. Will the average user task time be less than 2 minutes (or another time)?

Just as with completion rates, confidence intervals can also show you the most likely range for the average task time for users that complete a task. For example, if you observe an average task time of 120 seconds, this does not mean that if you were to test all users the average would be exactly 120 seconds. The best estimate is generated using a confidence interval after log-transforming the raw times (See the section "How do I compute a confidence interval around an average task time?"). The confidence interval is formed based on the amount of variability in the task times and the sample size. More variability and smaller samples mean wider intervals. A wide confidence interval means there is a lot of uncertainty in the estimate and can be discouraging.

For example, take the following task times: 132, 109, 82, 260, 130, 70, 172, 60, 107, 153, 89. The average task time (geometric mean) is 114 seconds and the 95% confidence interval is between 86 and 152 seconds. This interval is informative but represents a margin of error of almost 30%. To put the data in context we compare the average time to a criterion.

Lets say we know it took users 150 seconds to complete the task on the previous version of the application. Given the above times can we conclude that all users will be able to complete the task in less than 150 seconds on the new version?

To find out we perform a 1-sample t-test on the data to test against the criteria of 150 seconds. This test can be found in both Statistics Packages for Usability Testing

- Basic : http://www.measuringusability.com/products/statsPak
- Expanded: http://www.measuringusability.com/products/expandedStats

Given the sample size and variability from the data, the difference between the sample mean of 114 and test mean of 150 seconds (36 seconds) is more than what we'd expect from chance alone. In fact, we can be 97.1% sure all users will be able to complete the task in than 150 seconds.

Reporting

68. What should go into the report?

If your company has a report template, use that for your framework. If they don't, then take a look at the Common Industry Format (CIF).

http://zing.ncsl.nist.gov/iusr/documents/cifv1.1b.htm

A good report has the following sections:

1. A brief, no more than 2 page executive summary that contains the highlights: quantitative and qualitative of the data and the highlights of your conclusions or recommendations
2. Summary tables by metric with high level explanations of errors encountered or other high-level findings
3. Summary tables by task: Include metric summary and explanations about what happened in each task.
4. Metric table summary and good explanations around the metrics.
5. User Summary Table: Describe the experience and job-function of the users, gender, age and geography can be helpful.
6. A list of usability issues (critical for a formative report)
7. Conclusions about what the results show (especially for a summative report)
8. Appendix: Include the task scenarios, the post-task and post-test questionnaires that were used.

69. Should I use the Common Industry Format for reporting results?

The idea behind the Common Industry Format (CIF) is to have a standardized reporting template that makes it easer for customers to evaluate the usability of an application they are considering. If you are reporting your results to an internal product team then you don't need to use the CIF format. The CIF contains a good template for the type of information a report should contain, with the emphasis on the quantitative results of the tasks and description of the testing scenario. One draw-back to the CIF is that it does not provide for the presentation of errors or usability problems.

At a recent Usability Professionals Association workshop (CUE-8), 15 teams generated quantitative usability test reports of the Budget.com website. None of the teams used the CIF template, although almost all teams provided the information required by the CIF. See also the section on what should go into a report.

See http://zing.ncsl.nist.gov/iusr/documents/cifv1.1b.htm for the CIF; for the team's reports see http://www.dialogdesign.dk/CUE-8.htm

70. Do you include failed task times in the reported average task time?

You should record and retain task times for all users—even those who fail a task. When you report task times you should exclude the failed times from the calculation of the mean time. Report this time as the average task completion time. The failed times are important to retain because you can also use this information for additional reporting or diagnostic reasons. For example you can report average time to fail as well as an average time spent on the task (which includes completed and failed tasks). Another reason is that if you have to change your task success criteria and determine some users did indeed pass the task, you'll then want to include those times in the average time to complete.

71. How should I report average task time?

Task times in usability tests tend to be positively skewed. When you graph the times, instead of having a symmetrical bell-shape they will be positively skewed (having a longer right tail). The figure below shows example times from a usability task with the characteristic skew.

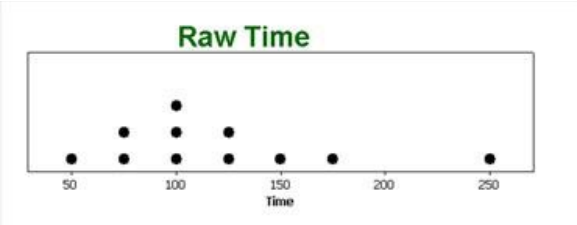

Figure 8 : Raw task times from a usability task. Task times show a positive skew.

This skew comes from users who take an unusually long time to complete the task, for example, users who have experienced more usability problems than other users. When data are roughly symmetric, the mean is the middle value. With a positive skew, however, the arithmetic mean becomes a relatively poor indicator of the center of a distribution due to the influence of the unusually long task times. Specifically, these long task times pull the mean to the right, so it tends to be greater than the middle task time. Some practitioners use the median which isn't influenced by longer times. The problem with the median at small sample sizes (under 25) is that the median tends to overstate the population median (the mid-time for all users) by as much as 10% and has more error. The best estimate is called the geometric mean. It sounds more complicated than it is. It requires log-transforming the raw times, finding the average, then transforming back. The calculator here will do it for you (http://www.measuringusability.com/time_intervals.php).

When you transform the values, the same data from above looks like the figure below. Notice how the values have become more symmetric and bell-shaped.

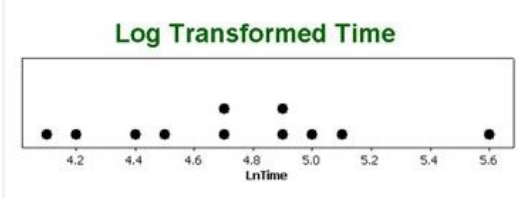

Figure 9 : Log-transformed task times from a usability task. The transformed data removes much of the skewness.

If you choose not to use the log transformation then the median is the next best thing, just be aware that you're probably overestimating the true time if your sample is less than 25. The other advantage to using the geometric mean is that it can be used for subsequent statistical analysis such as the t-test, ANOVA and in reporting confidence intervals which provide more precise results than the statistical procedures used on the median.

72. Can I report percentages on completion rates from a small sample?

When you have a small sample size such as 5 users attempting a task, some people have expressed concern about reporting 4/5 as an 80% completion rate. The rationale is that because the word percentage literally means "per 100," you can't really say 4/5 is 80%. While I understand the concern with overstating the precision of a small sample estimate, it is perfectly legitimate to report 4/5 as 80%. You don't need to have at least a sample of

100 to use percentages and there is no point at which the sample suddenly allows it to become permissible to use percentages. With that said, at small samples sizes the best estimate of the completion rate will almost certainly be wrong. That is why I strongly recommend providing a confidence interval around this percentage. If you want to be certain about a completion rate, then use the lower boundary of the confidence interval (see the section "Why & how do I compute a confidence interval around a completion rate?").

References

Bangor, A., Kortum, P. T., Miller, J. T. (2008) An Empirical Evaluation of the System Usability Scale. International Journal of Human-Computer Interaction. 24, 574—594

Brooke, J.: SUS (1996): *A "quick and dirty" usability scale*. In: Jordan, P. W., Thomas, B., Weerdmeester, B. A., McClelland (eds.) Usability Evaluation in Industry pp. 189--194. Taylor & Francis, London, UK (1996)

Card, Moran, & Newell, (1983) The Psychology of Human Computer Interaction

Hornbæk , K., & Law, E. (2007). Meta-analysis of correlations among usability measures. In *Proceedings of CHI 2007* (pp. 617-626). San Jose, CA: ACM.

Lewis, J. R. (1991). *Psychometric evaluation of an after- scenario questionnaire for computer usability studies: The ASQ.* SIGCHI Bulletin, 23 (1), *78-81.*
Available here for download:
http://drjim.0catch.com/PsychometricEvaluationOfAnAfter-ScenarioQuestionnaire.pdf

Lewis, J. R. (1993). Multipoint scales: Mean and median differences and observed significance levels. *International Journal of Human-Computer Interaction*, 5, 383-392.
Available here for download:
http://drjim.0catch.com/PsychometricEvaluationOfAnAfter-ScenarioQuestionnaire.pdf

Lewis, J. R. (2002). Psychometric evaluation of the PSSUQ using data from five years of usability studies. *International Journal of Human-Computer Interaction*, 14, 463-488.
Available for download here:
http://drjim.0catch.com/PsychometricEvaluationOfThePSSUQ.pdf

Lewis, J. R. 2006. Sample sizes for usability tests: mostly math, not magic. *Interactions* 13, 6 (Nov. 2006), 29-33. DOI= http://doi.acm.org/10.1145/1167948.1167973

Nielsen, J. (2000) Why You Only Need to Test with 5 Users,
http://www.useit.com/alertbox/20000319.html

Nunnally, Jum (1978) Psychometric Methods 2nd Edition

Park, Kyung S. (1997). Human Error. In Gavriel Salvendy (Ed.), The Handbook of Human Factors and Ergonomics, (2rd Edition). John Wiley & Sons.

Sauro, J. & Kindlund E. (2005) "How long Should a Task Take? Identifying Specification Limits for Task Times in Usability Tests" in Proceeding of the Human Computer Interaction International Conference (HCII 2005), Las Vegas, USA

Sauro, J. & Dumas J. (2009) Comparison of Three One-Question, Post-Task Usability Questionnaires. in *Proceedings of the Conference in Human Factors in Computing Systems (CHI 2009)* Boston, MA. Available for download here: http://www.measuringusability.com/papers/Sauro_Dumas_CHI2009.pdf

Sauro, J. (2009) Composite Operators for Keystroke Level Modeling. in *Proceedings of the Human Computer Interaction International Conference (HCII 2009), San Diego CA, USA*. Available for download here: http://www.measuringusability.com/papers/Sauro_HCII2009.pdf

Tullis, T. and Stetson, J. (2004) A Comparison of Questionnaires for Assessing Website Usability. Available for download here: http://home.comcast.net/~tomtullis/publications/UPA2004TullisStetson.pdf

Tullis, T., Albert R. (2008) Measuring the User Experience. *Morgan Kaufman*

About the Author

Jeff Sauro is founder of Measuring Usability LLC (www.MeasuringUsability.com), a quantitative research firm based in Denver, Colorado USA focusing on the statistical analysis of human behavior and quantifying the user experience.

Jeff is a Six-Sigma trained statistical analyst and pioneer in quantifying the user experience. He specializes in making statistical concepts understandable and actionable. He has consulted with companies such as eBay, Autodesk and Sage on usability and statistics issues and has worked for Oracle, Intuit and General Electric.

Jeff has published over fifteen peer-reviewed research articles and presents tutorials and papers regularly at the leading Human Computer Interaction conferences. He is currently writing a book for Morgan Kaufman tentatively titled: Practical Statistics for User Research.

Jeff received his Masters in Learning, Design and Technology from Stanford University with a concentration in statistical concepts. Prior to Stanford, he received his B.S. in Information Management & Technology and BS in Television, Radio and Film from Syracuse University. While at Syracuse he completed a two-year thesis study on web-usability.

Contact Jeff: jeff@measuringusability.com || Twitter: @MsrUsability