Model and Measurement for Web Application Usability from an End User Perspective[1]

Philip Lew, Li Zhang, Shouxin Wang

School of Computer Science and Engineering, Beihang University, China
philiplew@gmail.com, lily@buaa.edu.cn, shouxin_wang@126.com

Abstract. Determining quality for web applications requires an orientation beyond traditional quality models. Researchers have pointed to usability as a key component of web application quality. Usability is not a new quality characteristic. What is new is that it is a key success factor for web applications. For conventional software applications, usability may be a 'nice to have', but for web applications, it is critical due to the shifts in user expectations and business models that enable users to switch applications quickly. There has been little or no attention paid to measure software usability from an end-user viewpoint using quantitative methods. In this paper, we propose a model for web application usability from an end-user viewpoint and the basis for measurement and data collection methods to collect user activity and behavior.

1 Introduction and Background

Software usability has thus far been researched from two perspectives, usability evaluation or testing, and usability design. There are basically two methods for usability evaluation, User-based, and Expert-based. User-based methods are conducted in a laboratory in a controlled environment with participants and observers that evaluate usage based on predefined scenarios. Expert-based methods are heuristic based where experts apply a set of criteria to benchmark usability and make recommendations from a design perspective. These evaluation methods are subjective; depend on the participants, observers, expert's participation, procedures, and judgment. Both methods are time and resource intensive and hence, expensive and not practical for large scale usability evaluation and measurement.

In the area of usability design, many researchers and designers have developed usability guidelines. For example, Massey et al. [1] examined the Microsoft Usability Guidelines' design requirements and the importance of these requirements to end users. Others such as Palmer [2] and Zhang [3] have researched web site usability and design, but these are not entirely applicable because these are guidelines for design, and targeted toward web sites, rather than measuring software in use or from a users point of view. There is research specific to usability metrics as in Palmer [2], but these are for websites, rather than web applications.

[1] Thanks to the support by National Basic Research Program of China（973 project） (No. 2007CB310803).

1.1 Importance of Usability

Usability is one of the most important quality factors for web applications. Applications that have low usability are easily left for another as there are many alternatives available with another click and another short trial subscription. This is a new phenomenon resulting from the Software as a Service (SaaS) business model where trials are free so there is no cost factor involved in switching applications. Offutt [4] reinforces its connection with usability, stating that users have little loyalty and will quickly switch if an application is not easy to use.

1.2 Reasoning and Approach

Research has been done but mainly focuses on web sites and is not entirely applicable to web applications. Bruno [5] examined web applications and their many characteristics which make them different than traditional applications regarding usability. Ahmad [6] mentions that a user would not tolerate more than 3 clicks to achieve their objective.

Our reasoning is to rationally combine ideas for usability evaluation and design a quantitative evaluation measurement framework. Then, using that framework, we utilize Web 2.0 technologies and concepts to collect user behavior and activity information to determine software usability from an end user perspective. Previous techniques for websites are applicable, but their perspective is to analyze user behavior with respect to increasing sales, clicks, user return rates, time per visit. We are not directly concerned with more sales (although ultimately greater usability will lead to this), nor are we concerned with how many clicks or how long a user stays.

The remainder of the paper is structured as follows. Section 2 defines usability from an end user perspective for web applications. Section 3 describes a usability model combined with measurement factors. Section 4 discusses measurements methods and calculations while Section 5 describes our data collection methodology. Section 6 summarizes our contributions and outlines future work.

2 Defining Usability for Web Applications

In the process of developing a usability model, it is necessary to examine what usability is as defined by others in their research. Then, we add the two additional dimensions for end user viewpoint and for web applications.

2.1 What is Usability?

ISO 9241-11 defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction, in a specified context of use”. ISO 9126 standard defines usability from a software product perspective. It treats usability as one of the ‘Quality in use’ metrics that are used to evaluate the impact of the use of the software by the user. It is also addressed in the standard as “the capability of the software product to be understood, learned, and liked by the user, when used under specified conditions’.

Definitions of usability in research literature and standards include user satisfaction and quality with the three concepts usually combined together in some way. Some of the models investigated from existing research such as Abran [7] also depict this. Abran’s [7] model extends the ISO 9241 standard definition of usability to include learn-ability and security as illustrated below.

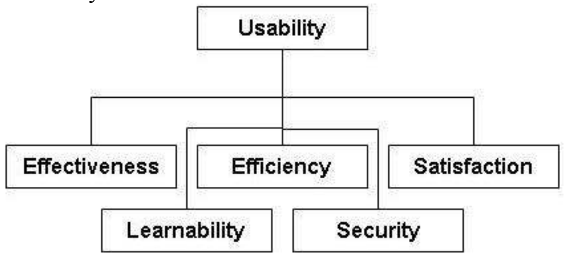


Figure 1. Consolidated usability model by Abran[7]

Learn-ability in their model is defined as the time it takes to learn while security includes access audit-ability, access controllability, data corruption prevention, and data encryption. Research by Frokjear [8] determined that effectiveness, efficiency, and satisfaction had weak correlation and should be measured separately, but did not develop any specific models.

Kappel et al. [9] examined existing usability models and correlated them to ISO 9241-11 as shown in the table below:

Table 1. Usability Model Comparisons

ISO/IEC 9241-11	Nielsen 1994	Shneiderman and Plaisant 2005
Efficiency	Efficiency	Speed of performance
	Learnability	Time to learn
	Memorability	Retention over time
Effectiveness	Errors/safety	Rate of errors by users
Satisfaction	Satisfaction	Subjective satisfaction

2.2 Usability Extensions for Web Applications

Web applications exhibit some unique characteristics that warrant additional considerations or adaptations from usability of conventional ‘shrink-wrapped’ software and websites to usability for web applications from an end user perspective.

User Diversity. Conventional software applications were targeted toward a specific user group. Web applications reach a very large and diverse user community not only common users who may want to try out the software, but also a targeted expert user group. So, as stated by Abrahao et al. [10], a large contributing factor of web application complexity comes from the need to satisfy requirements of different user profiles.

In the past, “go online” meant getting a website up and running, but today, web applications have much more functionality, leading to more user scenarios and since users are more diverse, so are the hardware and software platforms that they use.

What an application is designed for and the way a large diversity users[11] actually use the application can be very different. This leads to large quantities of unpredictable user scenarios from a user base with broader experience ranges than previous conventional software applications which targeted a small user group.

In addition, web applications differ from conventional websites in that *registered* users are not anonymous. Rather, there is a diverse range of known users with known profiles.

Self-serve and Learnability. Previously when there was a software update, new feature release, or new software replacing an old one, there was user training. There were manuals sold after-market on how to use different kinds of software such as “Excel for Dummies”. Today’s web applications must be easy to learn, yet have the complexity to satisfy expert users with little or no documentation or training other than online help. As described by Kappel et al. [9], applications must be self-explanatory. So, as pointed out by Lowe [11], increased emphasis on the user interface is one of the major differences between web applications and conventional software. SaaS has led to impatient users who do not want to invest too much time to learn an application. Thus, learn-ability is a key factor in usability for web applications.

Context Driven. Rather than simple information conveyance or e-commerce sites, today’s web applications have much more complex functionality than the websites of a few years ago. Users will have a broader experience range than previous conventional software applications leading to countless and unpredictable user scenarios. Considering the ‘specified’ part of the ISO 9241-11 definition, we must consider usage context and particular user needs. Bevan [12] indicated that usability context includes the user, tasks performed, equipment used, environment, and product being used. An application’s effectiveness cannot be measured without relativity to others in the same domain, or relative to itself over time.

3 Defining a Usability Model for Web Applications

This section defines a usability model for web applications with two main objectives. The first is measurement and the second is improving the usability of the software under measurement. As opposed to website logging which tracks where user activity as related to their purchasing behavior or interest in content, our measurement objectives are directed toward measuring user behavior with the goal of making software more usable. Our model and method focuses on the path behavior of the users, where they access these functions from, what functions they don’t use and how often among others. From these measurements, we can characterize user behavior and derive usability. Our model, adapted from Abran [7] has three components that determine usability, effectiveness, efficiency, and learn-ability. Our model differs from the others in that satisfaction is not an element that impacts usability. Rather usability impacts satisfaction and satisfaction is derived from usability and many other factors besides usability. Security, a part of Abran's usability model, has been removed and would be a factor influencing satisfaction rather than usability. Satisfaction in our model is not an element impacting usability, and as shown by Frokjaer [8], effectiveness, efficiency, and satisfaction are only weakly correlated so they should be analyzed separately.

Our usability model, shown below in Figure 2, has 3 key components, effectiveness, efficiency, and learnability. This layered model, adapted from Kappel [9] shows each of these components has a set of non-exclusive measurement elements which we denote as User Activity and Behavior factors. Each factor is influenced by a number of characteristics determined by the Application Design and environment.

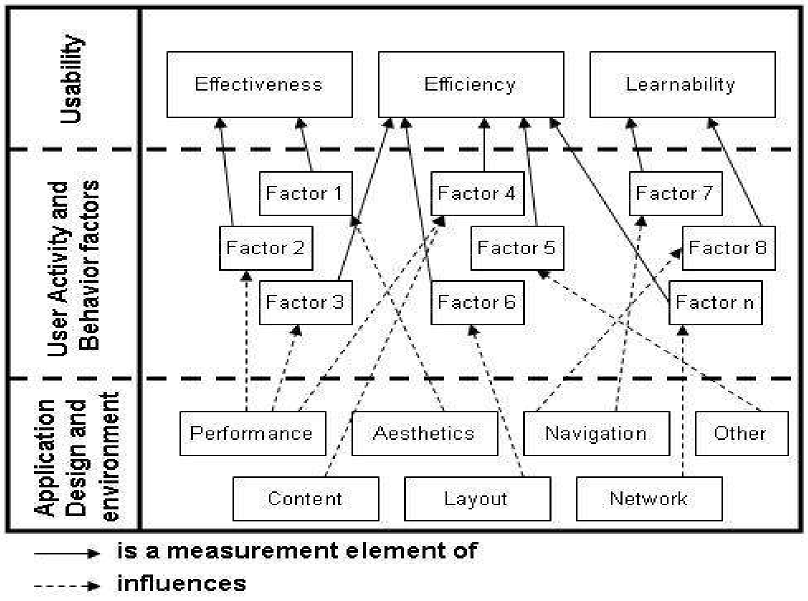


Figure 2. Web application usability model from end-user perspective

4 Measuring Usability

Unlike conventional applications, web applications are able collect data on user activity and preferences on a wide scale enabling analysis and segmentation of user activity. Existing research traces paths and other user activity and tracked data such as clicks, time, number of visits, first-time/repeat visitor, etc, but these metrics do not apply directly to web applications as they do for websites. Ahmad [6] developed methods for measuring navigational burden which included 3 factors; click distance, time taken, and number of errors. Their work cited that users should reach to their desired information within 3 clicks and that the maximum acceptable number of clicks that the user should click to complete his task is 7. For complex tasks in a web application, it may take many more clicks to complete. This, and other principles applying to websites such as stickiness do not apply to web applications because the length of time that a user spends working in an application is not necessarily directly related to usability. Martin [13] also researched user activity and usability and equated short time on a page with lack of useful information and more time spent on a page with user difficulty.

Web applications can provide good data on user interaction that can be used for usability analysis. Using the ideas from past research, our method begins with determining how to mathematically represent user activity, followed by metrics that use this representation to measure the usability factors of our model.

4.1 Measurement Concepts and Notation

This section defines three general notation methods as the basis for our measurements for each of the factors in Figure 2. These notation methods, namely User access and User path, are the basis for measuring user activity and behavior and are used for determining measurement for the usability factors.

User Access. To measure user activity, we simply measure the number of times a user accessed a function or part of the application within a time internal. This is not novel, but extended from normal website analysis into a web application. As developed by Song [14] and shown in the matrix M below, we use the same notation constructs but adapt it to apply to application functionality access rather than URL access.

$$\begin{array}{cc} & n \text{ users} \\ m \text{ function/url/menu item} & \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{1n} \\ a_{21} & & & & \\ a_{31} & & & & \\ a_{41} & & & & \\ a_{m1} & & & & a_{mn} \end{bmatrix} \end{array}$$

Figure 3. User access matrix

In the User access matrix M above:

- a_{ij} is number of times user j within a time period visits a function/url or menu item i.
- There are j users during the time period that access the various functions of the system and i functions/url/menu items in the system
- Each row vector M[m,j] represents all the users that use that part of the system. In this instance below, function or part 1 of the application for all users j=1 to m.

$$\sum_{j=1}^{m} a_{1j} \cdot \tag{1}$$

- Each column vector M[i,n] represents the total usage of each part of the system for that user. In this instance below for all the functions of the application from i=1 to n, for one user, user 1.

$$\sum_{i=1}^{n} a_{i1} \tag{2}$$

Depending on how granular the application is broken up and then organized and grouped in the matrix, we can determine user access patterns. Column vectors reflect the activity of an individual user or could be groups or segmented by user group. So instead of j=1 representing user 1, it would represent the activity of user group 1, i.e. administrative users. If we examine the row vector similarity, we can equate this to the access similarity across the application by all users depicting usage access characteristics. Likewise, we can examine the column vector similarity and determine user diversity and usage across groups, and customer access patterns. As described by [15], Hamming distance measures the minimum number of substitutions required to change one into the other and as adapted by Song [14] for computing vector similarity based on the Hamming distance whereby the smaller the value for Hamming distance, the higher the degree of similarity.
Definition. (Hamming distance): For $\forall \mathrm{M}[i,j] > 0$, M [i, j]=1, Hamming distance between vectors $X, Y \in \{0,1\}^n$, n>=1,

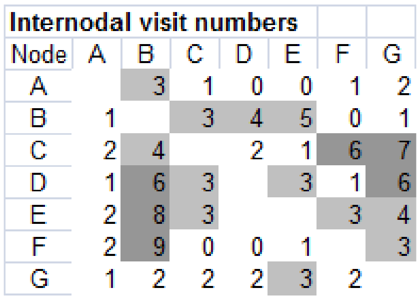
$$\mathrm{H}_d(X,Y) = \sum X[i] \oplus Y[i] \,. \tag{3}$$

As an example, the matrix shown in Figure 4 depicts the Hamming distance between the last two vertical vectors is 1. This shows node or function access density across users and user clustering behavior on a function by function basis.

$$\begin{array}{cc} & n \text{ users} \\ m \text{ function/url/menu item} & \left[\begin{array}{c:c:c:c:c} 1 & 1 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 3 & 1 & 1 \\ \hdashline 1 & 2 & 3 & 1 & 2 \end{array}\right] \end{array}$$

Figure 4. Hamming distance example matrix

User Path. To depict the path of a user from node to node, or function to function, we denote their path with a simple two dimensional matrix as shown below.



Internodal visit numbers							
Node	A	B	C	D	E	F	G
A		3	1	0	0	1	2
B	1		3	4	5	0	1
C	2	4		2	1	6	7
D	1	6	3		3	1	6
E	2	8	3			3	4
F	2	9	0	0	1		3
G	1	2	2	2	3	2	

Figure 5. User path matrix - 2 dimensions

For visualization purposes, darker shades represent those paths visited more frequently. To extend this measurement method to more than 2 nodes/2 functions, dimensions of the matrix are extended as shown in Figure 6.

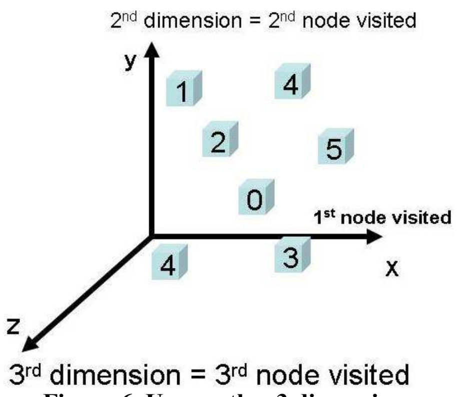

2nd dimension = 2nd node visited
y
1
4
2
5
0
1st node visited
4
3
x
z
3rd dimension = 3rd node visited

Figure 6. User path – 3 dimensions

As shown in Figure 6, a task is equated to a path and the paths that a user can take within a software application are represented by an n-dimensional matrix with n being the number of nodes in the path. For example, 5 functions or menus accessed would be represented by a 5 dimensional matrix. The sequential path of a user is represented by a vector within the matrix.

Using this path concept, we extend the work of Song [14] to represent not just visitation of individual URL's or functions by users, but the path of users as shown in Figure 7 below.

$$\begin{array}{cc} & n \text{ users} \\ m \text{ path} & \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{1n} \\ a_{21} & & & & \\ a_{31} & & & & \\ a_{41} & & & & \\ a_{m1} & & & & a_{mn} \end{bmatrix} \end{array}$$

Figure 7. User path matrix

Using this matrix representation:

- Each row vector represents a path and it's corresponding usage across all users.
- Each column vector represents a user's (or group of users) path usage.
- Paths can be of different lengths and are ranked with most prevalent paths first so the in terms of path frequency:

$$\sum_{i=1}^{n} a_{m,i} > \sum_{i=1}^{n} a_{m,i+1} \qquad (4)$$

By extending our logic from analyzing access to parts of an application, the Hamming distance between the horizontal vectors (paths) indicates path similarity in usage among the users while the difference between the vertical vectors tells us usage similarity. We extend this notation further into the functionality of the software as granular as necessary depending on our analysis. For instance, each node can represent a function, and functions can lead to other functions, similar to a use case.

To analyze the similarity of different length paths, we use the Levenshtein distance which is more appropriate than the Hamming distance. **Levenshtein distance** is a metric for measuring the amount of difference between two sequences [15] and includes additions and deletions in addition to changes to account for shorter and longer paths.

4.2 Model Factors and Measurement Method

Each of the notation methods in the previous section serve as a basis for measuring the usability factors below. Note that when measuring the time dimension, this means that the factor is measured over calendar time at the desired level of granularity (day, week, month, year, etc…) to measure trends. Measuring a usability factor in the user dimension indicates measuring users at different levels of granularity as well (user, user group, or all users). In the section below, we list and describe usability factors. Each of these factors is input to the three core elements of our model: effectiveness, efficiency, learnability.

This section describes each usability factor in our model. Each factor is described using measurement notations in the previous section and where possible calculations are proposed. Note that we meaning of node is interchangeable with function or part of an application. In general, the frequency of that path indicates the popularity of that task or the usefulness of that part of the application to users. For web applications, we can get more granular down to the function and part of a page that users are viewing. Examining where users are on each page/part/function of the application as indicated by Martin [13] can also tell us about user interest. In a website, if the user never scrolls down, this may indicate useless information as perceived by the user. For an application, we can note which functions are called more or less frequently. For instance, every user will do the login task. For an accounting application, 50 percent of users may do the account reconciliation task and follow that path through the application.

Effectiveness. In simple terms, effectiveness means getting the job done accurately and completely while achieving the stated goals as measured by the quality of the solution and number of errors.
End node preference: Multiple paths with the same ending node indicate either more than one way to accomplish the same task, different nuances or parameter inputs (along the path) to that node, or an individual node's importance or usage. Different frequencies for different paths with the same end node may also indicate preferential path/usage to complete a similar function or task. Using our User path notation:
1. Identify all vectors that end with the same node
2. Calculate the Levenshtein distance between these vectors for that end node.
3. Calculate the mean and standard deviation of the Levenshtein distances
Conceptually, this can be visualized in Figure 8. The same end node is represented by the vertex, with the differing paths getting to that end node and the spaces between the vectors depicting the Levenshtein distance or similarity/dis-similarity. There are different ways to interpret the meaning of the number of vectors with the same end node and the Levenshtein distances between these vectors. More than one way to accomplish the same task increases the effectiveness of the application by giving users the choice of completing a function or task. A close similarity or short Levenshtein distance between different paths could also confuse a user depending on the task they are trying to complete.

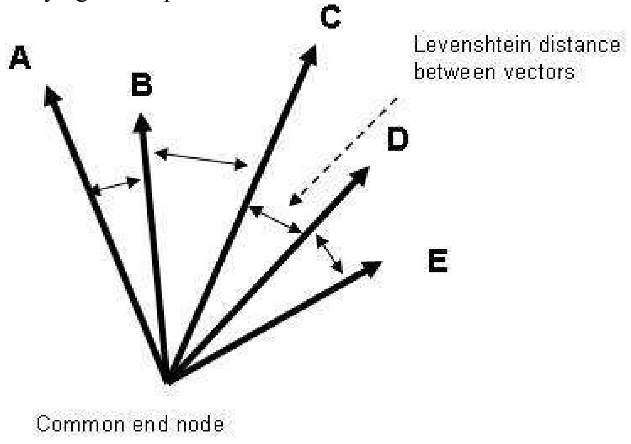


Figure 8. Levenshtein vector similarity

Error rate: This counts the errors that a user makes which can indicate either the application is complex (or possibly has defects), or the user is unskilled. If the user is unskilled, then this rate should decrease over time for that user. If not, then the application can be considered complex. This is calculated as errors/session, or errors/task completed where n is the number of sessions or task, and Sum(e) represents the total number of errors made by a user/user group in the i^{th} session (S) or task (T):

$$ER_s = \frac{\sum_{i=1}^{n} Sum(e)_{Si}}{n} \quad \textbf{(5)}$$

$$ER_t = \frac{\sum_{i=1}^{n} Sum(e)_{Ti}}{n} \quad \textbf{(6)}$$

Backtracking: A user that goes to a function or part of the application and then rapidly returns to his original place, or rapidly leaves without accomplishing any task negatively impacts efficiency. This can be measured for a given time threshold, for a particular user group or for all users. A user currently at node n, or function n, and moves to node n+1, and does not surpass the time threshold, and the goes back to node n, then increase count by 1 for that node n, n+1 pair. This is calculated as backtracks/session, or backtracks/task completed where n is the number of sessions or task, and Sum(b) represents the total number of backtracks by a user/user group in the i^{th} session (S, shown below) or task (T):

$$BR_s = \frac{\sum_{i=1}^{n} Sum(b)_{Si}}{n} \quad \textbf{(7)}$$

Help usage: Using online help can be positive or negative for usability. If used frequently, this means that the application is not intuitive or not easily understood. Yet, accessing online help and finding what is needed can also be considered positive. This can be measured as number of accesses, time per access, per session per user or user group. Using our User access notation, and then all accesses to help from node n are counted for node n. This is calculated as help/session, or help/task completed where n is the number of sessions or task, and Sum(h) represents the total number of help accesses made by a user/user group in the i^{th} session (S, shown below) or task (T):

$$HR_s = \frac{\sum_{i=1}^{n} Sum(h)_{Si}}{n} \quad \textbf{(8)}$$

Unused: Using the user access notation above, this tracks the nodes or functions of the application that are never or seldom used. This indicates that this part of the application is either difficult to use or has limited value, or has limited users who want or need this function. This is measured by taking nodes least visited (using a less than threshold per time period) for a user or user group. This is calculated as unused/session, or unused/task completed where n is the number of sessions or task, and Sum(h) represents the total number of unused nodes by a user/user group in the i th session (S, shown below) or task (T):

$$UR_s = \frac{\sum_{i=1}^{n} Sum(u)_{Si}}{n} \tag{9}$$

Efficiency. Efficiency means how fast the task can get done taking into account the resources expended. Efficiency factors are tracked over the time dimension and user dimension.

Task length: Longer paths to complete a given task negatively affect efficiency in the same context. Using our User path notation method above, we take the average number of dimensions for the paths in the application in the time and user dimensions. Although we have not completed our experiments, we expect the task length follow a normal distribution. Using this assumption, if a given task can be completed on average of 5 nodes and less than one standard deviation below the average, for example, less than or equal to 3 nodes (if std. dev = 2) ,we can assign a good rating or 100% . On the other hand, one standard deviation higher than the average, greater than 7 nodes, is unacceptable and will get 0% score. To find the score value for in between values of 3 to 7, we use interpolation as shown in Table 2 for a user completing a task in 6 nodes:

Table 2. Nodes to Complete task & Score Relationship

	Nodes	Score	
X1	$\rightarrow$ 3	100%	$\leftarrow$ Y1
X2	$\rightarrow$ X2	Y2	$\leftarrow$ Y2
X3	$\rightarrow$ 7	0%	$\leftarrow$ Y3

In the table above we can see that for 3 nodes the score is 100% and for 7 nodes the score is 0%. To find the score (Y2) for 6 clicks (X2) we use the following expression which in our example yields 0.25 (25%) for 6 clicks:

$$\frac{y3 - y1}{y2 - y1} = \frac{x3 - x1}{x2 - x1} \tag{10}$$

$$y2 = \frac{7 - x2}{4}$$

Function time: The amount of time spent on any particular node or function could indicate the complexity of that part of the application, or value provided to users.

Frequent paths indicate often executed tasks which should be analyzed for task speed. By measuring the task speed or time spent during a function, the standard deviation across users indicates the diversity of the user group. We utilize our User path notation to sum the time spent for a path in both user and time dimensions or the User access notation to sum the time access per node/session across user and time dimensions and compare this to the time spent in the entire application. Let $t_{m,n}$ = time for function m or path m (to complete a task), for user n, then:

$$FT_m = \sum_{i=1}^{n} t_{m,i}\, a_{m,i} \,. \tag{11}$$

This is the total time for all users for each function or path m. Then we can get the average by:

$$FT_{avg} = FT_m / n \,. \tag{12}$$

Learn-ability. According to ISO 9126, learnability is the capability of a software product to enable the user to learn how to use it. Over time, if the application is highly learn-able, this time will decrease for returning users. The rate of decrease over a time period tells us the user's speed of learning. For instance, decreasing task speed by 20 percent in 10 days is much better than improvement by the same amount over 2 months. A user can not keep getting faster and faster, but if they initially have great speed increases, this indicates the user learned that task quickly and can find the appropriate functions with decreasing effort. Therefore, measuring rate of change over time rather is more important than pure speed in itself.
Learning rate: As users become more familiar with an application, they can get the same task done faster. This is measured by the rate at which their speed increases over identical paths. By first taking the paths with the highest frequency usage from our User path analysis, then we measure the time for each these paths, and after a fixed time period, we measure the speed again to get the difference/time period. This can be measured in the user and time dimension. Using the same notation from function time, we use the difference over a time period T for a path m for measurements taken at time t1 and t2.

$$LR = (FT_{avg}(t1) - FT_{avg}(t2))/T. \tag{13}$$

Task time deviation: Lower standard deviation across user groups for a given task indicates inexperienced or novice users can catch up and learn quickly and therefore be able to complete a task the same as an expert user. As time progresses, a new novice user should catch up to an expert user quickly if the application is easily learned. In contrast, measurements of high deviation indicate poorer learn-ability. This is measured using our User path method and over the time and user dimensions. Using our previous definitions with n = number of measurements, and X representing individual measurements of FT_m.

$$\text{FTstd}=\sqrt{\frac{\sum(X-FT_{avg})^2}{n-1}} \tag{14}$$

Short path: Similar to backtracking, a node where a user often stops and exits the application may indicate a mistake, or confusion. Using our User path method, we fix a short path threshold according to the number of dimensions and then a time threshold. For instance, by setting dimension threshold to 2 and time threshold to 2 seconds, we take all paths that have only 2 nodes, with less than 2 seconds spent at node 2. This is calculated as short paths/session, or short paths /task completed where n is the number of sessions or task, and Sum(sp) represents the total number of help accesses made by a user/user group in the ith session (S, shown below) or task (T):

$$SP_s = \frac{\sum_{i=1}^{n} Sum(sp)_{Si}}{n} \tag{15}$$

Note that our layered usability model as shown in Figure 2 has dotted lines from the Application Design and Environment depicting that these elements influence the factors above and some factors may be measurement elements of more than one usability component. This is a work in progress as there are several steps left to finishing the model including:

- Normalizing each factor input
- Determining an overall calculation method for usability
- Determining which measurement elements impact more than one usability component as shown by the solid arrows in our model
- Verifying each factor's logic through survey or other means

5 Measurement and Data Collection Methodology

As researched by Martin [13], we identify users by using a session ID which enables us to identify a unique user. Then, implementing our methodology is done iteratively where start with inserting code into the application under study. We begin collecting data into our User path and User access matrices which tell us which paths and nodes need to be analyzed the most. Then we analyze the data using our calculations for each factor. After doing our calculations, we can determine the need for further calculations and what paths to analyze further and then go back and revise the data we are collecting or insert more code into the different parts of that path, including the separate functions inside the application. Data can then be analyzed in many dimensions and aggregations depending on our data schema design. In the process of analysis, we may find other factors that should be calculated or others currently in the model that should be eliminated.

6 Summary and Future Work

The main contributions of the paper are a model for determining usability for web applications from an end user perspective combined with measurement methods to determine benchmark measurements of usability for web applications. We use the term benchmarks because any measurements must be done within domain context of different web applications or for different versions of the same application.

In the future, we need to verify and normalize calculations, and develop an overall calculation for usability based on each factor calculated. Each measured factor may also impact more than one component of usability. For instance, navigational complexity may impact both effectiveness and efficiency. This will require several experiments with different applications in the same domain in order to validate and perfect our model and measurement methods.

7 References

1. Massey, Khatri, and Montoya-Weiss, Online Services, Customer Characteristics and Usability Requirements, Proceedings of the 41st Hawaii International Conference on System Sciences 2008.
2. Palmer, J. (2002). Web Site Usability, Design and Performance Metrics, Information Systems Research 13(2): 151-167.
3. Zhang, P., Von Dran, G.M. (2000). Satisfiers and dissatisfiers: A two-factor model for website design and evaluation. Journal of the American Society for Information Science 51(14): 1253-1268.
4. Wu, Offutt, Modeling and Testing Web Applications, 2002.
5. Bruno, V., Tam, A., Thom. J. (2005). Characteristics of web applications that affect usability: a review. In ACM International Conference Proceeding of the 17th Australia conference on Computer-Human Interaction (HCI'05) (pp. 1-4).
6. Ahmad, Zhang, and Azam, Measuring 'Navigational Burden', Proceedings of the Fourth Int'l, Conf. on Software Engr. Research, Management and Applications, 2006.
7. Abran, Khelifi, and Suryn, Usability Meanings and Interpretations in ISO Standards, Software Quality Journal, 2003-11, 325-338.
8. Frokjaer, Hertzum, Hornbaek, Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?, SIGCHI conference on Human factors in computing systems 2000, 345-352.
9. Kappel et al., The Discipline of Systematic Development of Web Applications, 2003, John Wiley and Sons.
10. Abrahao, Pastor, and Olsina, A Quality Model for Early Usability Evaluation, Int'l COST294 WS UIQM, 2005.
11. Lowe, Web system requirements: an overview, Requirements Eng (2003) 8: 102–113.
12. Bevan, Macleod, Usability Measurement in Context, Behavior and Information Technology, 1994 Vol. 13, Pages 132-145.
13. Martin, Usability Analysis and Visualization of Web 2.0 Applications, 10th IEEE International Symposium on Web Site Evolution, 2008, 121-124.
14. Song et al., An Efficient and Multi-purpose Algorithm Form in Weblogs, International Conference on Computer Networks and Mobile Computing, 2001.
15. Wikipedia, Hamming Distance, 31, May 2009, http://en.wikipedia.org/wiki/Hamming_distance.