VOL. 3, NO. 4, June 2013 ISSN 2222-9833
ARPN Journal of Systems and Software

http://www.scientific-journals.org

An Effort-Based Framework for Evaluating Software Usability Design

[1]Dan Tamir, [2]Carl J. Mueller, [3]Oleg V. Komogortsev
[1,3]Assoc. Prof., Department of Computer Science, Texas State University
[2]Assist. Prof., Department of Computer Information Systems, Texas A&M University Central Texas,

[1]dt19@txstate.edu,[2]muellercj@ct.tamus.edu, [3]ok11@txstate.edu

ABSTRACT

One of the major stakeholder complaints is the usability of software applications. Although there is a rich amount of material on good usability design and evaluation practice, software engineers may need an integrated framework facilitating effective quality assessment. A novel element of the framework, presented in this paper, is its effort-based measure of usability providing developers with an informative model to evaluate software quality, validate usability requirements, and identify missing functionality. Another innovative aspect of this framework is its focus on learning in the process of assessing usability measurements and building the evaluation process around Unified Modeling Languages Use Cases. The framework also provides for additional developer feedback through the notion of designer's and expert's effort representing effort necessary to complete a task. In this paper, we present an effort-based usability model in conjunction with a framework for designing and conducting the evaluation. Experimental results provide evidence of the frameworks utility.

Keywords:*Software Development, Software Usability, Eye Tracking, Graphical User Interface*

1. INTRODUCTION

The decades that followed the introduction of the concept of interactive user interface along with important computer graphics concepts by Sutherland have produced a proliferation of user interface devices and procedures, as well as an abundance of approaches and methods for evaluating the effectiveness of the interface or its usability [1-3]. There are several definitions for software usability. The IEEE standard glossary defines software usability as "the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component" [4]. As the use of advanced interface techniques became a prominent trend and almost a requirement, the importance of the concept of software usability became the cornerstone of user interface design and evaluation. Many hardware and software companies and vendors found themselves either riding the wave of good usability to enormous success or facing a painful failure due to lack of usability. Software usability is one of the most important factors in a user's decision on whether to acquire and use a system as well as a major factor affecting software failures [5, 6].

The research reported in this paper presents a novel framework for enhancing the capability of the HCI community to advance the state of the art of usability research and practice. Under this framework, a set of users executes a set of identical independent tasks, which emerge from a single scenario. While the tasks share a scenario, they differ in important parameters so that the user cannot "just" memorize a sequence of interaction activities. Throughout the interaction process, certain user activities such as eye movement, time on task, keyboard, and mouse activities are logged. The accumulated data is reduced and several metrics that relate to user effort as well as time on task are extracted. Next, the averages of the per task measurement are compared to an expected learning curve which represents the user's mastery as progress through the set of tasks. We show that the average learning curve can improve a multitude of current usability testing and evaluation techniques. First, we show that the **learnability** of software systems can be accurately assessed and the point of the user's mastery can be identified. Furthermore, the learning curve can be used to compare the **operability** and **understandability** of different systems or different groups of users using the same system. We explain how software designers and developers can employ the novel concepts of **designer's** and **expert'seffort** in the process of user interface design and evaluation, and demonstrate the use of the effort based metrics to provide interface designers and developers with a methodology to evaluate their designs as they are completed. In summary this paper, presents a framework permitting software engineers to utilize time and effort based measures to validate software usability and compare the usability of similar systems.

The remainder of the paper is organized in the following way. Section 2 provides a review of related work. Section 3 presents a time and effort based usability model. Section 4 presents a usability testing framework and demonstrates the use of the metrics presented in Section 3. Section 5 presents experimental results demonstrating the validly of the usability model and testing framework. Section 6 presents conclusions and future research.

2. RELATED WORK

The Human Computer Interface (HCI) community is involved with all the aspects of usability including definition, standardization, assessment, and measurement along with research and development of new concepts [3, 7-21]. Research shows that despite the fact that software developers are fully aware of the importance of usability they tend to neglect this aspect during the software development lifecycle [22]. This may be due to confusion about the way to design a usable interface and perform usability tests or lack of precise usability "tools" available to software developers.

Two international standards are addressing the issue of usability, the ISO 9241-11 and the ISO/IEC 9126-1:2001 [23, 24]. The ISO 9241-11 standard views software usability from the prospective of the user and provides three main metrics: **satisfaction, efficiency, and effectiveness**. These characteristics are important because users can relate to them, and assign measurable values to their experience with the software in terms of these attributes. In addition to these three attributes, the ISO 9241-11 standard provides the evaluator with a set of optional usability characteristics. The ISO/IEC 9126-11 standard views software quality from three perspectives; the perspective of the end user, a test engineer, and a developer [24]. An end user's perspective of software quality includes **effectiveness, productivity, satisfaction, and safety.** For a test engineer, software quality is composed of externally visible quality characteristics describing **functionality, reliability, efficiency, and usability**. Internal quality or the developer's perspective of quality in the ISO/IEC 9126 includes **maintainability and portability**. Evaluating software usability using the ISO/IEC 9126 standard requires evaluating both the quality in use and the external quality. The ISO/IEC 9126 defines the following sub-characteristics of usability: **Operability, learnability, understandability, attractiveness, and compliance**. Operability is the capability of a user to use the software to accomplish a specific goal. Learnability is the ease with which a user learns to use the software. Understandability is the ability of a user to understand the capabilities of the software. Attractiveness relates to the appeal of the software to a user. Finally, Compliance measures how well the software adheres to standards and regulations relating to usability. Recommended approaches for measuring each of the sub-characteristics are provided [24, 25]. These recommended measurements rely heavily on psychometric and cognitive evaluation techniques.

Many experts view ease of learning or time to learn as an important characteristic of software usability, and it is also essential to the IEEE definition of usability. [1, 3, 26-28]. Nielsen's adds memorability as a major characteristic of software usability. Shneiderman and Nielsen classify computer users as novice and experts. Shneiderman extends this definition to include an occasional user. These categorizations emphasize learning as a primary characteristic of usability. A novice is in the process of learning to use the software. An occasional user is continually relearning to use the software. An expert has already mastered the software and learns very little each time they use the software. This leads to the conclusion that it is desirable to design the evaluation of usability in a way that enables establishing the learnability of the software.

Software engineers generally employ a number of different methods to evaluate software quality characteristics, such as correctness proofs, inspections, and testing [29-34]. Usability evaluation has a much richer assortment of methodologies available to the evaluator [3, 8, 9, 11, 14, 16, 17, 20, 35-38]. The paper "Strategies for Evaluating Software Usability" illustrates the large assortment of evaluation methods [37]. The list of generic methods cited by Fitzpatrick, includes: Observation, Questionnaire, Interview, Empirical Methods, User Groups, Cognitive Walkthroughs, Heuristic Methods, Review Methods, and Modeling Methods.

Each of these evaluation methods has strengths and weaknesses. Reviews, Cognitive Walkthroughs, and Heuristic Methods are good approaches for evaluating user interface and providing quality feedback to designers early in the development process. Inspections, however, are limited to investigating interface issues and do not evaluate the way that the hardware and software interact. Modeling methodologies, such as Goals, Operators, Methods, and Selection rules (GOMS), provide excellent techniques for evaluating user interface designs before they are implemented [18]. Observation, Questionnaire, Interviews, and User Groups offer excellent feedback about the user perception of the usability of the software. The Empirical approach provides valuable data about the components' interaction with users and employs the same concepts as those used in the validation stage of development to evaluate other quality characteristics exhibited by the software.

Observation, Questionnaires, Interviews, User Groups and Empirical or Execution based testing methodologies employ users' experience with the software as the basis of their evaluation. From the prospective of a software developer, all of these methodologies are problematic. Acquiring accurate data from the user-based methods requires having all of the hardware and software for the facility under evaluation. This type of summative evaluation in a traditional or a waterfall development process is limited to the validation or final phase of a development cycle [4, 33, 39, 40].

Most software development processes divide quality evaluation methods into Verification and Validation (V&V) activities. Classifying an evaluation methodology as a Verification or Validation task depends on the type of information produced by the activity. If an evaluation methodology produces data about the product under development, it is a verification activity [4]. If an evaluation methodology produces data about the way a product addresses the users' needs, the methodology is classified as a validation activity and serves as a formative evaluation [4, 40]. A rigorous V&V plan employs multiple evaluation methods in order to provide the evaluator with the maximum amount of data for their evaluation, and serves as a normative evaluation of the product [33, 41, 42].

Most of the literature on empirical usability evaluation or usability testing comes from the Human Computer Interaction (HCI) community [3, 8, 14, 20]. One of the most comprehensive discussions on usability evaluation occurs in Handbook of Usability Testing[20]. The approach described in this text provides a good combination of inspections and empirical evaluation or testing activities. Even though the text provides no differentiation between inspections and tests, the authors provide a robust definition permitting most software engineers to identify the activity. For example, the authors propose an exploratory test conducted early in the cycle to provide feedback between the requirements and design phase. Software testing terminology

defines a test in terms of the execution of software [4, 34]. Using a classical development methodology, such as the waterfall model, there is nothing to execute during this phase of the development. Even when prototyping concepts are employed, evaluations at this point in time are characterized as an inspection [31].

Most cognitive evaluation methods however, require expertise in the cognitive science and are not readily available to the developers. Moreover, currently cognitive evaluation does not exploit contemporary and advanced technology to its fullest. For example, time on task, which is an important factor of usability, is often measured using a manual stop-watch operated by an HCI expert during an observation session. Current technology can provide the HCI experts with a "perfect stop-watch" that measures to a sub second accuracy the timing events related to the interface and correlate them with spatial and temporal user and system activities. Moreover, it can supply the evaluator with correlation of time on task and interface events in different areas of the screen, correlation between timing and eye movements of the user which can give indication on widget placement, as well as an indication that at a specific time the user rapidly moved the mouse back and forth from one widget to another. In addition, the concepts introduced in the following section can assist in the development and validation of software usability, and provide an enhanced issue and defect resolution techniques.

3. EFFORT BASED USABILITY METRICS

As stated in the previous section, the ISO/IEC 9126-1 lists 5 main sub characteristics of usability: operability, learnability, understandability, attractiveness, and compliance. The model described in this paper concentrates on learnability, operability, and understandability, where we list learnability first since it is a cornerstone of the new approach and is one of the most frequently cited software usability characteristics.

It is possible to construct a model of usability using the sub-characteristics of operability, learnability, and understandability. With the elimination of the subjective characteristics such as attractiveness, compliance, and satisfaction, the model may lack a level of precision but should provide a basis for both summative and formative evaluation. In this section, we present the theoretical model of effort based usability and elaborate on the learnability, operability, and understandability aspects.

3.1 Effort-Based Usability Model

One of the hypotheses that govern this research is that effort, which can be measured by several objective measures, closely (and inversely) correlates with usability. For this model, E denotes all the effort required to complete a task with computer software, as defined by the following vectors:

$$E = \begin{pmatrix} E_{mental} \\ E_{physical} \end{pmatrix}$$

$$E_{mental} = \begin{pmatrix} E_{eye\_mental} \\ E_{other\_mental} \end{pmatrix}$$

$$E_{physical} = \begin{pmatrix} E_{manual\_physical} \\ E_{eye\_physical} \\ E_{other\_physical} \end{pmatrix}$$

Where:

E_{eye_mental}	the amount of mental effort to complete the task measured by eye related metrics.
E_{other_mental}	the amount of mental effort measured by other metrics.
$E_{physical}$	the amount of physical effort to complete the task.
$E_{manual\_physical}$	the amount of manual effort to complete the task. Manual effort includes, but is not limited to, the movement of fingers, hands, arms, etc.
$E_{eye\_physical}$	the amount of physical effort measured by eye movement related metrics.
$E_{other\_physical}$	the amount of physical effort measured by other metrics.

The accurate estimation of the total effort (E) requires a complete knowledge of the mental and physical state of a human being. This is not possible with the current technology; therefore, approximations techniques are used to estimate total effort (E). Logging keystroke and mouse activity approximates the manual effort ($E_{manual\_physical}$) expended by a subject. An eye-tracking device allows logging eye position data to estimate the amount of mental effort (E_{eye_mental}) and physical effort ($E_{eye\_physical}$) in terms of eye movement metrics. Terms such as E_{other_mental} and $E_{other\_physical}$ are presented as estimation factors that might be contributing to the effort required for task completion that cannot be measured accurately by current technology.

The accurate estimation of the total effort (E) requires a complete knowledge of the mental and physical state of a human being. This is not possible with the current technology; therefore, approximations techniques are used to estimate total effort (E). Logging keystroke and mouse activity approximates the manual effort ($E_{manual\_physical}$) expended by a subject. An eye-tracking device allows logging eye position data to estimate the amount of mental effort (E_{eye_mental}) and physical effort ($E_{eye\_physical}$) in terms of eye movement metrics. Terms such as E_{other_mental} and $E_{other\_physical}$ are presented as estimation factors that might be contributing to the effort required for task completion that cannot be measured accurately by current technology.

3.1.1 Mental Effort

Salomon defines the notion of Mental effort as the "number of non-automatic elaborations applied to a unit of material"[43]. He employs the concept in motivational and cognitive aspects of information processing. A related term is cognitive load, which refers to the ability to process new information under time constraints. The accurate assessment of information processing and cognitive load is hard due to the fact that it involves attention, perception, recognition, memory, learning, etc. Nevertheless, it can be partially estimated by eye movement metrics measured by an eye tracking device[44, 45]. Modern eye tracking devices are similar to web cameras, without any parts affixed to the subject's body [46]. Eye trackers provide useful data even in the absence of overt behavior. With this device, it is possible to record eye position and identify several eye movement types. The main types of the eye movements are [46]:

Fixation – eye movement that keeps an eye gaze stable with respect to a stationary target providing visual pictures with high acuity,

Saccade – very rapid eye movement from one fixation point to another,

Pursuit – stabilizes the retina with respect to a moving object of interest.

In the absence of dynamically moving targets, the Human Visual System usually does not exhibit pursuits. Therefore, parameters related to smooth pursuit are not discussed in this paper. In addition to basic eye movement types, eye tracking systems can provide biometric data such as pupil diameter. Since pupil diameter might be sensitive to light conditions and changes of brightness level of the screen, it is not included as a mental effort metric in this research.

Many researchers consider the following metrics as a measure of the cognitive load and mental effort:

Average fixation duration: measured in milliseconds [47, 48].

Average pupil diameter: Eye tracking systems enable measuring biometric data such as pupil diameter. [49], [50, 51].

Number of fixations: Due to non-optimal representation, overall fixations relate to less efficient searching [52]. Increased effort is associated with high amounts of fixations.

Average saccade amplitude: Large saccade amplitude, measured in degrees, can be indicative of meaningful cues [52, 53]. To a certain extent, large average saccade amplitude represents low mental effort due to the notion that saccades of large amplitudes indicate easier instruction of meaningful cues [54].

Number of saccades: High number of saccades indicates extensive searching, therefore less efficient time allocation to task completion[52]. Increased effort is associated with high saccade levels.

Generally, the user expands some type of effort at any given time. This effort might be related to eye muscle movements, brain activity, and manual activity. Hence, $E(t)$, the effort expanded by the user at time t, is a continuous time function (i.e., analog function). For the mental effort, it is assumed that:

$$E_{eye\_mental}(t) = \int_{t_0}^{t} \big(f_1(t) + f_2(t) + f_3(t)\big)dt$$

Where: $f_1(t)$ is a function of the fixation duration and number of fixations; $f_2(t)$ is a function of the pupil diameter, and $f_3(t)$ is a function of the saccade-amplitude and number of saccades. Note that $E(t)$is a monotonically increasing function. The definition of effort uses continuous time functions. In practice, given the discrete nature of computer interaction, these measures are quantized by converting integrals to sums. Occasionally, eye-tracking devices produce data that is below a reliability threshold. Periods where the data is not reliable are excluded from integration.

Direct evaluation of the actual functions $E(t), f_1(t), f_2(t)$,and $f_3(t)$ is complicated and is beyond the scope of this research. Nevertheless, a research to estimate the expanded effort through a model of the eye muscles is currently ongoing. In the current research, we represent $E_{eye\_mental}(t)$ as a vector consisting of two elements:

$$E_{eye\_mental}(t) = \begin{Bmatrix} \text{Average fixation duration} \\ \text{Number of fixations} \end{Bmatrix}$$

Where the average fixation duration and the number of fixations are calculated over a period $[t_0, t]$. Since saccades are highly correlated to physical effort, they are listed under $E_{eye\_physical}$

3.1.2 Physical Effort

3.1.2.1 Manual Physical

The main components of the manual physical effort expanded by the user relate to mouse and keyboard activities. Hence, in the case of interactive computer tasks, it may be possible to calculate effort by counting mouse clicks, keyboard clicks, Mickeys, etc. The term Mickey denotes the number of pixels (at the mouse resolution) traversed by the user while moving the mouse from a point(x_0, y_0) to a point(x_1, y_1).

In similarity to the definition of$E_{eye_mental}(t)$, $E_{manual\_physical}(t)$ is defined to be:

$$E_{manual\_physical}(t) = \int_{t_0}^{t} \big(f_4(t) + f_5(t) + f_6(t) + f_7(t)\big)dt$$

Where: $f_4(t), f_5(t)$and$f_6(t)$ are (respectively) functions of the number of mickeysF, mouse clicks, and keystrokes by a subject during the time interval$[t_0, t]$; and $f_7(t)$ is a function that serves as a penalty factor that measures the number of times the user switched from mouse to keyboard or vice versa during the interval.

As in the case of mental effort, direct evolution of the functions$f_4(t), f_5(t), f_6(t)$, and $f_7(t)$ is beyond the scope of

this research. In the current research we represent $E_{manual\ \_physical}\ (t)$ as a vector consisting of the elements:

$$E_{manual\ \_physical}\ (t) = \begin{Bmatrix} \text{mickeys count} \\ \text{mouse clicks count} \\ \text{kestrokes count} \\ \text{transitions count} \end{Bmatrix}$$

Where the counts are calculated over a period $[t_0, t]$.

3.1.2.2 Eye Physical

Ideally, the effort expended by the Human Visual System (HVS) to complete a task is represented by the amount of energy spent by the HVS during the task. The energy expended is dependent on the amount of eye movements exhibited by the HVS, the total eye path traversed and the amount of force exerted by each individual extraocular muscle force during each eye rotation. In similarity to the discussion on $E_{eye\ \_mental}$, the following parameters are considered.

Average saccade amplitude (see definition above).

Number of saccades(see definition above).

Total eye path traversed: This metric, measured in degrees, presents the total distance traversed by the eyes between consecutive fixation points during a task.This metric takes into account the number of fixations, number of saccades and exhibited saccades' amplitudes.The length of the path traversed by the eye is proportional to the effort expended by the HVS.

Extraocular muscle force: The amount of energy, measured in grams per degrees per second, required for the operation of extraocular muscles relates to the amount of force that each muscle applies to the eye globe during fixations and saccades. We also hypothesize that based on the Oculomotor Plant MathematicalModel[55], it is possible to extract individual extraocular muscle force values from recorded eye position points..

The total eye physical effort is approximated by:

$$E_{eye\ \_physical}\ (t) = \int_{t_0}^{t} \left(f_8(t) + f_9(t) + f_{10}(t)\right) dt$$

Where: $f_8(t)$ is a function of the saccades amplitude and number of saccades; $f_9(t)$ is a function of eye path traversed, and $f_{10}(t)$ is a function of the total amount of force exerted by the extraocular muscles. The integration includes only periods with reliable data.

In this research we represent $E_{eye\ \_physical}\ (t)$ as a vector consisting of the elements:

$$E_{eye\ \_physical}\ (t) = \begin{Bmatrix} \text{average saccade amplitude} \\ \text{saccade count} \\ \text{eye path traveresed} \end{Bmatrix}$$

Where the counts are calculated over the period $[t_0, t]$.

3.2 Learnability-Based Usability Model

The methodology proposed in this paper is centered on concepts that relate to learning and learnability. The idea is to evaluate several aspects of usability as the user completes a set of tasks originating from a single scenario. Typically, as subjects master an application, the time to complete tasks with the same scenario becomes shorter [56-58]. To illustrate, consider the following example. Assume that a set of n subjects selected at random complete a set of k tasks. Further, assume that the subjects are computer literate but unfamiliar with the application under evaluation. The objective of each task is to make travel reservations, and each task requires about the same effort. The set of k tasks have the same scenario with different data and different constraints. When plotting the Time-On-Task (TOT) averages (T_{avg}) for these subjects, a curve with a strong fit to either a power law or exponential decay curve is said to reflect learning or represents a learning curve [8, 57-59].

Selection of a model depends on how subjects learn. If a human's performance improves based on a fixed percentage, then the exponential decay curve is appropriate [59]. Using an exponential function assumes a uniform learning rate where learning everything about the software is possible. On the other hand, if a human's performance improves on an ever decreasing rate, then the power law is the appropriate choice. In this research, the power law is used because mastering computer software is measured as an ever decreasing percent of what is possible to learn. Furthermore, experience gained from our experiments supports the use of the power law. In this context, learning is modeled by time on task and/or by effort per task according to the following equation:

$$Time\ (effort) = E_{base} + \alpha + B \times N^{-\beta} \quad (1)$$

Where: E_{base} , the baseline effort, is a constant thatrepresents the minimal time or effort required for completing the task, and $\alpha \geq 0$ is a constant relating understandability. Due to understandability issues, some users may never reach the minimum time or effort required to complete a task and their performance converges to a value higher than E_{base}. B, referred to as the learning range, represents the approximate time or effort for completion of the first task. N is the number of trials. β is a positive number representing the rate of subject learning. Note that after a large number of tasks, the time or effort approaches the constant$E_{base} + \alpha$.

To further elaborate, E_{base} , the minimal time or effort required for completing the task can be associated with the expert or designer effort. That is, the level of performance of one or more experts, such as the application's designer or a person accepted as an expert on a specific Commercial-Of-The-Shelf (COTS) application. E_{exp} , is referred to as the expert effort. In many cases, an expert may not achieve a baseline (E_{base}) level of performance.

Figure 1 illustrates a usability model based on average effort to complete a set of tasks with a common scenario. Assuming that learning to an acceptable level of performance occurs during the execution of the first few tasks, the task where the subject's effort reaches this acceptable level of performance is the learning point(L_P). Summing the average task duration to the left of the learning point(L_P) indicates how much time (L_T) the average subject requires to reach an acceptable level of performance. Data to the right of the learning point (L_P) describes the amount of effort required to accomplish a task by a trained user. Learnability, Operability and Understandability are the sub-characteristics that put the subject's effort into a context and are described in the next sections.

3.3 Learnability

Learnability, the ease with which a user learns to use the software, is possibly the most critical characteristic of software usability; and it may account for many of the user complaints about software usability. All computer based tasks require some learning. Current human interface design practice doesn't always address this concept directly but usually addresses it indirectly by identifying levels of user expertise [1, 3].

It is possible to measure learnability by plotting either the average Time-On-Task (TOT) or the average Effort-On-Task (EOT), that is, the average effort (E_{avg}) expended by a group of subjects for a task, and then fitting the subjects' average performance for each task in a set to a Power Law curve [8, 56-59]. When there is a tight fit to the Power Law curve, then it is possible to say that learning has occurred. By using both the goodness of fit (R^2) and the learning rate (β), it is possible to establish measurable requirements for the software and to compare the learnability of different applications. If the effort or time a subject expends on a series of tasks has a strong fit ($R^2 \geq .7$) to a Power Law curve, then it is possible to assert that learning is observed in the evaluation. Since humans always learn, a plot of time or effort that does not produce a good fit indicates that there is another problem masking the learning. Using the learning rate, a test engineer can estimate the number of tasks necessary to evaluate the software.

Another learnability feature that can be inferred from the learning model is the learning point (L_P), which indicates that the average subject has reached an acceptable level of performance (E_L). This first requires establishing a satisfactory level of learning (r). It is possible to set the level of learning (r) as a percent of the constantE_{base}, from Equation 1. Where E_{base} represents the minimal time (effort) required for completing the task. It is then possible to calculate the acceptable effort (E_L) in the following way:

$$E_L = \frac{X}{100} \times E_{base}$$

Where: X is a number between 0 and 100 representing a percentage of mastery. When X=100, the subjects have completely mastered the set of tasks. A more realistic value for X might be 80 denoting that the subjects have reached a level of 80% of the optimal mastery. The learning point (L_P)is defined to be the first task where $E_{avg} \leq E_L$. Another

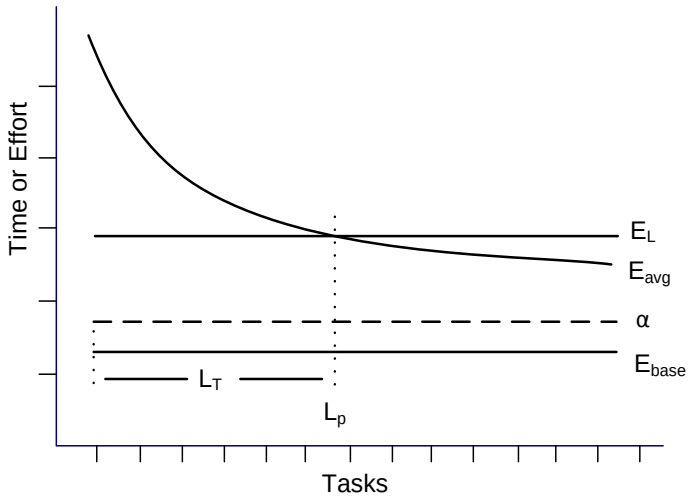

Figure 1 Performance-Based Model

method for establishing the learning point (E_L) is based on requirements provide by the stakeholders.

3.4 Effort-based Operability

Operability is the capability of a user to use the software to accomplish a specific goal. A unique feature of this research is that operability can be derived from the learning model. Consider Equation 1. Several correlated parameters derived from this equation can be used to denote operability. The term E_{base} can be used to denote the ultimate operability of a scenario. It is also possible to define operability in terms of either expert effort ($E_{\exp}$)or acceptable effort (E_L), that is, the acceptable performance of a non expert after learning the system. Recall that $E_{base} + \alpha$ is the asymptote of Equation 1 with respect to non-expert users. Hence, one can define the operability of a system as:

$$Op = E_{base} + \alpha$$

3.5 Understandability

One method of evaluating understandability is to compare the average subjects' performance on a specific set of tasks to the baseline performance, such as the designer or a person accepted as an expert. A more precise definition of Understandability with a learning model is the ratio of the operability to the baseline performance, as described in the following equation:

$$U = \frac{E_{base} + \alpha}{E_{base}}.$$

In practice, Understandability can be approximated in the following way. Let L_P denote the learning point as defined in section 3.2, then understandability is approximated by:

$$U = \frac{\frac{1}{n}\sum_{i=L_p+1}^{n}\left(E_{avg}\right)_i}{E_{exp}}$$

Where: n is the number of tasks performed. Expert effort ($E_{\exp}$) is used in the equation as the best approximation of baseline effort(E_{base}).

3.6 Relative Measures of Usability

Sections 3.3 to 3.5 defined absolute measures of usability in terms of learnability, operability, and understandability. Often, it is desired to compare the usability of two or more systems. Consider two systems, system S_1 and system S_2, where S_1 and S_2 are characterized by the learning curves $EOT_1 = E_{base\,1} + \alpha_1 + B_1 \times N^{-\beta_1}$ and $EOT_2 = E_{base\,2} + \alpha_2 + B_2 \times N^{-\beta_2}$ respectively, then each of the components of the sets: $\{E_{basei}, \alpha_i, B_i, \beta_1, L_{Pi}, E_{Li}, O_i, U_i\}$ for $i = 1,2$ can shed light on the relative usability of system S_1 compared to system S_2.

4. UTILIZING THE EFFORT BASED MEASURE APPROACH

This section presents a methodology for applying the theoretical concepts developed in section 3, in the process of user interface design and evaluation. In specific, it demonstrates the use of the effort based metrics along with the concept of the baseline effort (E_{base}) to provide interface designers and developers with a methodology to evaluate their designs as they are completed, and a usability testing technique that is applicable to both the verification and validation. This framework provides the designer with system level feedback for a specific design element. It is similar to the feedback on the quality of source provided by a unit test.

There are a few ways to estimate the baseline effort. An expert in using the software can be used as an approximation. In a new development, the best "experts" on using an application are the designers of the use case and its human interface. They can evaluate the minimal effort required for completion of each task embedded in the designed software by using a tool or rubric to estimate an ideal subject's effort or measure the effort. Their estimate of E_{base} is referred to as the designer's effort. Finally, it is possible to establish the expert effort (E_{exp}) analytically.

4.1 Designer's Effort

Often the person who is most knowledgeable about the usability of the software is the interface designer who is the actual expert. In this case, the terms expert effort and designer's effort are interchangeable.

Designer's Effort is a notion providing developers with a tool that can reduce the cost of design reviews and prototypes. It also provides the designer with feedback on the quality of the design from a usability test. One of the main benefits of the designer effort evaluation is that it provides designers with a timely low-cost method of evaluating their design and making trade-off decisions in a manner similar to those used to develop other software components.

To further illustrate, assume that a group of 10 subjects records an average Time-On-Task (TOT) of 420 seconds on a specific task. Asking whether this is a good or bad TOT is meaningless. Nevertheless, if the notion of expert effort (E_{exp}) is added to the question, then there is a basis for comparison. For example, assume that a group of 10 subjects recorded an average TOT of *x* seconds, and an expert user recorded a time of y seconds on the same task. This provides information for sound evaluation of the usability of the application. Having an expectation of the target software function and performance is also one of the fundamental principles of software testing [34].

As shown in Figure 2, the expert effort (E_{exp}) provides a reference point placing the subject data into a context, making a meaningful evaluation possible. However, comparing the performance of an expert to a group of individuals just becoming familiar with the software is not a valuable comparison. It is only possible to compare the expert's performance to that of the subjects after the subjects have passed the learning point(L_P).

Calculating the effort on a new interface design is not difficult. First, the designer should count the number of keystrokes and the number of mouse button presses. Then measure the distance necessary to move the mouse and count the number of keyboard-to-mouse interchanges. Another less tedious approach is to develop a test harness that displays the interface and a data logging utility. The test harness does not need to be functional beyond the point of operating the interface. A data logging utility, which is the same tool used in the evaluation to collect subject data, can be used by the designer. The ability to calculate effort provides the developer with this feedback mechanism. Extending the notion of effort-based interface evaluation to unit testing provides designers with feedback focused on a specific scenario. Using an eye tracking device provides additional insight into effort required by the interface.

It is also possible to use the notion of Designer's Effort in the evaluation of Commercial-Off-The-Shelf (COTS) software. An evaluation team could use either results from an expert user or an analysis of the user interface to establish the Designer's Effort. A software publisher could provide usability data on the application, but COTS evaluators may find that developing theirown data for the application provides an independent review of the software.

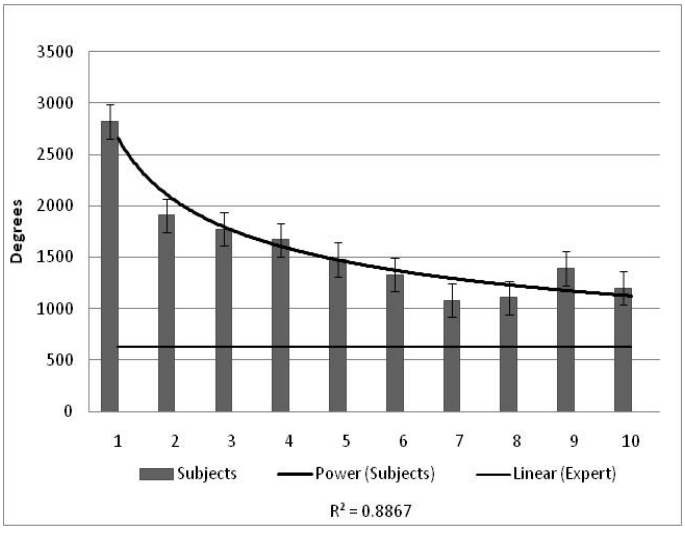

Figure 2. Designer's Effort (eye) for System B

Providing an interface designer with a technique for evaluating the ideal efficiency of the interface provides the developer with a method of evaluating designs without calling a meeting or constructing a prototype. Just evaluating manual effort ($E_{manual\_physical}$) would provide a designer with a basis for making tradeoff decisions. For example, one thing that can greatly increase effort is making a switch from the keyboard to the mouse and back. Many designers include "combo box" widgets in a design. There are several different implementations of a "combo box". Generally, they provide a drop down menu to aid the user's selection. Some implementations require the user to make their selection with a mouse button press; other implementations permit the user to type the first character until reaching their selection.
Generally, a widget employing a mouse drag and click requires more effort than one that doesn't. Using an effort-based interface evaluation, the designer can see the total effect of their design and, when possible, can select tools or objects to make the design more physically efficient.

In addition to cost effective evaluation of user interfaces, designer's effort provides an approach to establish subject understanding of the application. For example, if after learning the application, the subjects have an understanding equal to the designer's effort, then it is possible to say the subject's knowledge of the application is equal to the designer's knowledge. Normally, subjects expend more effort in completing a set of tasks than an expert, and it is possible to use the difference to express the usability / understandability of an application.

4.2 Designing the Test

There are a number of widely accepted references for designing a usability test [3, 8, 14, 20]. These references provide detailed guidelines on almost every aspect of usability testing from the laboratory design to reporting results. Nevertheless, the framework used in this paper and the focus on learning as the vehicle for deriving other usability measures necessitates additional attention in designing tests. Some of the additional considerations relate to the focus of tests on specific parts of the system, the creation of tasks that exploit the new framework, and the number of tasks that have to be conducted in order to evaluate a specific component of the system. Like any other type of test, the process for a usability test consists of preparation, execution, and analysis phases. The following section identifies the test design framework highlighting the elements that are unique to the effort based usability approach.

One of the first steps in constructing a usability test is to establish the usability requirements for the software under evaluation. At a minimum, clients should provide a profile for each user of the application and requirements for the "In Use" Quality characteristics and learnability [24]. The user profile should include characteristics such as education, experience with user interfaces, skills with a rating of expertise, etc. Describing the systems functionality using Unified Modeling Language (UML) use cases provides a focus for both specifying requirements and evaluating the software [60]. It is logical to assume that different tasks require different amount of effort than other tasks; therefore, each use case should have its own set of requirements.

After establishing requirements for each use case, the next step is to design a set of goals or tasks to evaluate a specific use case. The current method for constructing a usability tests concentrates on real world situations and uses them as the basis for designing tasks [3, 8, 14, 20]. In light of the experience gained from developing this framework, two more components are required from a test suite:

1. It has to contain tasks that allow the subject to master the use of the system before making measurements of usability.
2. It has to enable a software engineer to diagnose issues.

For this end, an approach that uses a set of test cases or tasks from a scenario based test design technique utilizing a use case diagram [60, 61] is adopted. It provides the developer with a known focus, so that issues identified in the test trace to a specific set of interface components. Designing tasks based on use cases also insures complete coverage of the application.

Many human beings learn by repeating a task or action a number of times. If tasks are identical, however, then the subjects can memorize the solution without real learning of the way to solve that class of problems. To address this issue, the developer has to create a series of tasks that are different but based on the same scenario; such a set is referred to as a set of **identical independent tasks**. Figure 3 includes the top level use case diagram for a travel reservation system. For this application, it is possible to randomly create a series of tasks of travel to different destinations under different sets of constraints, such as budget and time constraints, rental car requirements, and accommodation requirements. For example, a few tasks might require a hotel with a pool while others require internet connection. Building a series of tasks from a single scenario and providing complete coverage make it possible to construct multiple identical and independent tasks. Next, a relatively small group of subjects completing a set of identical and independent tasks would allow the developer to measure and thereby observe the learning and performance of the subjects.

With the simple example of the Travel Reservation System illustrated in figure 3, it is feasible to provide 100% coverage of all of the use cases, but in a more complex application, this discipline needs to assure coverage of all of the different scenarios. Furthermore, random selection of tasks would present a problem since it is important to select tasks that require about the same completion time (or effort). This would enable observing learning and the improvement of user performance as they master tasks.

Developing the set of tasks consists of a few steps:

1. Select a use case for evaluation.
2. Convert the input for the use case into a narrative.
3. Identify important events, conditions, or constraints and add their description to the narrative.
4. Test the scenario on all the systems that are under evaluation.

5. Replace the specifics (e.g., a constraint related to budget) of the scenario with blanks or with an option list creating a template.
6. Convert the template into a set of tasks by filling in the blanks with specific and valid data selecting a single occurrence from each option list.

Test all of the tasks on all of the systems under evaluation.

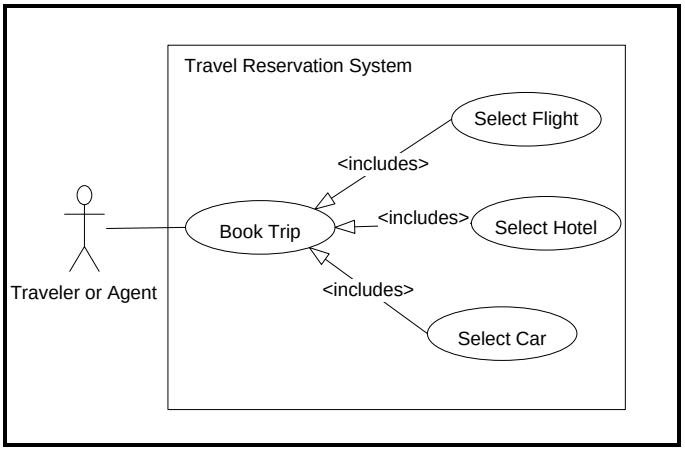

Figure 3. Use Case Diagram

Another set of question relates to the appropriate number of tasks, task length, and number of subjects. For this type of usability test, the literature suggests a number of subjects from six to twenty [3, 62]. Using the experience gained in a large set of field tests, the approach adopted for this research is to use 6 – 10 subjects (according to availability) conduct about 10 identical independent tasks and limit the duration of each task to 6-15 minutes to reduce subject fatigue. Conducting 10 tasks enables accurate identification of the learning point [63].

A small pool of 6-10 subjects permits using this approach as part of the construction phase, after the developers have completed their normal testing or as part of an iterative development process. When using this technique as part of the construction phase with scenarios without any unusual conditions, the test provides the designer with feedback about the quality of the use case early in the development process. Conducting a complete usability test is better when the software is at its most stable configuration.

The main requirement for a test facility is a minimal number of distractions. A small office or conference room is adequate. In addition, the designers would use a harness for logging user activity. There is no need for a stopwatch to record the time since the logging harness contains this information in addition to other valuable information, such as manual user activity and potential eye tracking data. While this technique is not intended to replace elaborate facilities to conduct usability tests, it can be used to complement current usability evaluations and reduce the number of elaborate testing, thereby reducing the total usability evaluation cost [14, 20]. Another novelty of this framework is the addition of a software module that measures the percent completion of each task by the user. This compares to the user perception of task completion.

The largest external expense to implement the tools and techniques discussed in this paper is the cost of acquiring subjects. Compensation for university students is less expensive and might have a number of non-monetary alternatives. A research that is currently ongoing identifies several cases where a student population can serve as a good sample for the actual system users. In other cases, however, temporary agencies can probably supply an adequate number of subjects conforming to the user profile.
The next section elaborates on a set of experiments performed to assess the utility of the new framework.

5. EXPERIMENTAL RESULTS

An experiment using two travel reservation systems (referred to as system *A* and system *B*) was conducted to ascertain the assertions of effort-based usability. For this experiment, each subject completed 10 travel reservation tasks. Ten subjects provided data for System A and 10 for System B. In addition, this experiment also provides a great deal of insight into designing and conducting a usability test.

5.1 Travel Reservation System Comparison

The data acquired for logging actual interaction and eye tracking produced a number of very important results. Trend analysis of physical effort expended by the users corresponds to the expected learning curve. In addition, the data, verified via ANOVA analysis, supports the framework's model. The following sections contain a detailed account of the results.

5.1.1 Data Reduction and Analysis

An event driven logging program is devised to obtain details of mouse and keystroke activities from the operating system event queue. The program saves each event along with a time stamp into a file. The logged events are: Mickeys, keystrokes, mouse button clicks, mouse wheel rolling, and mouse wheel clicks. In the reported experiments, the program has generated about 60,000 time stamped events per task (about 10 minutes). The eye tracking system produces anextensive log of time stamped events including parameters such as fixation duration and saccade amplitude. In addition, accurate measurement of task completion time is enabled through the eye tracking device.

A data reduction program applied to the events log, counts the total number of events (e.g., Mickeys) per task. A similar program is used for eye activity events. Both programs execute the entire data set (log of manual activity and eye activity) which consists of several millions of points in less than an hour. With 20 subjects, each completing 10 tasks, the data reduction program generates 200 data points. The data obtained from the data reduction stage is averaged per task per travel reservation system. Hence, a set of 20 points is generated where each point denotes the average count of events per task per reservation system.

5.1.2 Results and Evaluation

Figure 4 illustrates the average task-completion-time per task per system. Compared to System *B*, System *A* has a jittered trend, yet it follows a similar slope. In addition, the

task completion time for System *A* is more than twice than the completion times for System *B*. The standard deviation values

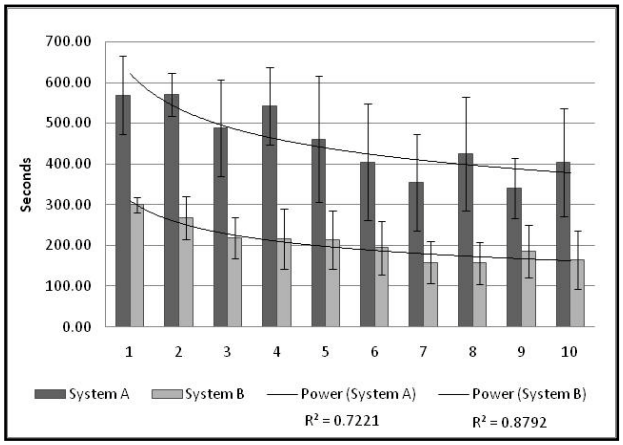

Figure 4. Average Task Completion Time

values of System *B*. System *A* and System *B* implement the same application, yet from the data presented in Figure 4X, it appears that System *B* subjects learn faster than System *A* subjects. Furthermore, the figure demonstrates that System *A* subjects are less productive than System *B* subjects. Hence, it is safe to conclude that System *B* is more operable and learnable than System *A*.

Figures 5 and 6 provide additional evidence of the usability model's soundness. Figure 5 depicts the average Mickeys per task for System B, and indicates a high correlation with the time and effort usability model.

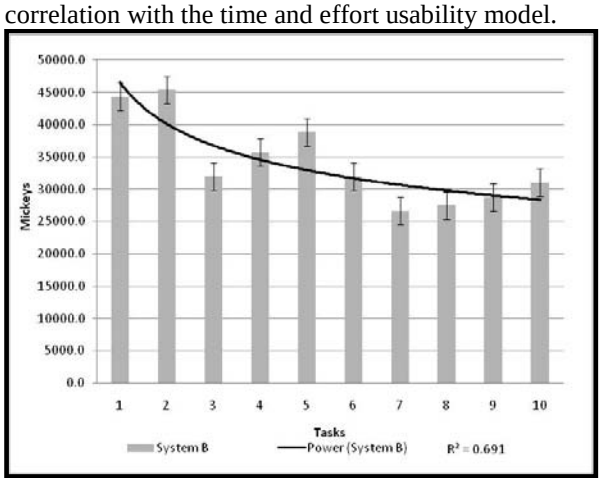

Figure 5 Average Mickeys for System B

Figure 6 depicts approximate eye physical effort by using the product of average saccade amplitude and the number of detected saccades. A strong fit to a power law curve was observed (R2=0.88) with learning point reached after the 5th task.

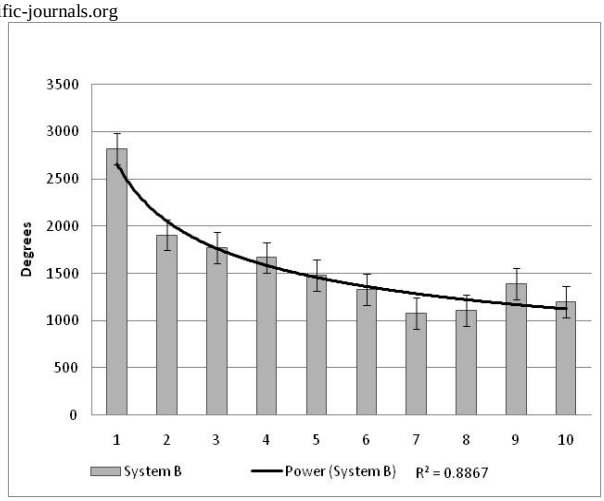

Figure 6. Approximate Eye physical effort for System B.

Like Figures 4 and 5, Figure 6 indicates an agreement with the effort-based usability model. Moreover, a spike in activity with respect to task 9 can be used as an example of the capability of the metrics to discover potential interface shortfalls. Using the usability model to discover or pinpoint usability issues is currently under investigation.

6. CONCLUSIONS AND FUTURE RESEARCH

This paper has presented an innovative framework for measuring and evaluating software usability. The framework is comprised of two major elements:

- An effort based usability model that is centered on learning.
- A coherent approach for designing and conducting software usability tests.

A learning centered model provides a vehicle to evaluate software usability and adds the capability of a direct comparison of two or more equivalent systems or different versions of the same system. Another advantage of using a learning centered model of usability is that it provides evaluators with a method of predicting subjects' performance. With the usability validation framework presented in this paper, test engineers have information to design the tasks and procedures necessary to conduct a high quality formative evaluation. The experiments presented in this paper provide objective evidence of the effectiveness of the framework.

In conducting this research, it became apparent that one of the major challenges confronting software developers is discovering the specific cause or causes of usability issues. Discovering a cause for a usability issue requires providing a developer with a set of techniques to pinpoint the specific quality characteristic and software element responsible for the user performance, such as interface component placement, instructions, and help facilities. The framework presented in this paper makes a major step to identifying the software elements involved in the use case scenario that is the basis to the tasks. A future research topic is developing a set of techniques, utilizing the usability model, to pinpoint issues

within a task, and the devices providing additional insight into the discovery of the cause of anomalies.

Another major gap in the tools for software designers that might improve the usability of the software is a better feedback mechanism. Goals, Operators, Methods, and Selection rules (GOMS) provides designers with a technique to evaluate the usability of their designs. Integrating GOMS into the effort-based usability model and the validation framework is yet another topic of future research [18].

Another direction of future research is to consider a dynamic scenario where the system adapts to the user and enables user specific improvements in usability at run time. This would permit designers to use a “flexible” interface.

REFERENCES

[1] B. Shneiderman, Designing the user interface: strategies for effective human-computer-interaction. Reading, MA: Addison Wesley Longman, Inc., 1998.

[2] I. E. Sutherland, "Sketchpad: A man-machine graphical communication system," Ph.D., Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1963.

[3] J. Nielsen, Usability Engineering. San Francisco, CA, USA: Academic Press, 1993.

[4] IEEE, "IEEE Std 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology," ed. New York, NY: Institute of Electrical and Electronic Engineers, 1990.

[5] R. N. Chartette. (2005, September) Why Software Fails. Spectrum. Available: http://www.spectrum.ieee.org/sep05/1685

[6] N. Leveson and C. S. Turner, "An Investigation of the Therac-25 Accident," IEEE Computer, vol. 26 no. 7, 1993.

[7] D. Tamir, O. V. Komogortsev, and C. J. Mueller, "An Effort and Time Based Measure of Usability," presented at the Proceedings of the 6th International Workshop on Software quality, Leipzig, Germany, 2008.

[8] T. Tullis and B. Albert, Measuring The User Experience: collecting, analyzing, and presenting usability metrics. Burlington, MA: Morgan Kaufmann, 2008.

[9] T. S. Andre, H. R. Hartson, S. M. Belz, and F. A. McCreary, "The user action framework: a reliable foundation for usability engineering support tools," Int. J. Hum.-Comput. Stud., vol. 54, pp. 107-136, 2001.

[10] N. Bevan, "Quality in use: Meeting user needs for quality," J. Syst. Softw., vol. 49, pp. 89-96, 1999.

[11] A. Blandford, T. R. G. Green, D. Furniss, and S. Makri, "Evaluating system utility and conceptual fit using CASSM," Int. J. Hum.-Comput. Stud., vol. 66, pp. 393-409, 2008.

[12] D. A. Caulton, "Relaxing the homogeneity assumption in usability testing," Behavior & Information Technology, vol. 20, p. 7, 2001.

[13] J. T. Dennerlein and P. W. Johnson, "Different computer tasks affect the exposure of the upper extremity to biomechanical risk factors," Ergonomics, vol. 49 vol 1., January 2006.

[14] J. S. Dumas and J. C. Redish, A Practical Guide to Usability Testing. Portland, OR, USA: Intellect Books, 1999.

[15] E. Folmer, J. Van Gurp, and J. Bosch, "A Framework for Capturing the Relationship between Usability and Software Architecture.," Software Process Improve and Practice, vol. 8, pp. 67-87, 2003.

[16] K. Hornbaeck, "Current practice in measuring usability: Challenges to usability studies and research," Int. J. Hum.-Comput. Stud., vol. 64, pp. 79-102, 2006.

[17] J. Howarth, T. Smith-Jackson, and R. Hartson, "Supporting novice usability practitioners with usability engineering tools," Int. J. Hum.-Comput. Stud., vol. 67, pp. 533-549, 2009.

[18] B. E. John and D. E. Kieras, "The GOMS family of user interface analysis techniques: comparison and contrast," ACM Trans. Comput.-Hum. Interact., vol. 3, pp. 320-351, 1996.

[19] P. Moore and C. Fitz, "Gestalt theory and instructional design," Journal of Technical Writing and Communication, vol. 23, pp. 137-157, 1993.

[20] J. Rubin and D. Chisnell, Handbook of Usability Testing: How to Plan , Design, and Conduct Effective Tests. Indianapolis, IN, USA: Wiley Publishing, Inc., 2008.

[21] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, "Usability measurement and metrics: A consolidated model," Software Quality Journal, vol. 14, pp. 159-178, May 13, 2006 2006.

[22] L. Vukelja, L. Müller, and K. Opwis, "Are Engineers Condemned to Design? A Survey on Software Engineering and UI Design in Switzerland," presented at the INTERACT 2007, Rio de Janeiro, Brazil, 2007.

[23] ISO, "ISO 9241-11 Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability," ed. Geneva Switzerland: International Organization for Standardization, 1998.

[24] ISO, "ISO/IEC 9126-1:2001 Software Engineering-Product Quality-Part 1: Quality Model," ed. Geneva Switzerland: International Standards Organization, 2001.

[25] ISO, "ISO/IEC 9126-2:2003 Software Engineering-Product Quality-Part 2: External Metrics," ed. Geneva Switzerland: International Standards Organization, 2003.

[26] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality," Nat'l Tech. Information Service, , 1977.

[27] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, "Usability measurement and metrics: A consolidated model," Software Quality Control, vol. 14, pp. 159-178, 2006.

[28] S. Winter, S. Wagner, and F. Deissenboeck, "A Comprehensive Model of Usability," in Engineering

Interactive Systems: EIS 2007 Joint Working Conferences, EHCI 2007, DSV-IS 2007, HCSE 2007, Salamanca, Spain, March 22-24, 2007. Selected Papers, ed: Springer-Verlag, 2008, pp. 106-122.

[29] M. E. Fagan, "Design and code inspections to reduce errors in program development," IBM Systems Journal, vol. 15, pp. 182-211, 1976.

[30] D. Gries, The Science of Programming: Springer-Verlag New York, Inc., 1987.

[31] R. Pressman, Software Engineering: A Practitioner's Approach, 7th ed. New York, NY.: McGraw-Hill, 2010.

[32] B. Beizer, Software Testing Techniques 2nd Edition. Boston, MA: Thomson Inter-Science, 1990.

[33] E. Kit, Software Testing in the Real World. Reading, MA: Addison-Wesley, 1995.

[34] G. Myers, The Art of Software Testing. New York, NY: John Wiley & Sons, 1979.

[35] C. Mills, K. F. Bury, T. Roberts, B. Tognazzini, A. Wichansky, and P. Reed, "Usability testing in the real world," presented at the Proceedings of the SIGCHI conference on Human factors in computing systems, Boston, Massachusetts, United States, 1986.

[36] M. B. Skov and J. Stage, "Supporting problem identification in usability evaluations," presented at the 17th Australia conference on Computer-Human Interaction, Canberra, Australia, 2005.

[37] R. Fitzpatrick. (1999, March, 2009). Strategies for Evaluating Software Usability. Available: http://www.comp.dit.ie/rfitzpatrick/papers/chi99%20strategies.pdf

[38] J. Nielsen. (2005, 1 March 2009). Heuristic Evaluation. Available: http://www.useit.com/papers/heuristic/

[39] W. W. Royce, "Managing the development of large software systems: concepts and techniques," presented at the Procedings of IEEE WESCON, 1970.

[40] D. M. Hilbert and D. F. Redmiles, "Extracting usability information from user interface events," ACM Comput. Surv., vol. 32, pp. 384-421, 2000.

[41] IEEE, "IEEE Std 730-2002 Standard for Software Quality Assurance Plans," ed. New York, NY: IEEE, 2002.

[42] IEEE, "IEEE Std 1012-2004 Standard for Verification and Validation," ed. New York, NY: IEEE, 2004.

[43] G. Solomon, "Television watching and mental effort: A social psychological view," in Children's understanding of television, J. Bryant and D. Anderson, Eds., ed New York: Academic Press, 1983.

[44] J. L. Andreassi, Psychophysiology: Human Behavior and Physiological Response, 3rd ed. Hillsdale, NJ: Lawrence Erlbaum, 1995.

[45] C. S. Ikehara and M. E. Crosby, "Assessing Cognitive Load with Physiological Sensors," presented at the Hawaii International Conference on System Sciences, 2005.

[46] A. Duchowski, Eye Tracking Methodology: Theory and Practice, 2nd ed.: Springer, 2007.

[47] R. Jacob and K. Karn, "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises.," in The Mind's Eyes: Cognitive and Applied Aspects of Eye Movements, R. Radach, J. Hyona, and H. Deubel, Eds., ed: Elsevier Science, 2003.

[48] M. Crosby, M. Iding, and D. Chin, "Visual Search and Background Complexity: Does the Forest Hide the Trees?," in User Modeling 2001, ed, 2009, pp. 225-227.

[49] P. M. Fitts, R. E. Jones, and J. L. Milton, "Eye movements of aircraft pilots during instrument-landing approaches," Aeronautical Engineering Review, vol. 9, pp. 24-29, 1950.

[50] S. P. Marshall, "The Index of Cognitive Activity: measuring cognitive workload," in Human Factors and Power Plants, 2002. Proceedings of the 2002 IEEE 7th Conference on, 2002, pp. 7-5-7-9.

[51] D. Kahneman, Attention and effort. Englewood Cliffs, NJ: Prentice-Hall, 1973.

[52] J. H. Goldberg and X. P. Kotval, "Computer interface evaluation using eye movements: methods and constructs," International Journal of Industrial Ergonomics, vol. 24, pp. 631-645, 1999.

[53] J. Goldberg, H., M. J. Stimson, M. Lewenstein, N. Scott, and A. Wichansky, M., "Eye tracking in web search tasks: Design implications.," in Proceedings of the Eye Tracking Research and Application Symposium, 2002, pp. 51-58.

[54] S. Fuhrmann, O. Komogortsev, and D. Tamir, "Investigating Hologram-based Route Planning," Transactions of Geographical Information Science, p. in press, 2009.

[55] O. V. Komogortsev and J. Khan, "Eye Movement Prediction by Oculomotor Plant Kalman Filter with Brainstem Control," Journal of Control Theory and Applications, vol. 7, 2009.

[56] H. Ebbinghaus. (1885). Memory: A Contribution to Experimental Psychology. Available: http://psy.ed.asu.edu/~classics/Ebbinghaus/index.htm

[57] A. C. Hax and N. S. Majluf, "Competitive cost dynamics: the experience curve," Interfaces, vol. 12, pp. 50-61, October 1982.

[58] T. P. Wright, "Factors Affecting the Cost of Airplanes," Journal of Aeronautical Sciences, vol. 3, pp. 122-128, 1936.

[59] F. E. Ritter and L. J. Schooler, "The Learning Curve," in Internatonal Encyclopedia of Social & Behavioral Sciences, ed: Elsevier Science Ltd., 2001.

[60] J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language Reference Manual. Reading, MA: Addison Wesley Longman, Inc., 1999.

[61] C. Kaner. (2003, December, 2008). An Introduction to Scenario Testing. Available: http://www.testingeducation.org/a/scenario2.pdf

[62] J. Nielsen. (2008, December, 2008). Logging Actual Use. Available: http://www.usabilityhome.com/FramedLi.htm?Logging.htm

[63] O. Komogortsev, C. Mueller, D. Tamir, and L. Feldman, "An Effort Based Model of Software Usability," presented at the 2009 International Conference on Software Engineering Theory and Practice (SETP-09), Orlando, FL, 2009.

AUTHOR PROFILES

Dr. Tamir (http://cs.txstate.edu/~dt19/) is an associate professor of Computer Science at the Texas State University, San Marcos, Texas (2005 - to date). He obtained the PhD-CS from Florida State University in1989, and the MS/BS-EE from Ben-Gurion University, Israel.From 1996-2005, he managed applied research and design in DSP Core technology in Motorola-SPS/Freescale. From 1989-1996, he served as an assistant/associate professor in the CS Department at Florida Tech. Between 1983-1986, he worked in the applied research division, Tadiran, Israel. Dr. Tamir is conducting funded research in power aware task scheduling, combinatorial optimization, computer vision, and data compression. He has been a member of the Israeli delegation to the MPEG committee and a Summer Fellow at NASA KSC.

Dr. Mueller obtained a PhD-CS with research in automated software testing from Illinois Institute of Technology under the supervision of Dr. Bogdan Korel. Dr. Mueller has over 7 years of teaching experience and more than 35 years of industrial experience specializing in developing and testing safety critical and/or high reliability applications (medical devices, telephony, and other applications). Currently, Dr. Mueller is conducting research into software development security and authentication.

Dr. Komogortsev received the degree in computer science from Kent State University in 2007. He is conducting research in the areas of Biometrics, Human Computer Interaction, Eye tracking, and Bioengineering.His research has been funded by the National Science Foundation, National Institute of Standards, Sigma Xi the Scientific Research Society,and Emerson Management. More information about Dr. Komogortsev's research can be found at http://cs.txstate.edu/~ok11/