Usability of User Interface Styles for Learning Graphical Software Applications

Andisheh Feizi *feiziandisheh@yahoo.com*
Interface Design Department, Faculty of Creative Multimedia
Multimedia University
Cyberjaya, 63100, Malaysia

Chui Yin Wong *cywong@mmu.edu.my*
Interface Design Department, Faculty of Creative Multimedia
Multimedia University
Cyberjaya, 63100, Malaysia

Abstract

This paper examines usability of different user interface styles for learning graphical software applications, namely Adobe Flash CS4 and Microsoft Expression Blend 4. An empirical study was performed to investigate the usability attributes of effectiveness, efficiency and satisfaction scores for learning the graphical software applications. There were 32 participants recruited whom consist of interface designers and software developers. A set of 7 tasks was designed to compare the different effects of user interface styles including graphical user interface (GUI) and command line interface (CLI). User Performance variables (effectiveness, efficiency, duration, number of errors and number of helps) were measured for tasks performed by all the participants in the test. Satisfaction score was measured using QUIS (Questionnaire for User Interface Satisfaction) tool. The result revealed that the average effectiveness scores are higher than 75% for both software applications. Although Adobe Flash CS4 gained slightly higher on effectiveness, Microsoft Expression Blend 4 obtained better results in terms of efficiency, duration, errors and helps. The user satisfaction rates also showed Microsoft Expression Blend 4 gained higher satisfaction comparing Adobe Flash CS4. Generally, both software applications gained scores above average (>3.5) for majority of the user interface satisfaction attributes of software regardless of users' background.

Keywords: Usability, User Interface Styles, Graphical User Interface (GUI), Command Line Interface (CLI), Graphical Software Application.

1. INTRODUCTION

User Interfaces (UIs) have been around since the invention of computers, even before the field of Human-Computer Interaction (HCI) was initiated [1]. Users carry out information communication efficiently with computers through User Interface (UI) to complete their tasks [2]. Since UI design is an important component of HCI system [2], great burden has been on software designers to create interfaces that effectively predict and interpret the operator's needs besides allowing the user to perform tasks in natural ways [3].

Throughout the last four decades, programming has evolved from platforms with great amount of difficulties and constrains to enter, read and debug a programme into command language strategies, and eventually into the approach of Graphical User Interface (GUI) [4]. In the context of programming, user interface design plays an important role [5].

The development of GUI software applications has been one of the noteworthy improvements in programming field that reduce the difficulty of remembering syntax and semantics with the guidance of menu-based interactive properties it delivers [6]. However, there are still circumstances that require the users to use command-line interface (CLI) since CLIs often

afford more options than their equivalent GUIs, leading to greater flexibility available for users or one can perform a task by using command that its function is not supported by its GUI counterpart [7]. Command Line Interfaces (CLI) are considered quite inconvenient environment for new generation of users since they are substantially used to GUIs [8]. Hence, it is important to examine usability level of different user interface styles of using GUI and/or CLI in learning graphical software applications.

The International Organization for Standardization and the International Electro Technical Commission ISO/IEC 9126-1 classify software quality attributes into six categories: functionality, usability, reliability, efficiency, maintainability and portability [9]. Usability is increasingly accepted as a significant quality factor for interactive software systems like GUI style applications, Web sites, and variety of mobile interactive services [10]. Benson et al. [11] consider usable software *"a win–win situation for developers, corporations, and the users"*.

In regards of the issue of users learning a software application to produce new knowledge, usability is considered as an essential attribute for quality of software design. We are concerned with the usability of software packages because nowadays, large numbers of people use applications at work and for personal tasks as well. These users desire to learn software to meet their professional needs. Some may use a software application frequently or occasionally, but they do not use it intensively, as clerical workers do [12]. So, they rarely become experts in the use of software [12].

Having analyzed the above-mentioned issues, this paper aims to examine the usability of different user interface design styles including GUI and CLI for learning a graphical software application by interface designers and software developers. The followings are the objectives of this research:

- To evaluate usability attributes in terms of user performance measure (effectiveness, efficiency, time duration, number of errors, and number of helps) for two competing graphical software applications;
- To evaluate user satisfaction for user interfaces of two competing graphical software applications;
- To examine usability of different user interface styles (i.e. GUI and CLI) for learning a graphical software application.

2. LITERATURE REVIEW

Users "have contact with an information system only with the help of an interface that defines information flow rules between a human and a machine" [13]. Lauesen [14] defines the user interface (UI) as organizing and designing screens in a way that user can easily understand and efficiently utilize the system. User Interface Design (UID) refers to the "overall process of designing how a user will be able to interact with a system" [15]. UID concerns about "facilitating clear and accurate information exchanges, efficient transactions, and high-quality collaborative work" [16].

Software user interface is an essential medium for information transmission between users and computers for successfully performing various tasks, besides designing new software products [17]. Software user interface features will dramatically influence the user's efficiency and attitude towards it [18]. It is the user interface of a computer program, which provides users with the perception of what a user interface can do and how to do it [19]. User interface design is a fundamental concern for the usability of a software product [20] and is also one of the significant concerns in HCI field [17].

In Human-Computer Interaction (HCI) discipline, researchers in the field mainly focus on five 'E' of usability, which propose an interactive system must be 'effective, efficient, engaging, error tolerance, and easy to learn'. HCI is a branch of human factors field, which involves user interface design, human-computer communications, and user engagement [21]. The goal is to provide users with information systems (i.e. software interface) and work environments in which they can do their tasks efficiently [22]. Despite the importance of

usability in software development, it is still insufficient in majority of software applications [23] [24]. The IFIP Working Group comments that 'there are major gaps of communication between the fields of HCI and software engineering (SE) [25]. Since the most prevalent perspective in the field of SE is that usability is mainly related to the UI [23] [26]. Some mentioned usability primarily concerns the UI rather than the system's core. However, Juristo et al. [27] demonstrate that usability is not confined to the interface and can affect the core functionality of a system. They believe usability is associated with the entire user–system interaction, not just the UI [27].

2.1 Usability of Software Design

Based on research from theoretical and practical perspectives in software field, some guideline standards established for clarifying the usability of software products [28] [29] [30]. Usability is defined as "the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component" [31]. Gould [32] categorizes usability into system performance, system functions, and user interface. McCall et al. [33] outlines usability as operability, training and communicativeness. Booth [34] explains that usability has four attributes as usefulness, effectiveness, learnability, and attitude. Hix et al. [35] classify usability into performance, learnability, retainability, first impression, and long-term user satisfaction. Software Usability Measurement Inventory (SUMI) determines usability in terms of efficiency, effectiveness, helpfulness, control and learnability [36]. Donyaee et al. [37] established quality in use integrated measurement (QUIM) model including attributes as effectiveness, efficiency, satisfaction, productivity, safety, accessibility, and internationality. Battleson et al. [38] discusses that to enhance usability, an interface must be easy to learn, use, and remember with few errors for its intended users. Sauro et al. [39] planned a 'single and summated' usability metric for each task by averaging four values for task time, errors, completion, and satisfaction. Shneiderman [40] claims that 'a clever design for one community of users may be inappropriate for another community' and 'an efficient design for one class of tasks may be inefficient for another class'.

Bevan [29] proposed a detailed description for the term of usability, which considers effectiveness, efficiency, and satisfaction as quality factors for usability. Usability is defined, in ISO 9241-11, as the extent to which a software product can be employed by particular users to achieve specific objectives with satisfaction, efficiency and effectiveness within a certain context of use [41]. Subsequently, ISO 9126-1 explains usability in terms of learnability, operability, understandability, and attractiveness [42].

Nielsen [43] proposed one of the popular definitions for usability involving the learnability and memorability of a software program, its capacity to avoid and control user errors, its efficiency of use and user satisfaction. Technically speaking, "efficient use of the computer is intrinsic to usability" [44]. To the sense that in assessing the usability of a software product, it is required to examine user performance in addition to considering the amount of effort a user puts in applying the software. Therefore, a system is not usable if it requires high amount of effort in order to complete a task with a high performance [44]. The most prevalent perspective in the field of software engineering (SE) is that usability is intertwined with the user interface [23]. Shneiderman et al. [45] determine features of user interface design based on evaluation of several human factors such as length of time to learn, learner's rate of errors, pace of performance, user's satisfaction, and retention over time. However, despite the significance of usability in software development, it is still unsatisfactory in majority of software programs [23].

Measuring software usability is a significant indicator of the deficiency level of software application, and software testing is the foundation for software usability enhancement [2]. Software usability is not directly measurable; it can be simply evaluated indirectly through observing measures, such as effectiveness, user's satisfaction and performance assessment [46].

Software tools used for usability evaluation have been available since 1980s. They consisted of two groups, questionnaire tools measuring user's perception and satisfaction (e.g. QUIS) and behavioral data collection software to capture and record user's performance (e.g. Camtasia) [47]. The procedure usability practitioners establish includes (1) observing subjects individually in real-time session to collect instant physical and verbal behaviors, (2) obtaining

performance measures such as number of errors, and time on task (3) comparing two or more systems, designs, or product features (usually from competitors (4) performing statistical analysis of the collected data to justify product design [47].

2.2 Categories of User Interface Styles

The style of human–computer information flow within a single-user interface is determined by the application of interactions [48]. Generally, basic interaction styles include command-line languages, filling forms, menus, direct manipulation, and natural language [48]. According to International Business Machines [49] user interfaces can be categorized into three fundamental groups:

(i) Command line user interface that is a full-text display mode on a computer screen controlled by a keyboard, in which users type in data, commands or instructions notifying the computer to do a task. A common example of a Command Line Interface (CLI) is UNIX-based that text is only shown on the entire screen [50].

(ii) Menu-Driven user interfaces, "in which a user is provided with a hierarchically organized set of choices" [6]. Robertson et al. [51] mention that users fail to correctly perform a task on a menu when structure of the menu is complex. However, Gray [52] believes that such a result can be regarded to the psychological issues with user interfaces and the limitation of learner's short-term memory. In a menu-based environment, a user clicks on a command from pre-defined array of commands exhibited in menus. If command names on the menus are understandable and well organized, users can easily perform their tasks since discovering a command in a menu is equivalent to recognition instead of recall [53]. This type of interface is ideal for novice learners as they support error handling; however, they can appeal to expert users if arrangement and selection processes are quick enough as well as convenient shortcuts are provided. On the other hand, possibly menus are slow for regular users besides the fact that numerous numbers of menus may result in overload and too much complexity.

(iii) Graphical User Interfaces (GUIs), which is an interactive human-computer interface that makes use of widgets including windows, icons, menus, buttons, dialog boxes and etc. It is often directly manipulated by a computer mouse, and to a limited extent by a keyboard [7]. The widgets are basic visual blocks combined in an application that hold all the data processed by the application and the available interactions required to achieve goals of the user. Users can interact with information by manipulating visual widgets provided; according to the kind of data they hold [54]. GUIs are direct manipulation systems currently familiar to users in the Windows environment [55].

Nowadays, we have become so accustomed to interact with a Graphical User Interface (GUI) since it makes it easier for people to work with computer software regardless of their computer skills [56].

Researchers working with students regarding learning programming languages came to conclusion that interface of software applications play an important role in quality of learning and efficiency, and the learning process should be underpinned by a rich programming environment [5]. According to Shneiderman [57], employment of Direct Manipulation Interfaces (DMI in which GUIs are included) has reinforced the accuracy and diminished errors, besides facilitating learning. Another investigation on the influence of interface styles on perceived ease of use and usefulness came to the conclusion that menu-based interface was more beneficial rather than command-based interface [58]. Davis et al. [59] compared DMI and CLI styles. Their results indicated no significant distinction on perceived ease of use. Davis [60] looked at user perception in using a text editor and electronic mail applications finding out that system features had considerable effect on ease of use. Wiedenbeck et al. [61] examined DMI, menu-driven and CLI. Their outcome showed that interface style did not affect participants' perception towards the usefulness of the system, however, DMI style was considered easier to use by participants. Moreover, Gururajan et al. [62] investigated on icon-based and menu-based interfaces claiming that interface style has no considerable influences on ease of use. Shneiderman [6] states users can track down information more quickly with GUIs compared to CLIs. Besides that, a user's understanding and satisfaction is higher for

GUI applications. Additionally, Faulkner [55] expresses that there is an evidence to confirm humans recall pictures better than words [22].

However, the most popular UI for software today is the GUI [63]; McGraw [64] points out, even GUIs can bring difficulties to navigate and use. Virvou et al. [65] describes that users of current GUIs may repeatedly find themselves brought into problematic situations even without realizing it. Testing GUIs produces many challenges, due to the enormous number of possible combinations of commands that can be executed on the GUI. Testing all possible orderings of events is not practical. Alternatively, testers of GUI applications attempt to limit the number of test cases that need to be executed [63].

3. RESEARCH METHODOLOGY

We conducted a usability testing to evaluate the usability attributes of comparing different user interface design (i.e. GUI and CLI) of learning graphical software applications using Adobe Flash CS4 and Microsoft Expression Blend 4. The research method is an experimental study with a mixture of observation, user interface satisfaction questionnaire (QUIS) and user testing for data collection. The QUIS questionnaire was adapted for this research purpose since it is a validated instrument for conducting comparative evaluations for software applications [66].

3.1 Rationale of Graphical Software Applications

There have been several popular graphical software applications in the market that integrate graphical user interface and programming scripting functions all-in-one for interface designer and software developer to work seamlessly for software development work. Adobe Flash CS4 and Microsoft Expression Blend 4 are selected for the study because they are competitor software specifically designed by Adobe™ and Microsoft to bridge the development platform for interface designers and developers/programmers. They combine GUI and CLI in the software applications and enable the developers and designers to work apart more effectively without losing each other's work in a software development process.

Generally, Adobe Flash and Microsoft Expression Blend are authoring tools that can be utilized to design and create presentations, software applications that act in response to user's interactions. Projects created by them can contain animation, video content and complex user interfaces. On one hand, they allow software developer to use Command Languages (a scripting language) to create functions and determine how the elements in the application act, and the code also allows adding interactivity and logic in a project.

3.2 Apparatus and Testing Facility

This study was conducted at User Interface Lab. The apparatus use for the user testing is a laptop (with a 14 inch monitor, 4 GB RAM, 2.20 GHZ CPU having 1280 * 800 display resolution) as a workstation, Windows Vista Home version as operating system, and Adobe Flash CS4 and Microsoft Expression Blend 4 as graphical software tools to carry out the task sets, and Camtasia Studio 3 was employed to record the screen capture for data analysis.

3.3 Participants and Tasks

32 participants were recruited with the background of interface design or programming for the study. The participants were tested individually. The user testing took around an average of 1 to 1.5 hours. Upon arrival, all participants were given a consent form before the experiment commenced. They were then asked to complete a demographic and software products experience questionnaire. Then, the participants were randomly given 10-minute trainings to learn the basic conventions of Adobe Flash CS4 and Microsoft Expression Blend 4. The participants are also given an average of 7 minutes to practice on their own and gain confidence before taking the tests. Subsequently, all participants were given a set of 7 tasks (Table 1) to perform in the software applications. If the participants were unable to complete a task, they were free to proceed without task completion. Figure 1 and 2 show an example of screen shot for Task 4 and Figure 3 and 4 indicate an example of screen shot for Task 6.

TABLE 1: Tasks During User Testing.

Task number	Adobe Flash CS4		Microsoft Expression Blend 4	
	Task type	Task explanation	Task type	Task explanation
1	GUI	Set background color	GUI	Set background color
2	GUI	Create text	GUI	Create text
3	GUI + CLI	Create animation for text	GUI	Create animation for text
4	GUI	Import image to the file	GUI	Import image to the project
5	GUI	Place image on screen, resize it and make it symbol	GUI	Place image on screen and resize it
6	CLI	Create Mouse Over event for image	GUI	Create Mouse Over event for image
7	CLI	Change image transparency	CLI (with suggestion)	Change image transparency

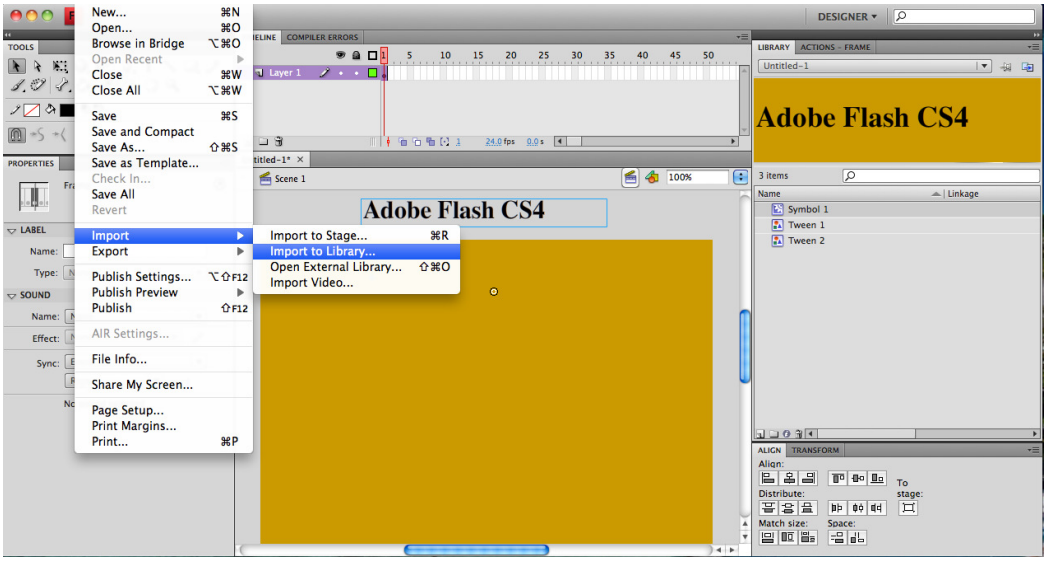

FIGURE 1: Task 4 (import image to the file) screenshot in Adobe Flash CS4.

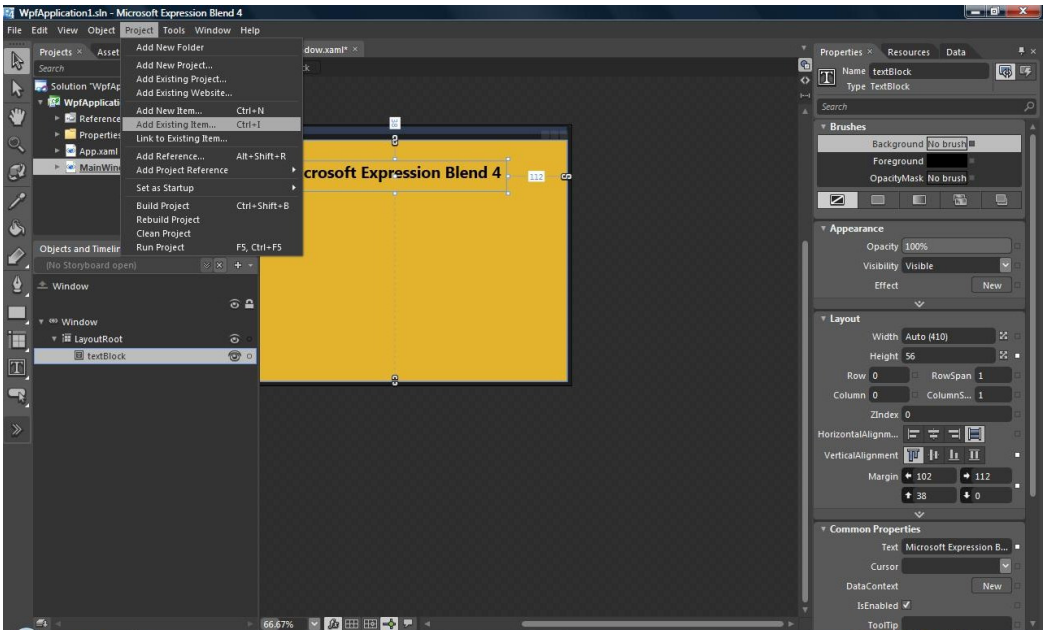

FIGURE 2: Task 4 (import image to the file) screenshot in Microsoft Expression Blend 4.

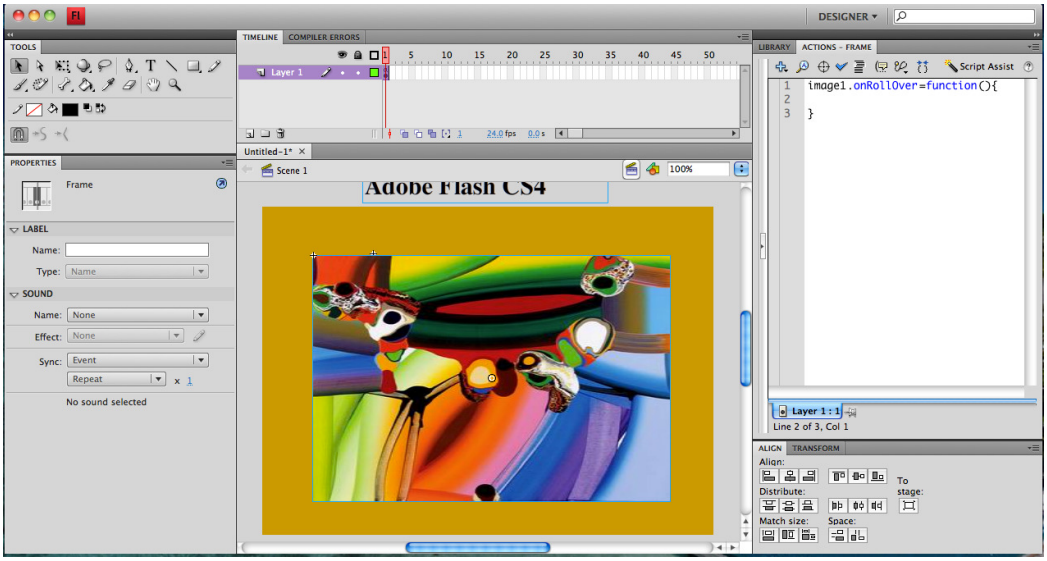

FIGURE 3: Task 6 (create mouse over event for image) screenshot in Adobe Flash CS4.

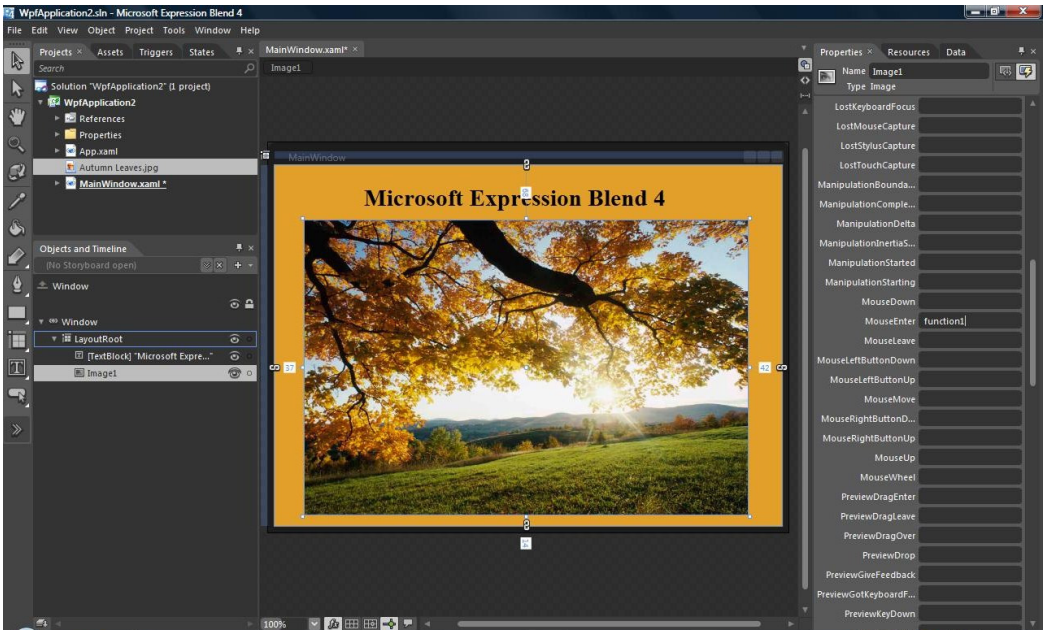

FIGURE 4: Task 6 (create mouse over event for image) screenshot in Microsoft Expression Blend 4.

3.4 Usability Metrics

The User Performance Variables for usability as follows:

- Effectiveness score: Percentage of successful completion for each task.
- Time taken: The total time spent to complete each task.
- Efficiency Rate: It is calculated by dividing effectiveness score by the time taken to do the task.
- Error: Any error made during performing each task.
- Help: Any help received during performing each task.

For subjective satisfaction, QUIS questionnaires measure users' subjective satisfaction using a 7-point semantic differential scale on the interfaces of both software applications. Each questionnaire covered items such as overall reaction, screen, terminology and software feedback, learning, and software capabilities.

4. RESEARCH RESULTS

The result is analyzed using a statistical analysis software, SPSS 16. The data gathered from the user testing were analyzed using descriptive analysis with mean and standard deviation. The rationale was to compare effects of two different GUI styles and CLI styles on learning Adobe Flash CS4 and Microsoft Expression Blend 4. The performance measure of each task is calculated for all users in the test.

4.1 User Performance Analysis

Among all 32 users, 17 (53.13%) participants were designers with interface design background while 15 (46.88%) participants were programmers from IT background. Users were from both genders (22 male and 10 female). Users were asked about their knowledge and usage of Adobe Flash and Microsoft Expression Blend, 28 (87.5%) individuals already had training for Adobe Flash while 4 individuals did not have (12.50%), and 5 (15.63%) individuals already had training for Microsoft Expression Blend while 27 (84.38%) individuals did not have any training.

One-Sample Kolmogorov-Smirnov test was conducted to examine normal distribution of effectiveness scores, duration, efficiency, number of errors and helps for all 7 tasks. Table 2 shows the results of user performance measure for Adobe Flash CS4 test. Table 3 indicates the result of user performance measure for Microsoft Expression Blend 4.

TABLE 2: A summary of usability testing results for Adobe flash CS4.

Task no.	Effectiveness (%)	Time Duration (Sec)	Efficiency	Error	Help
1	100	10.84	14.08	0.19	0.13
2	100	15.13	7.86	0.0	0.0
3	92.97	54.38	2.23	0.44	0.31
4	100	12.34	9.12	0.0	0.06
5	98.44	39.94	2.99	0.16	0.06
6	85.94	47.47	2.74	0.25	0.63
7	91.41	21.63	5.85	0.22	0.34
Mean*	95.54	28.81	6.41	0.18	0.22
SD**	5.54	18.06	4.3	0.15	0.22

Mean* indicates average for total 7 tasks.

SD*=standard deviation

The result in Table 2 shows that the effectiveness scores higher than 75% for all the 7 tasks (Mean for total task=95.54%, SD=5.54). However, the minimal effectiveness score is associated with CLI (Task 6 of creating Mouse Over event) and all GUI tasks are linked with maximum effectiveness score. On one hand, the minimum time duration taken to complete the tasks is 10.84 min (Task 1 of setting background color), which also means the task is fairly simple to achieve. On another hand, the maximum time duration is 54.38 min (for Task 3), which indicates the task is more complex and challenging to complete in terms of creating animation using GUI and CLI technique. The average time for 7 tasks is 28.81 minutes (SD=18.06).

In terms of efficiency, the total tasks score for 6.41 in average. Task 1 again achieves the highest efficiency rate of 14.08; however, the least efficient task to complete is Task 3 (2.23). This indicates that a task that combines GUI and CLI will take longer time for task completion. Apart from this, the average error rate for an overall task accounts for 0.18, which is considered minimal, and Task 3 shows the highest score of making mistakes. The number of help accounts for 0.22 for the total task, which is almost acceptable for task performance. The highest number of seeking help is Task 6, which shows the users are not familiar with creating a mouse over event for an image using CLI approach.

TABLE 3: A summary of usability testing results for Microsoft Expression Blend 4.

Task no.	Effectiveness (%)	Time Duration (Sec)	Efficiency	Error	Help
1	92.67	27	5.74	0.25	0.34
2	100	20.71	5.5	0.06	0
3	96.09	42.46	2.88	0.15	0.15
4	92.96	18.25	7.07	0.18	0.12
5	98.43	18.12	6.59	0.03	0.03
6	84.37	12.25	14.61	0.18	0.12
7	94.53	19.78	7.75	0.15	0.09
Mean*	94.15	22.65	7.16	0.14	0.12
SD**	5.09	9.76	3.63	0.07	0.11

Mean* indicates average for total 7 tasks.
SD*=standard deviation

The result in Table 3 demonstrates that the effectiveness score is higher than 75% for all the 7 tasks again (Mean for total task=94.15%, SD=5.09). However, the minimal effectiveness score is associated with GUI (Task 6 of creating Mouse Over event) and CLI task is nearly linked with average effectiveness score. The minimum time duration taken to complete the tasks is 12.25 min (Task 6 of creating Mouse Over event), meaning the task is rather effortless to achieve. But then again, the maximum time duration is 42.46 min (for Task 3), which indicates the task is intricate and challenging to complete in terms of creating animation via GUI. The average time for 7 tasks is 22.65 minutes (SD=9.76). Regarding efficiency, the total tasks score for 7.16 in average. Task 6 achieves the highest efficiency rate of 14.61; however, the least efficient task to complete is Task 3 (2.88). This shows that even a GUI task can be so much complex taking longer time for task completion. Apart from this, the average error rate for an overall task accounts for 0.15, which is considered minimal, and Task 1 shows the highest score of making mistakes (Mean=0.25). The number of help accounts for 0.12 for the total task, which is absolutely acceptable for task performance. The highest number of seeking help is Task 1, which shows the users are not familiar with setting background color using GUI approach in Microsoft Expression Blend 4.
Performance variables gathered from user testing was analyzed by parametric (Independent-Samples T-Test) and non-parametric (Mann-Whitney) tests to compare usability of the two applications. Below are the results:

4.1.1 Effectiveness

The effectiveness score of each task is calculated for every user in tests. Tasks 2, 3, 5, 6 and 7 did not show any significant difference ($p>0.05$) comparing two applications. Regarding Task 1: setting background color, Mann-Whitney test shows effectiveness score was significantly higher for Adobe Flash CS4 (Mean = 100) rather Microsoft Expression Blend 4 (Mean = 92.67), $U = 416$, $Z = -2.55$, $p = 0.011$. Regarding Task 4: importing image to the project, Mann-Whitney test indicates effectiveness score was significantly higher for Adobe Flash CS4 (Mean = 100) rather than Microsoft Expression Blend 4 (Mean = 92.96), $U = 432$, $Z = -2.3$, $p = 0.021$.

The total numbers of effectiveness for the two software applications were compared by Mann-Whitney test. The result showed no significant difference for Adobe Flash CS4 and Microsoft Expression Blend 4 ($U = 469$, $Z = -0.6$, $p = 0.54$).

4.1.2 Duration

Each task duration is calculated for every user for both tests. Tasks 3 and 7 did not show any significant difference ($p>0.05$) comparing two applications. Regarding Task 1: setting background color, Independent-Samples T-Test shows less time duration for Adobe Flash CS4 ($M = 10.84$, $SD = 11.65$) rather than Microsoft Expression Blend 4 ($M = 27$, $SD = 16.56$) condition; $t(55.65) = 4.51$, $p = 0.000$. Regarding Task 2: creating text, Independent-Samples T-Test shows less time duration for Adobe Flash CS4 ($M = 15.13$, $SD = 7.27$) comparing to Microsoft Expression Blend 4 ($M = 20.71$, $SD = 7.64$) condition; $t(62) = -2.99$, $p = 0.004$. Regarding Task 4: importing image to the project, Independent-Samples T-Test shows less time duration for Adobe Flash CS4 ($M = 12.34$, $SD = 4.62$) comparing to Microsoft Expression Blend 4 ($M = 18.25$, $SD = 10.58$) condition; $t(42.42) = -2.89$, $p = 0.006$. Regarding Task 5: placing image on screen, Independent-Samples T-Test shows less time duration for Microsoft Expression Blend 4 ($M = 18.12$, $SD = 8.94$) rather than Adobe Flash CS4 ($M = 39.94$, $SD = 18.25$) condition; $t(45.05) = 6.07$, $p = 0.000$. Regarding Task 6: creating Mouse Over event for image, Independent-Samples T-Test shows less time duration for Microsoft Expression Blend 4 ($M = 12.25$, $SD = 10.06$) comparing to Adobe Flash CS4 ($M = 47.47$, $SD = 30.11$) condition; $t(38.92) = 6.21$, $p = 0.000$.

The total time duration for the two software applications were compared by Independent-Samples T test. The result showed that time duration for Adobe Flash CS4 ($M = 201.72$, $SD = 68.44$) is higher than Microsoft Expression Blend 4 ($M = 156.50$, $SD = 49.71$); $t(56.58) = 3.02$, $p = 0.004$.

4.1.3 Efficiency Rate

Each task efficiency rate was calculated for every user for both tests. Tasks 3 and 7 did not show any significant difference ($p>0.05$) comparing two applications. Regarding Task 1:

setting background color, Independent-Samples T-Test shows higher efficiency for Adobe Flash CS4 (M = 14.08, SD = 6.89) comparing Microsoft Expression Blend 4 (M = 5.74, SD = 4.39) condition; t (62) = 5.76, p = 0.000. Regarding Task 2: creating text, Independent-Samples T-Test shows higher efficiency for Adobe Flash CS4 (M = 7.86, SD = 3.11) comparing Microsoft Expression Blend (M = 5.5, SD = 2.19) condition; t (62) = 3.5, p = 0.001. Regarding Task 4: importing image to the project, Independent-Samples T-Test shows higher efficiency for Adobe Flash CS4 (M = 9.12, SD = 3.11) comparing Microsoft Expression Blend 4 (M = 7.07, SD = 4.16) condition; t (62) = 2.23, p = 0.029. Regarding Task 5: placing image on screen, Independent-Samples T-Test shows less efficiency for Adobe Flash CS4 (M = 3.01, SD = 1.36) comparing Microsoft Expression Blend 4 (M = 6.59, SD = 2.85) condition; t (44.46) = - 6.39, p = 0.000. Regarding Task 6: creating Mouse Over event for image, Independent-Samples T-Test shows less efficiency for Adobe Flash CS4 (M = 2.74, SD = 1.96) comparing Microsoft Expression Blend (M = 14.61, SD = 11.02) condition; t (32.96) = -4.83, p = 0.000.

The total efficiency rates for the two software applications were compared by Independent-Samples T test. The result showed that efficiency rate for Microsoft Expression Blend 4 (M = 0.67, SD = 0.25) is higher than Adobe Flash CS4 (M = 0.53, SD = 0.17); t (62) = -2.63, p = 0.011.

4.1.4 Errors

Number of errors is counted for every user related to do every single task for both tests. Tasks 1, 2, 3, 4, 5, 6 and 7 did not show any significant difference (p>0.05) comparing two applications.

The total number of errors during tests for the two software applications were calculated and compared by Mann-Whitney test. The result showed no significant difference for Adobe Flash CS4 and Microsoft Expression Blend 4 (U = 473, Z = -0.55, p = 0.57).

4.1.5 Helps

Number of helps is calculated for every user seeking for assistance when performing every single task for both tests. Tasks 1, 2, 3, 4 and 5 did not show any significant difference (p>0.05) comparing two applications. Regarding Task 6: creating Mouse Over event for image, Mann-Whitney test showed the number of helps was significantly higher for Adobe Flash CS4 (Mean = 0.63) rather than Microsoft Expression Blend 4 (Mean = 0.12), U = 250, Z = -4.02, p = 0.000. Regarding Task 7: changing image transparency, the number of helps was significantly higher for Adobe Flash CS4 (Mean = 0.34) rather than Microsoft Expression Blend (Mean = 0.09), U = 398.5, Z = -2.18, p = 0.029.

The total numbers of helps for the two software applications were compared by Mann-Whitney test. The result showed the number of helps was higher for Adobe Flash CS4 (Mean = 0.22) rather than Microsoft Expression Blend 4 (Mean = 0.12); U = 347, Z = -2.31, p = 0.021.

4.2 QUIS Questionnaire Analysis

For subjective user satisfaction, data collected through QUIS questionnaire at the end of the test is summarized below (see Table 4). Users' satisfaction for Adobe Flash CS4 and Microsoft Expression Blend 4 software were measured on a 7-point semantic differential scale. Software applications were ranked by users from different aspects (i.e. overall software performance, screen, terminology and software feedback, learning and software capabilities).

TABLE 4: QUIS questionnaire result analysis

	Adobe Flash CS4		**Microsoft Expression Blend 4**	
Items	**Mean**	**Standard deviation**	**Mean**	**Standard deviation**
Category: Overall software performance				

TERRIBLE-WONDERFUL	5.16	1.19	5.94	0.88
DIFFICULT-EASY	4.63	1.34	5.34	1.21
INEFFICIENT-EFFICIENT	5.34	1.12	5.53	0.94
UNFRIENDLY-FRIENDLY	4.69	1.47	5.69	1.02
FRUSTRATING-SATISFYING	4.56	1.41	5.78	0.94
INEFFECTIVE-EFFECTIVE	5.59	1.07	5.72	0.89
RIGID-FLEXIBLE	4.87	1.43	5.94	1.22
Category: Screen				
ONSCREEN INFORMATION (Inadequate-Adequate)	4.91	1.63	5.16	1.27
USER INTERFACE ARRANGEMENT (Not organized-Organized)	5.34	1.28	5.47	1.48
EASY TO FIND FUNCTIONS (Never-Always)	4.38	1.48	4.56	1.46
READING CHARACTERS (Difficult-Easy)	5.38	1.10	5.34	1.49
SCREEN BACKTRACK (Difficult-Easy)	5.81	1.35	6.16	1.30
CREATING NEW PROJECT (Confusing-Very clear)	6.19	1.09	5.94	1.16
TOOLBAR ACCESS (Difficult-Easy)	5.91	1.28	5.75	1.32
Category: Terminology and software feedback				
SIMPLE AND NATURAL DIALOGUE (Never-Always)	4.74	1.29	4.94	1.32
TERMS USED IN THE SOFTWARE (Inconsistent-Consistent)	5.5	1.28	5.69	1.35
POSITION OF WINDOWS DIALOG BOX (Inconsistent-Consistent)	5.31	1.28	5.69	1.12
INFORMS ABOUT WORK PROGRESS (Never-Always)	4.48	1.71	5.41	1.36
ERROR MESSAGES (Unhelpful-Helpful)	3.97	1.97	5.62	2.07
PROMPT FOR DOING SCRIPTING INPUT (Confusing-Clear)	3.35**	1.64	5.66	1.54
Category: Learning				
SOFTWARE LEARNING (Difficult-Easy)	4.47	1.70	5.75	1.05
EXPLORING BY TRIAL AND ERROR (Difficult-Easy)	3.65*	1.70	5.19	1.78
REMEMBERING	4.13	1.56	5.5	1.46

COMMANDS (Difficult-Easy)				
PERFORMING TASKS IS SIMPLE (Never-Always)	4.50	1.19	5.5	1.11
HELP ACCESS OR DOCUMENT (Difficult-Easy)	4.25	1.8	6.28	1.61
HELP MESSAGES ON SCREEN (Unhelpful-Helpful)	4.31	1.97	5.94	1.13
Category: Software capabilities				
CORRECTING MISTAKES (Difficult-Easy)	4.13	2.06	5.31	1.60
DESIGNED FOR ALL LEVELS OF USERS (Never-Always)	3.19**	1.55	4.62	1.86
IMPORT AND EXPORT PROJECT IN AND OUT OF SOFTWARE (Difficult-Easy)	5.38	1.41	5.87	11.38
SOFTWARE RELIABILITY (Unreliable-Reliable)	5.37	1.27	6.03	1.15

**<3.5; *just above 3.5

For Adobe Flash CS4, the average user satisfaction for the overall software performance, screen, terminology and software feedback, learning and software capabilities are 4.99, 5.41, 4.72, 4.51 and 4.62 respectively. In general, the overall subjective user satisfaction for Adobe Flash CS4 scores (Mean=4.88) above average of >3.5 for all the above-mentioned 5 categories. However, under the category of 'Software capabilities', 'Designed for all levels of users (Never-Always)', it scored only 3.19, which is lower than the average level (3.5). Again, under the category of 'terminology and software feedback', 'prompt for doing scripting input (Confusing-Clear)' rates only 3.35 score, which is also lower than the average acceptance level (3.5). This user feedback is consistent and proven by the CLI performance result of having more difficulties of performing Task 6 and 7 that requires prompts for doing scripting input in Adobe Flash. The application reaches its highest convenience in terms of screen due to the familiar and clear menu labels, ease of toolbar access, organized interface arrangement and ease of reading characters. Furthermore, it obtains its lowest satisfaction in terms of software learning due to the difficulty of exploring by trial and error, remembering commands and help material.

Regarding Microsoft Expression Blend 4, the average user satisfaction for the overall software performance, screen, terminology and software feedback, learning and software capabilities are 5.71, 5.48, 5.5, 5.69 and 5.33 respectively. Overall, the subjective user satisfaction for Microsoft Expression Blend 4 scores (Mean=5.55) above average of >3.5 for all the categories. The application reaches its highest convenience in terms of overall software performance due to its flexibility, effectiveness, satisfactory and wonderful user experience. Furthermore, it obtains its lowest satisfaction scores in terms of software capabilities due to the difficulty of correcting mistakes, and lack of design for all levels of users.

5. DISCUSSIONS

This study evaluates the usability aspect in terms of user performance and satisfaction towards two graphical software applications, Adobe Flash CS4 and Microsoft Expression Blend 4, among user interface designers and software programmers. Users' performance in terms of effectiveness, efficiency, task duration, errors and number of help on working with GUI and CLI of Adobe Flash CS4 and Microsoft Expression Blend 4 were investigated in the usability testing. It showed that users could easily pick up the interfaces' functionalities when some training was given.

5.1. User Satisfaction

To examine user satisfaction in usability tests, QUIS questionnaires were used on Adobe Flash CS4 and Microsoft Expression Blend 4. The finding showed that Microsoft Expression Blend 4 and Adobe Flash CS4, both scored higher than average acceptance level. Although Microsoft Expression Blend 4 gained higher satisfaction rates in terms of overall software performance, terminology and software feedback, learning and software capabilities, they score almost the same in terms of screen. In regards of sub-categories, Adobe Flash CS4 gained higher rank than its counterpart for some 'screen' sub-categories (i.e. Reading Characters, Creating new project and Toolbar access).

5.2. User Performance

In terms of user performance variables for the usability test of both applications, CLI task in Adobe Flash CS4 was associated with higher duration, errors and helps and less efficiency rate comparing to its equivalent GUI in Microsoft Expression Blend 4. Another CLI task in Adobe Flash was linked with less effectiveness, efficiency and higher duration, errors and helps comparing to its equivalent CLI with suggestion task in Microsoft Expression Blend. A combination of GUI and CLI task in Adobe Flash again scored less effectiveness, efficiency and higher duration, errors and helps comparing to its equivalent GUI task on Microsoft Expression Blend. It is consistent with Wisher et al. [67] claiming that failing to remember just one of the essential facts leaves some tasks unachievable. This result is consistent with Wiedenbeck et al. [61] considering Direct Manipulation Interface easier to use as compared to menu-driven and CLIs. In addition, Shneiderman [6] claiming that users could track down information more quickly with GUI as compared to CLI. The result is also consistent with Gunderloy [68] stating that learning and using GUI software is easy and effortless, Schneiderman's [57] study on DMI interfaces and another investigation on benefits of menu-based interface rather than command-based interface [58].

Comparing GUI tasks in two applications, GUI tasks including same step of completion scored the same in terms of effectiveness and number of helps despite the differences in interface design for both software applications. However, finding the right icon to perform the task in Microsoft Blend took more time and is considered less efficient with higher number of errors due to the less efficacy icon design on Microsoft Expression Blend comparing to Adobe Flash. A GUI task on Microsoft Expression Blend having intricacy in completion scored less effectiveness and efficiency, but higher duration, errors and help comparing to the same GUI task with a standard design on Adobe Flash. Another GUI task on Microsoft Expression Blend performed by clicking on an unusual label of a menu, gained less effectiveness and efficiency, and higher duration, errors and helps compared to the same GUI task in Adobe Flash with standard and common menu label. Last but not least, a GUI task in Microsoft Blend consisting of fewer steps rather than its equivalent in Adobe Flash is associated with less duration, errors and helps and higher efficiency rate.

5.3. Comparison of User Task Performance for Both Graphical Software

Concerning Task 1 (Set background color) users simply need to change the color of background via the properties menu available to them in Adobe Flash CS4 workspace, While for Microsoft Expression Blend 4, they first have to select the background item in the object menu in order to activate the properties menu and complete the task. Comparing performance measures for Task 1 for both applications indicates that Adobe Flash is more successful since the completion of task requires less steps besides the fact that it made use of the ordinary method of performing such a task, therefore, individuals could learn to carry out the task more efficiently. However, Microsoft Expression Blend implementing two distinct menus in properties menu and switching between them using a small icon designed in the menu is confusing and baffling for the users.

Concerning Task 2 (Create text) users need to click on the text icon from the tools menu and start typing a text. Comparing performance measures for Task 2 for both applications shows that Microsoft Expression Blend was associated with lower efficiency due to the weak design of the icon. It took more time for users to find the proper icon on the related menu in Microsoft Expression Blend. Additionally, the rate of errors was higher in finding the text icon in the before-mentioned application.

Concerning Task 3 (Create animation for text) users are supposed to perform 5 steps in Adobe Flash (consisting of both CLI and GUI) and 3 steps in Microsoft Expression Blend to achieve a same result. Comparing performance measures for Task 3 for both applications reveals that Microsoft Expression Blend owns more usable design for this task. First of all, it calls for fewer steps. Secondly, it is done only via GUI. Thirdly, it made use of familiar labels comparing to confusing procedure, similar labels and options with different actions in Adobe Flash.

Concerning Task 4 (Import image) users need to find the proper label from the applications' menu. Comparing performance measures for Task 4 for both applications indicates that Adobe Flash again scored better due to the use of common labels while Microsoft Expression Blend is less usable as it offers variety of options through very similar labels.

Regarding Task 5 (Place image on screen, resize it) users gained better scores using Microsoft Expression Blend, as it requires fewer number of steps to achieve the result, while in Adobe Flash users made more mistakes due to the complex basic design feature of the application (i.e. necessity of converting every object to a specific kind of symbol in order to create events or animation). Therefore it is difficult for a user to get the basic idea of working with the software.

Concerning Task 6 (Create Mouse Over event for image) users need to write a line of command using Adobe Flash whilst they are just required to find the correct event label from list of events in properties menu using Microsoft Expression Blend. Comparing performance measures for Task 6 for both applications shows that Microsoft Expression Blend is more usable and efficient. Higher number of errors and helps for Adobe Flash arise from the fact that Individuals mostly forget pieces of script. Besides, writing a command line from memory takes much more time comparing to recognizing a label.

Concerning Task 7 (Change image transparency) that requires typing a command, users completed the task with fewer errors and help using Microsoft Expression Blend since it offers a list of suggested properties while typing a code. Adobe Flash was associated with less efficiency due to the unfamiliar property's name, syntax and scripting language.

Having examined the effects of participants' background study on Adobe Flash CS4 and Microsoft Expression Blend 4 tests, results showed no significant difference for interface designers and programmers on user performance variables; in other words, performing tasks in both GUI and CLI did not have significant difference on user performance among programmers and interface designers. More precisely, programmers carried out CLI tasks with the same performance as interface designers despite their background of study when the programming language is a new one for them. Moreover, investigating effects of prior knowledge of Adobe Flash and Microsoft Expression Blend did not show any significant difference on usability performance measures as well. This fact is associated with the skills decay after a long period of time and weak usability of both software applications as far as knowledge retention is concerned.

6. CONCLUSIONS AND FUTURE WORK

The goal of this research was to compare the impacts of different interface styles (GUI and CLI) on software applications for interface designers and developers. CLI were found to be more difficult to learn and less ease of use, even for software developers as well as designers. However, GUI was perceived to be simpler to learn for both groups; however, when it comes down to unfamiliar menu labels or icons difficulty to find, users can easily make the mistake of selecting wrong menu items. Moreover, for procedural tasks with higher number of levels, users more likely forget the series of steps due to the dependency to recall issues. For the software application to be usable for end-users, employment of familiar menu labels and toolbars, less number of steps necessary for accomplishment of tasks, and providing GUI equivalent for CLI tasks are highly recommended. Besides, experience of software will be more satisfying if it is designed for larger group of users not only professional ones. Utilizing commands without input prompting is where most of difficulties arise in usability of software. Nevertheless, we still witness development and implementation of many

new software applications by reputable companies undergoing the same flaws prevalent in the past.

Making use of standard icon designs or metaphors is of a great help for users to learn new software effortless, designing new metaphors for an application creates more memory load as opposed to the necessity of the reduction of memory burden to gain new knowledge. Arranging a group of related properties in one menu is highly recommended, whilst representing a great number of options in one menu creates a perplexing workspace for the user. Software interface designers are highly advised to plan a well-organized workspace without redundancy of options having distinct functioning. Users easily get mixed up with superfluity of information. They are also suggested to avoid similar labels providing various actions. Software designers are encouraged to employ the preceding logics implemented in earlier versions of applications instead of planning the whole new structure and forcing users to acquire entire new skills. Additionally, software developers are advocated to take advantages of GUI to the most possible extent as an alternative to CLI. In a situation that CLI cannot be evaded; employing CLI with suggestions is far more usable rather than CLI per se. However, in the context of GUI per se, efficient organization of options, using common labels and icons, procedure of performing tasks similar to popular software applications, and minimizing the complexity of performing tasks by reducing the number of required steps; are essential factors to achieve higher level of software usability.

All in all, software applications are basic tools for developers and interface designers to create new software. Therefore, it is important to ensure usability of software applications in terms of meeting the needs of their users. Since user interface is the basis for all interactions between users and applications. Thus, it is important that software applications have to be usable and provide more enjoyable experience that put users in control of interface and reduce their memory load.

Future studies will investigate various CLI or GUI applications by themselves. Looking at comparable CLI/GUI applications marketed by independent companies and studying the effects of their design styles on performance and retention of skill would be an essential affair for the next generation of software applications.

7. REFERENCES

1. A.H. Jorgensen and B.A. Myers. "User Interface History." Proceeding CHI '08 Extended Abstracts on Human Factors in Computing Systems, pp. 2415-2418, 2008.

2. P. Zhou and X. Fang. "Analysis of Cognitive Behavior in Software Interactive Interface." Computer-Aided Industrial Design and Conceptual Design, CAID/CD 9th International Conference, pp. 113-116, 2008.

3. C. Frankish, P. Morgan and R. Hull. "Recognition Accuracy and Usability of Pen-Based Interfaces." IEE Savoy Place. Vol. 126, pp. 7/1- 7/6, 1996.

4. J.A. Jacko and A. Sears. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. New York: Taylor & Francis Group, 2002.

5. M. Fetaji, S. Loskovska and B. Fetaji. "Software Engineering Interactive Virtual Learning Environment," in Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications. C. Montgomerie and J. Seale, Ed. Chesapeake., VA: AACE , 2007, pp. 939-944.

6. B. Shneiderman. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Reading, MA: Addison Wesley, 1998.

7. The Linux Information Project. "GUI Definition". Internet: http://www.linfo.org/gui.html, [September 6, 2010].

8. A.O. Ajayi, E.A. Olajubu, D.F. Ninan, S.A. Akinboro and H.A. Soriyan. "Development and

Testing of a Graphical FORTRAN Learning Tool for Novice Programmers." Interdisciplinary Journal of Information, Knowledge, and Management, vol. 5, pp. 277-289, 2010.

9. International Standardization Organisation. "International Standard ISO/IEC 9126-1." Software Engineering—Product Quality—Part 1: Quality Model, 1st ed. pp. 9–10, 2001.

10. A. Seffah, M. Donyaee, R.B. Kline and H.K. Padda. "Usability Measurement and Metrics: A Consolidated Model." Software Qual J., vol. 14, pp. 159–178, 2006.

11. C. Benson, M. Muller-Prove and J. Mzourek. "Professional Usability in Open Source Projects: GNOME, OpenOffice.org, NetBeans." Proceedings of the CHI '04 Extended Abstracts on Human Factors in Computing Systems, pp. 1083-1084, 2004.

12. S. Davis and S. Wiedenbeck. "The Effect of Interaction Style and Training Method on End User Learning of Software Packages." Interacting with Computer, vol. 11, pp. 147–172, 1998.

13. R. Michalski, J. Grobelny and W. Karwowsk. "The Effects of Graphical Interface Design Characteristics on Human–Computer Interaction Task Efficiency." International Journal of Industrial Ergonomics, vol. 36. pp. 959–977, 2006.

14. S. Lauesen. User Interface Design: A Software Engineering Perspective. Addison Wesley: Reading MA, 2005.

15. Foviance. "Glossary" Internet: http://www.foviance.com/glossary/u/user-interface design, [December 20, 2010].

16. J.K. Burgoon, J.A. Bonito, B. Bengtsson, C. Cederberg, M. Lundeberg and L. Allspach. "Interactivity in Human–Computer Interaction: a Study of Credibility, Understanding, and Influence." Elsevier Science Publishers, vol. 16, pp. 553-574, 2000.

17. U. Jamil, T. Mustafa, A.R. Sattar, Shafia and F. Shahzad. "Cognitive Analysis of Software Interfaces." European Journal of Scientific Research, vol. 1, pp. 99-108, 2010.

18. X. Pan and Y. Lu. "Study of CAID Software User Interface Design Based on Usability." Computer-Aided Industrial Design and Conceptual Design, 9th International Conference, 2008, pp. 209-213.

19. K. Horvath and M. Lombard. "Social and Spatial Presence: An Application to Optimize Human-Computer Interaction." Psychology Journal, vol. 8, pp. 87–114, 2010.

20. R. Oppermann. "User-Interface Design." Internet: http://fit.fraunhofer.de/~oppi/publications/UserInterfaceLearningSystems.pdf, 2002 [September 6, 2010].

21. J. M. Carey. Human Factors in Information Systems: An Organizational Perspective. NJ: Ablex, Norwood, 1991.

22. G.A. Berg. "Human-Computer Interaction (HCI) in Educational Environments: Implications of Understanding Computers as Media." Journal of Educational Multimedia and Hypermedia. vol. 9. pp. 347-368, 2000.

23. A. Seffah and E. Metzker. "The Obstacles and Myths of Usability and Software Engineering." Communications of the ACM, vol. 47, pp. 71–76, 2004.

24. R. G. Bias and D.J. Mayhew. Cost-Justifying Usability: An Update for the Internet Age. San Francisco: Morgan Kaufmann, 2005.

25. IFIP Working Group 2.7/13.4. "On User Interface Engineering: Bridging the SE & HCI Communities" Internet: http://www.se-hci.org/bridging/index.html, [August 12, 2008].

26. E. Folmer, J. Group and J. Bosch. "Architecting for usability: A Survey." Journal of Systems and Software, vol. 70, pp. 61–78, 2004.

27. N. Juristo, A.M. Moreno and M. Sanchez-Segura. "Analysing the Impact of Usability on Software Design." The Journal of Systems and Software. vol. 80, pp. 1506–1516, 2007.

28. W. Dzida. "International User-Interface Standardization," in The Computer Science Engineering Handbook. J.A.B. Tucker, Ed. Florida: Boca Raton, CRC Press, 1997, pp. 1474–1493.

29. N. Bevan. "Human-Computer Interaction Standards." Elsevier Science B., vol. 20, pp. 349-354, 1995.

30. M. Macleod. "Usability in Context: Improving Quality of Use." Internet: http://www.nigelbevan.com/papers/mm- con94.pdf, [May 12, 2011].

31. IEEE Std. 1061. IEEE Standard for a Software Quality Metrics Methodology. New York: IEEE Computer Society Press, 1992.

32. J.D. Gould. "How to design usable systems," In Handbook of Human Computer Interaction. M. Helander, Ed. New York: Elsevier, 1988, pp. 757-789.

33. J.A. McCall, P.K. Richards and G.F. Walters. Factors in Software Quality. Springfield, VA: National Technical Information Service, 1977.

34. P. Booth. An Introduction to Human-Computer Interaction. USA: Lawrence Erlbaum Associates Publishers, Hillsdale, 1989.

35. D. Hix and H.R. Hartson. Developing User Interfaces: Ensuring Usability Through Product & Process. New York: John Wiley, 1993.

36. M. Porteous, J. Kirakowsky and M. Corbett. SUMI User Handbook. University College Cork: Human Factors Research Group, 1993.

37. M. Donyaee and A. Seffah. "QUIM: An Integrated Model for Specifying and Measuring Quality in Use," Eighth IFIP Conference on Human Computer Interaction, Tokyo, Japan, 2001.

38. B. Battleson, A. Booth and J. Weintrop, "Usability Testing of an Academic Library Web Site: A Case Study." The Journal of Academic Librarianship, vol. 27, pp. 188-198, 2001.

39. J. Sauro and E. Kindlund. "A Method to Standardize Usability Metrics Into a Single Score." CHI '05, pp. 401–409, 2005.

40. B. Shneiderman, Designing the User Interface, Strategies for Effective Human–Computer Interaction. USA: Addison Wesley, 2004.

41. ISO/DIS 9241-11. Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) Part 11: Guidance on Usability. 1998.

42. International Standard ISO/IEC 9126-1. Software Engineering—Product Quality—Part 1: Quality Model. 2001, pp. 9–10.

43. J. Nielsen. Usability Engineering. Boston, London: Academic Press, 1993.

44. N. Bevan and I. Curson. "Methods for Measuring Usability." Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction, 1997, pp. 672-673.

45. B. Shneiderman and C. Plaisant. Designing the User Interface, Strategies for Effective Human–Computer Interaction. Reading, MA: Addison Wesley, 2004.

46. A. Raza, L.F. Capretz and F. Ahmed. "Users' Perception of Open Source Usability: An Empirical Study." Engineering with Computers, vol. 28, pp. 109-121, 2001.

47. S. Rosenbaum. "The Future of Usability Evaluation: Increasing Impact on Value." Maturing Usability, pp. 344-378, 2008.

48. R. Michalski, J. Grobelny and W. Karwowsk. "The Effects of Graphical Interface Design Characteristics on Human–Computer Interaction Task Efficiency." International Journal of Industrial Ergonomics, vol. 36, pp. 959–977, 2006.

49. International Business Machines. "The library for system solutions End User Interface reference." Internet: http://www.redbooks.ibm.com/redbooks/pdfs/gg244107.pdf, [May 13, 2011].

50. The Linux Information Project. "GUI Definition." Internet: http://www.linfo.org/gui.html, 2004 [September 6, 2010]

51. O. Robertson, D. McCracken and A. Newell. "The ZOG Approach to Man- Machine Communication." International Journal of Human-Computer Studies, vol. 51, pp. 279-306, 1999.

52. J. Gray. "The Role of Menu Titles as a Navigational Aid in Hierarchical Menus." SIGCHI Bulletin, vol. 17, pp. 33-40, 1986.

53. M. Soegaard. "Interaction Styles." Internet: http://www.interaction design.org/encyclopedia/interactionstyles.html, [November 18, 2010].

54. D. Higgins. "Widget." Internet: http://whatis.techtarget.com/definition/0,,sid9_gci213364,00.html, [September 6, 2010].

55. C. Faulkner. The Essence of Human-Computer Interaction. New York: Prentice Hall, 1998.

56. S. Passini, F. Strazzari and A. Borghi. "Icon-Function Relationship in Toolbar Icons." Elsevier, vol. 29, pp. 521-525, 2008.

57. B. Shneiderman. "Direct Manipulation: A Step Beyond Programming Languages." Computer, vol. 16,pp. 57-69, 1983.

58. B. Hasan and M.U. Ahmed. "Effects of Interface Style on User Perceptions and Behavioral Intention to Use Computer Systems." Computers in Human Behavior, vol. 23, pp. 3025–3037, 2007.

59. S. A. Davis and R.P. Bostrom. "An Experimental Investigation of the Roles of the Computer Interface and Individual Characteristics in Learning of Computer Systems." International Journal of Human–Computer Interaction, vol. 4, pp. 143–172, 1992.

60. F.D. Davis. "User Acceptance of Information Technology: System Characteristics, User Perceptions and Behavioral Impacts." International Journal of Man–Machine Studies, vol. 38, pp.457–487, 1993.

61. S. Wiedenbeck and S. Davis. "The Influence of Interaction Style and Experience on User Perceptions of Software Packages." International Journal of Human–Computer Studies, vol. 46, pp. 563–588,1997.

62. R. Gururajan and D. Fink. "A Study of Influences of Application Interfaces on End User Training Outcomes." Internet: http://www.informingscience.org/proceedings/IS2002Proceedings/pa pers/Guruj098Study.pdf, [May 17, 2011].

63. P.A. Brooks and A.M. Memon. "Automated GUI Testing Guided By Usage Profiles." ACM, pp. 333-342, 2007.

64. K.L. McGraw. "Performance Support Systems: Integrating AI, Hypermedia and CBT to Enhance User Performance." Journal of Artificial Intelligence in Education, vol. 5, pp. 3–26, 1994.

65. M. Virvou and K. Kabassi. "An Empirical Study Concerning Graphical User Interfaces that Manipulate Files," Proceedings of ED-MEDIA 2000, World Conference on Educational Multimedia, Hypermedia and Telecommunications, AACE, Charlottesville, VA., 2000, pp. 1724–1726.

66. J.P. Chin, V.A. Diehl and K.L. Norman. "Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface," in CHI '88 Conference Proceedings: Human Factors in Computing Systems, Association for Computing Machinery. J. J. O'hare, Ed. New York, 1988, pp. 213-218.

67. R.A. Wisher, M.A. Sabol and J.A. Ellis. Staying Sharp: Retention of Military Knowledge and Skills(ARI Special Report 39). Alexandria, VA: U.S. Army Research Institute for the Social and Behavioral Sciences, 1999.

68. M. Gunderloy. Developer to Designer: GUI Design for the Busy Developer. Alameda, CA: Joel Fugazzotto, 2005.