How Long Should a Task Take?
Identifying Specification Limits for Task Times in Usability Tests

Jeff Sauro

Oracle
Denver, CO USA
Jeff.Sauro@oracle.com

Erika Kindlund

Intuit
Mountain View CA, USA
Erika_Kindlund@intuit.com

Abstract
This paper describes a method for deriving the maximum acceptable task completion time (specification limit) when information from existing user data, competitive products or contextual inquiries isn't available. Spec limits can be "bootstrapped" from an existing set of usability data by using the times from the most satisfied users who completed the task.

1 Introduction

Having specific, quantifiable usability objectives is an important component of usability engineering [1]. When gathering task time as a measure of efficiency (a key component of usability [3]), an analyst should have an objective to test against, such as "Users will complete the task in less than 10 minutes." Such an objective can be thought of as a specification limit. Setting meaningful task time specification limits isn't an easy exercise. Most task scenarios are unique to a specific domain and it isn't obvious how long a task should take. For spec limits to be effective, they should be meaningful to users and not set arbitrarily. Guidelines in the usability literature provide some approaches for setting task time spec limits. Lewis [4] suggested the following:

1. The test designers examine the task and set the criteria.
2. Identifying the expert or fastest task time and setting the unacceptable condition to 1.5 times (or other multiple) this time for each task.
3. Criteria defined based on historical tests with the product.
4. An agreed upon point based on negotiations for all parties responsible for the product.

Whiteside et al [9] also listed the following ways to set worst case, planned and best case task times with respect to:

1. An existing system or previous version
2. Competitive systems (e.g. market share, acclaimed user interface)
3. Carrying out the task without use of a computer system
4. An absolute scale
5. Your own prototype
6. User's own earlier performance
7. Each component of a system separately
8. A successive split of the difference between best and worst values observed in user tests

Not having any data on user behavior or from competing products makes task times the most difficult specification limit to set [1 p. 196]. We wanted a way to derive a specification limit that was better than arbitrary but still took into account user behavior when existing data is not available.

2 Identifying the Specification Limit: Investigation and Findings

Asking a user how long a task should take is problematic and unreliable [1]. While users aren't good at being able to specify how long a task should take ahead of time, we wanted to see if retrospective accounts would be a good indication of dissatisfaction or satisfaction with task duration. The retrospective accounts would come from the sample being tested and, as such, are the basis for the "bootstrapping" [1] technique described below.

2.1 "Bootstrapping" a Specification Limit

We looked at six data sets with 2500 observations from summative usability evaluations to see if any patterns emerged with user task times and post-task satisfaction scores. Retrospective accounts of task duration were collected using a post-task questionnaire. After each task, all participants were asked to complete a questionnaire containing 5-point semantic distance scales with the end points labeled (e.g. 1:Very Difficult to 5:Very Easy). For the analysis we created a composite satisfaction score by averaging the responses from questions of overall ease, satisfaction and perceived task time (See Table 1). The three questions had high internal-reliability (coefficient alpha > .85). The average of the responses (instead of the response from only one question) provided a less error-prone score and one more descriptive of the users' perceived sense of usability [5 esp. p15].

Table 1: Post-task Questionnaire

How would you describe how difficult or easy it was to complete this task?				
Very Difficult				Very Easy
1	2	3	4	5
How satisfied are you with using this application to complete this task?				
Very Unsatisfied				Very Satisfied
1	2	3	4	5
How would you rate the amount of time it took to complete this task?				
Too Much Time				Very Little Time
1	2	3	4	5

We found a moderately strong and significant correlation between post-task satisfaction and observed time on task (r = -.488 p <.001). That is, as tasks take longer, post task satisfaction goes down.

We used this relationship between satisfaction and task time to identify the maximum acceptable task time. We started with all observations, removed users who failed a task and then converted raw task times into standardized task times (z-scores).[2] This relationship is plotted in Figure 1. The relationship is noticeable—as users complete the task faster, their satisfaction scores increase.

[1] Derived from the phrase "pulling oneself up by one's bootstraps" since we are building the spec limit from the very data which we will then apply the spec limit to. The same expression is used in Statistics to describe a technique of making inferences about a sample by repeatedly taking samples from the data.

[2] Converting the raw times to z-scores allows tasks of different lengths to be compared. If the times weren't converted, then tasks that take on average 30 seconds would distort tasks that take on average 6 minutes. For the latter task, a time of 3 minutes would be a faster time for that task, but not for the 30-second task. The z-scores were calculated by subtracting the raw time from the mean and dividing by the standard deviation by task. For more information see: http://www.measuringusability.com/zcalc.htm

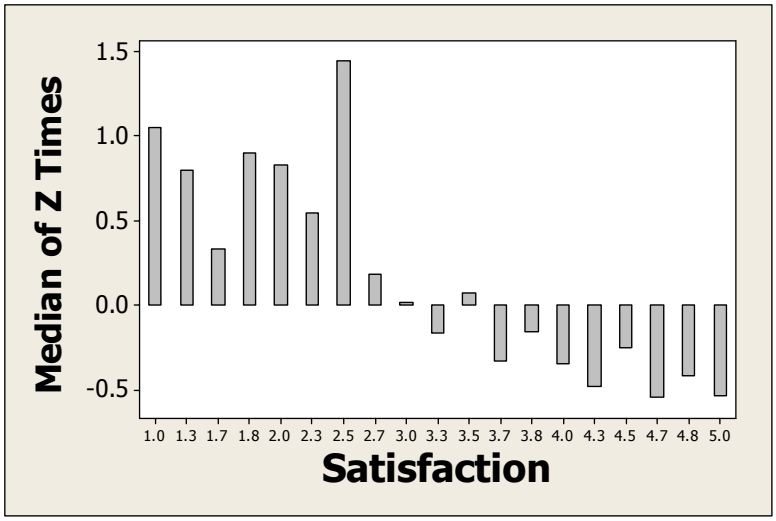

Figure 1: Median of standardized times (Z Times) for completed tasks only by Satisfaction scores. n = 1958

2.1.1 The Point of Increasing Satisfaction

One could rationalize that the specification limit should be set at the point at which some users begin showing a high degree of satisfaction with the task duration. From the Figure 1 it can be seen that at the satisfaction level of approximately 3.7 and above, tasks are being completed faster than average.

Nielsen and Levy conducted a meta-analysis of several products and analyzed user satisfaction data and found that users tended to rate systems they preferred a 4 or above (on 5 point scales) [3]." It would seem reasonable to use 4 as the planned value for subjective satisfaction in the goal setting phase of a usability-engineering process [7]." Our data also supports 4 as a reasonable breaking point for helping set the goal for task times.

To identify the maximum acceptable time, we excluded users that had less than a 4 in composite satisfaction and took the 95th percentile of the remaining times.[4] Depending on the nature of the task and the consequence of it taking too long, one could just as easily set the maximum to be the 50th percentile or 5th percentile. A specification limit set at the 5th percentile would mean the maximum acceptable time was determined by including only the fastest 5% of the most satisfied users.

In summary, to identify the maximum acceptable time (spec limit) using the bootstrap method:

1. Remove times from failed tasks.
2. Remove times where satisfaction scores are less than 4 (5 point scale).
3. Find the 95th percentile of the remaining times to arrive at the specification limit.

2.1.2 Weakness of this Method

The most obvious shortcoming to this method is that it relies on the sample of data to build a specification limit for the sample. This means the spec limit is product dependent [1 p.195]. That is, while users may be providing a sufficient satisfaction level after completing the tasks, the interface may still be forcing users to take too long to complete a task. For example, one could imagine testing the time it takes users to enter contacts into an address book application. Users may be sufficiently satisfied by the amount of time it takes to manually type in names and

[3] The same study also recommends 5.6 as the goal for 7-point scales.
[4] We used the 95th percentile instead of the 100th percentile (the slowest time for the most satisfied users) so as to offset some effects of a heavily skewed task time.

email addresses, but that same task may be able to be completed much more quickly in another application, say by the software automatically importing names and addresses from a file or from emails in an inbox. Also this method will only work if users actually successfully completed the task and rated it above a 4.

Relying on a sample of users just tested to set a specification limit for how long the task should have taken is not ideal. Task time is already highly variable [2] and it would vary, perhaps a lot, depending on the users who happened to be in the sample. However, given the alternatives (setting an arbitrary spec limit or having no spec) this method provides a reasonable starting point. A spec limit should always be evolving and take into account additional information. It is inline with recommendations for setting specification limits:

> "..any reasonable specification is better than none. Even an imperfect or incomplete specification is sure to reveal the worst of its own errors and omissions. Having done that, it serves as a reference frame within which modifications can be made [6]."

3 Examples and Evaluations

A common theme embodied in Niece, Whiteside et al and Lewis, is to take additional information and refine the specification limit. In this spirit we derived speciation limits using the steps described above and refined them as subsequent information became available. First we tested a commercially available software application with 48 users, derived specs for each task, and then tested another 49 unique users one year later using the same tasks and product. We then tested two additional samples on two competing products with 21 and 32 unique users respectively. Table 2 displays the results of using the method for all four products by task. When a faster time was encountered, the spec limit was updated (denoted by asterisks). This refined bootstrapped spec now takes into account both user based data and competitive data.

Table 2: Spec limits (in seconds) derived using the slowest 5% time from completed tasks by product and task. The N column represents the number of users that met the bootstrap requirements.

	Product 1 '03		**Product 1 '04**		**Product 2**		**Product 3**	
Task	Spec	*N*	Spec	*N*	Spec	*N*	Spec	*N*
1	214	*21*	228	*28*	135	*17*	***131**	*19*
2	95	*30*	***81**	*36*	109	*17*	96	*21*
3	223	*18*	253	*23*	***221**	*13*	340	*2*
4	202	*22*	***142**	*20*	180	*13*	161	*7*
5	296	*30*	207	*31*	242	*18*	***197**	*13*
6	454	*17*	458	*25*	**392**	*13*	522	*5*
7	415	*26*	***334**	*20*	460	*13*	361	*14*
8	263	*23*	217	*12*	***129**	*15*	206	*5*
9	224	*8*	216	*11*	235	*4*	***103**	*6*
10	154	*23*	***132**	*15*	369	*10*	262	*8*

* Denotes the fastest time by task

3.1.1 "Expert" User Time

Another alternative when no contextual-based user data is available is to use an "expert" time. The expert may be someone that works at the organization that produces the software, anyone familiar with the product and domain of the target users or even the usability engineer [4], [1 p. 197]. We were also familiar with some organizations using expert times to set time objectives (Daniel Rosenberg Personal Communication January 2004); (Christian Pantel, Personal Communication November 2004). We had an expert attempt the same tasks as the users in the four products tested above to see if any patterns emerged in helping set the spec limit. Table 3 displays the refined bootstrapped spec limit (fastest time from all four data sets) with the expert time and the ratio between the two.

Table 3: Refined Bootstrapped Spec Limit, Expert Time and Ratio between Expert and Refined Bootstrap

Task	Refined Bootstrapped Spec Limit	Expert Time	Expert to Spec Ratio
1	131	58	2.3
2	95	18	5.3
3	221	45	4.9
4	142	33	4.3
5	197	186	1.1
6	392	193	2.0
7	334	170	2.0
8	129	51	2.5
9	103	27	3.8
10	132	43	3.1

In addition to the expert times displayed above, we also examined expert times for one other product with two additional experts. We found no discernable pattern between the expert times and ratios with the bootstrapped spec limit (36 total ratios). There was a high degree of variability—a low of 1.1 and high of 7.6. The average ratio was 3.2 (SD 1.5) and the most common ratio (4.6) occurred only 3 times. More exploration is needed to understand how to more effectively use an "expert" time in building a spec limit. Expert times can still be used as a rough guide to understand minimum task times or the ratio between different tasks when planning a usability evaluation.

4 Conclusion

Setting specification limits for task times is an important step in knowing if users are taking too long to complete a task. Spec limits are also an integral part of standardizing data for use in a composite measure of usability [8]. It's best to take into account task times from a contextual inquiry, competing products or existing user data when determining spec limits. When this data isn't available, using times from the most satisfied users ensures a starting point for building a spec limit based on user data. The spec can then be refined as new data becomes available (such as completing tasks quicker on different products). No obvious patters emerged on using "expert" task times for setting the spec. Expert times can be used for helping understand relative differences between task times but more information is needed to use this data for setting spec limits.

5 References

1. Dumas, J., and Redish, J. C. (1999). A Practical Guide to Usability Testing. Portland, OR: Intellect.
2. Egan, Dennis (1988) "Individual Differences in HCI" in The Handbook of Human Computer Interaction Elsevier Science Publishers, Amsterdam, pp 541-565
3. ISO. (1998). Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability (ISO 9241-11:1998(E)). Geneva, Switzerland: Author
4. Lewis, J. R (1982) "Testing Small System Customer Setup" in Proceedings of the Human Factors Society 26th Annual Meeting p. 718-720
5. McIver, J. P., & Carmines, E. G. (1981). *Unidimensional scaling*. Thousand Oaks, CA: Sage.
6. Niece. EH Mac (1953). *Industrial Specifications* John Wiley and Sons.
7. Nielsen, J. and Levy, J. (1994) Measuring Usability: Preference vs. Performance. Communications of the ACM, 37, p. 66-76
8. Sauro, J & Kindlund E. (2005) "A Method to Standardize Usability Metrics into a Single Score." *in Proceedings of the Conference in Human Factors in Computing Systems 2005*
9. Whiteside, J., Bennett, J. and Holtzblatt, K. (1988) "Usability Engineering: Our Experience and Evolution" in The Handbook of Human Computer Interaction Elsevier Science Publishers, Amsterdam, pp 791-817