Non-invasive keyboard fatigue monitoring system for improving user performance and reducing incidences of Repetitive Strain Injuries

Satheesh Jayakumar

University of New Hampshire, Durham

Follow this and additional works at: https://scholars.unh.edu/thesis

Recommended Citation
https://scholars.unh.edu/thesis/330

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact nicole.hentz@unh.edu.
Non-invasive keyboard fatigue monitoring system for improving user performance and reducing incidences of Repetitive Strain Injuries

Abstract
Computers are ubiquitous in their application and deployment all over the world. Along with their universal appeal and versatility they also pose dangers to their various users in the form of ailments such as Repetitive Strain Injuries, Carpals Tunnel Syndrome, etc. which are all specifically related to keyboard use. The objective of this thesis was to explore the possibility of developing a deterministic and non-invasive method of detecting keyboard fatigue. A software application was developed which allowed us to reliably monitor this as a function of the latency between keyboard key-press and key-release events recorded by the resident operating system.

The latency trends that were observed through testing on three volunteers proved that the average latency calculated increased steadily with the onset of fatigue. Hence by estimating a threshold condition it was possible to train the system to estimate the fatigue level of the users and warn them appropriately at a considerably early stage of the condition.

Keywords
Engineering, Electronics and Electrical, Health Sciences, Occupational Health and Safety
NON-INVASIVE KEYBOARD FATIGUE MONITORING SYSTEM FOR IMPROVING USER PERFORMANCE AND REDUCING INCIDENCES OF REPETITIVE STRAIN INJURIES

BY

SATHEESH JAYAKUMAR
B.E, University of Madras, India (2004)

THESIS

Submitted to the University of New Hampshire in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Electrical Engineering

December, 2007
This thesis has been examined and approved.

Thesis Director, John R. LaCourse, Professor of Electrical and Computer Engineering

Kondagunta U. Sivaprasad, Professor of Electrical and Computer Engineering

L. Gordon Kraft, III, Professor of Electrical and Computer Engineering

12/13/07 Date
DEDICATION

This thesis is dedicated to my family and the Department of Electrical & Computer Engineering, University of New Hampshire, Durham.
ACKNOWLEDGEMENTS

I would like to first express my heartfelt thanks to Dr. John R. LaCourse for his guidance, encouragement, patience and understanding during the entire course of this thesis effort right from conception and design to final testing. I would also like to thank my thesis committee members whose advice and feedback steered this research project in the right direction.

I am also grateful to some of the recent alumni of the Department of Electrical Engineering namely Baghi and Chandra who gave me valuable insight into this avenue of research from their respective thesis experiences. Lastly, I would like to thank all the volunteers who participated in this research project for taking considerable time out of their busy schedules to help me drive this work to conclusion.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEDICATION</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
<tr>
<td>CHAPTER 1</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Repetitive Strain Injuries</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Micro-breaks</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Motivation and purpose</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Scope</td>
<td>8</td>
</tr>
<tr>
<td>1.5 Thesis Organization</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER 2</td>
<td>10</td>
</tr>
<tr>
<td>SOFTWARE ARCHITECTURE</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Choice of Development Environment</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Architecture Overview</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Logger</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1 Requirements</td>
<td>13</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.3.2 Program Flow</td>
<td>14</td>
</tr>
<tr>
<td>2.3.3 Algorithm/Pseudo-code</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Analyzer</td>
<td>19</td>
</tr>
<tr>
<td>2.4.1 Requirements</td>
<td>19</td>
</tr>
<tr>
<td>2.4.2 Program Flow</td>
<td>20</td>
</tr>
<tr>
<td>2.4.3 Algorithm/Pseudo-code</td>
<td>21</td>
</tr>
<tr>
<td>CHAPTER 3</td>
<td>25</td>
</tr>
<tr>
<td>HUMAN ASPECT</td>
<td>25</td>
</tr>
<tr>
<td>3.1 Overview</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Testing Logistics</td>
<td>26</td>
</tr>
<tr>
<td>3.3 Testing Procedure</td>
<td>26</td>
</tr>
<tr>
<td>3.4 Graphical User Interface</td>
<td>28</td>
</tr>
<tr>
<td>CHAPTER 4</td>
<td>35</td>
</tr>
<tr>
<td>RESULTS</td>
<td>35</td>
</tr>
<tr>
<td>4.1 Volunteer 1 Observations</td>
<td>36</td>
</tr>
<tr>
<td>4.2 Volunteer 2 Observations</td>
<td>42</td>
</tr>
<tr>
<td>4.3 Volunteer 3 Observations</td>
<td>48</td>
</tr>
<tr>
<td>CHAPTER 5</td>
<td>56</td>
</tr>
<tr>
<td>DISCUSSION</td>
<td>56</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1: Numerical Representation of Day 1 Results for Volunteer 1 __________ 38
Table 2: Numerical Representation of Day 2 Results for Volunteer 1 __________ 39
Table 3: Numerical Representation of the Work Week Results for Volunteer 1 __ 40
Table 4: Numerical Representation of Day 1 Results for Volunteer 2 ___________ 43
Table 5: Numerical Representation of Day 2 Results for Volunteer 2 ___________ 44
Table 6: Numerical Representation of the Work Week Results for Volunteer 2 __ 45
Table 7: Numerical Representation of Day 1 Results for Volunteer 3 ___________ 49
Table 8: Numerical Representation of Day 2 Results for Volunteer 3 ___________ 50
Table 9: Numerical Representation of the Work Week Results for Volunteer 3 __ 51
LIST OF FIGURES

Figure 1: Overview of the Software Architecture ____________________________ 12
Figure 2: Flowchart describing the logger component _________________________ 14
Figure 3: Flowchart describing the analyzer component ______________________ 20
Figure 4: User Interface of the Latency Monitor Application __________________ 29
Figure 5: The “Per Day” view of the display _______________________________ 31
Figure 6: The “Per Week” view of the display ______________________________ 32
Figure 7: Day 1 “Per Day” View for Volunteer 1 ____________________________ 38
Figure 8: Day 2 “Per Day” View for Volunteer 1 ____________________________ 39
Figure 9: “Per Week” View for Volunteer 1 ________________________________ 40
Figure 10: Day 1 “Per Day” View for Volunteer 2 ___________________________ 43
Figure 11: Day 2 “Per Day” View for Volunteer 2 __________________________ 44
Figure 12: “Per Week” View for Volunteer 2 ______________________________ 45
Figure 13: Day 1 “Per Day” View for Volunteer 3 __________________________ 49
Figure 14: Day 2 “Per Day” View for Volunteer 3 __________________________ 50
Figure 15: “Per Week” View for Volunteer 3 ______________________________ 51
Figure 16: Average of all average latencies across all volunteers on Day 1 ____ 64
Figure 17: Average of all average latencies across all volunteers on Day 2 ____ 65
Figure 18: Volunteer1 Latency distribution for “o” during 1st & 2nd hours ____ 66
Figure 19: Volunteer1 Latency distribution for “o” during 3rd & 4th hours ____ 67
ABSTRACT

NON-INVASIVE KEYBOARD FATIGUE MONITORING SYSTEM FOR IMPROVING USER PERFORMANCE AND REDUCING INCIDENCES OF REPETITIVE STRAIN INJURIES

by

Satheesh Jayakumar

University of New Hampshire, December, 2007

Computers are ubiquitous in their application and deployment all over the world. Along with their universal appeal and versatility they also pose dangers to their various users in the form of ailments such as Repetitive Strain Injuries, Carpals Tunnel Syndrome, etc. which are all specifically related to keyboard use. The objective of this thesis was to explore the possibility of developing a deterministic and non-invasive method of detecting keyboard fatigue. A software application was developed which allowed us to reliably monitor this as a function of the latency between keyboard key-press and key-release events recorded by the resident operating system.

The latency trends that were observed through testing on three volunteers proved that the average latency calculated increased steadily with the onset of fatigue. Hence by estimating a threshold condition it was possible to train the system to estimate the fatigue level of the users and warn them appropriately at a considerably early stage of the condition.
CHAPTER 1

INTRODUCTION

Performing some form of data entry on computers is a procedure that needs to be carried out by a massive number of users all over the world. This data entry operation involves the use of certain muscles in the wrist and arms that need to be contracted in sequence to get the job done. Even though the instantaneous force required to perform such work is quite low, the highly repetitive nature of such work combined with poor posture, equipment positioning, etc. will lead to muscle fatigue and over a period of time, lead to other serious conditions such as Carpals Tunnel Syndrome, RSI, etc. [1].

A series of research projects [2, 3, 4] have been carried out by various graduate students in the Department of Electrical Engineering under the leadership of Prof. John R. LaCourse to address these issues. All of them were in general aimed at developing and evaluating non-obtrusive means of estimating the fatigue level of keyboard users. The first among them was the thesis titled “Keyboard monitor as a predictor for onset fatigue” by Chandra Ayyalasomayajula [2] with the objective of predicting fatigue by monitoring the error rate and typing speed of the user. The second thesis was “Non-invasive and seamless technology to monitor fatigue for long term typing on a laptop cursor plate” by
Shibli Subhani [3] that was also aimed at determining the fatigue level of the user. The metrics that were employed for achieving this objective were contact area, contact force, contact speed and contact time. The third thesis was titled “Trigraph latency as a method to infer fatigue during typing” by Baghirathi Nagaraju [3]. This project was also aimed at estimating the fatigue level of the user by recording the latencies associated with the trigraphs derived from approximately twenty eight most used words in the English language. Hence in order to carry out further research in this general area and add to the wealth of knowledge accrued, the objective that was envisaged for this thesis was to develop a system that monitored fatigue based on a metric called key holding time. The key holding time was defined as the latency between each key-press and key-release event recorded by the operating system, when the users typed on the keyboard of the computer work station. The expectation is that the research done in this area so far will drive further investigation and effort in the future to develop an all encompassing, comprehensive, efficient, non-obtrusive and accurate system to predict the onset of fatigue in computer users.

1.1 Repetitive Strain Injuries

Repetitive Strain Injuries (RSIs) occur from repeated physical movement causing damage to tendons, nerves, muscles and other soft body tissues. Apart from computer users, RSIs are also known to affect people involved in other occupations such as meatpackers, musicians, etc. [5]. It is an occupational overuse syndrome affecting muscles, tendons and nerves in the arms and upper back; hence it is also known as work related upper limb disorder or WRULD. The medically accepted reason it occurs is when muscles in these
areas are kept tense for very long periods of time, due to poor posture and/or repetitive motions [6].

Repetitive Strain Injuries cannot be considered a specific disease and can be, more accurately thought of, as a loose group of more specific conditions. Most of these disorders are related and hence it would not be uncommon for a person to be afflicted with many of them at the same time. In this case it is often best to treat RSI as a single general disorder, targeting all major areas of the arms and upper back in the course of treatment. Some of them are Carpal Tunnel Syndrome, DeQuervain’s Syndrome, Intersection Syndrome, Tenosynovitis, Tendonitis, etc. RSI conditions are best prevented in their early stages before they become too difficult to control.

Several studies have been carried out over a sufficiently large sample base to confirm if there is a positive association between extended computer work and musculoskeletal disorders of the upper extremities. A recent study in Denmark “Neck and Upper Extremity Disorders among Technical Assistants” that investigated the effect of the dose of computer use confirmed this association through testing, statistical analysis and profiling [7, 8, 9]. Using the data from this study Lassen et al. [8] reported a linear relationship between symptoms and computer use. It was also concluded that mouse and keyboard time were not predictive of clinical cases, possibly due to clinical case criteria which resulted in too few cases to perform multivariate analysis with computer duration time. In the same study Kruger et al. [7] reported the 7-day prevalence of moderate to severe forearm pain as 4.3% and the incidence of new forearm pain of 1.3%. Right
forearm pain was related to mouse-use with a linear increase with exposures from 0 to more than 30 hours/week and to keyboard use for more than 15 hours/week. Andersen et al. [9] reported the prevalence of possible Carpal Tunnel Syndrome as 1.4–4.8% and the incidence of new cases in 12 months as 5.5% based on symptoms, and 1.2% when confirmed by a clinical interview. The multivariate model showed elevated risk of possible Carpal Tunnel Syndrome with weekly use of a mouse for more than 20 hours/week. Keyboard use was non-significant, but the mean usage was only 8–9 hours/week.

In Jensen et al.'s two-year study of 3,475 computer users in Denmark [10], self-reported use of a computer more than 75% time, compared with 50% of time, increased the risk of hand/wrist. Using a mouse for 50% of time, compared with 25% of time, increased risk for females. Interestingly, intense computer work with little mouse usage also increased risk of hand/arm symptoms. Through this study Jensen et al found this increased risk of symptoms in a population of workers in a call center with no mouse-use [10]. In the 38-month prospective study of 789 newly hired office workers in Atlanta [11, 12], the physical condition of the workers were monitored by completing daily work diaries, symptom reporting and physical examination where the researchers looked for symptoms through clinical diagnostic techniques. A symptom case was defined as a severity of more than 5 (on a 10-point scale) or use of medication to relieve symptoms. The 12-month incidence rate for symptoms was 38.8 cases/100 person years and for diagnosed hand/arm disorders 21.1. Most common diagnoses were DeQuervains (14.7/100 person years), and medial and lateral epicondylitis 1.4 and 3.5. Carpal Tunnel Syndrome
incidence was 0.9/100 person years. The risk of hand/arm symptoms and disorders increased with increasing hours/day of keyboard activity, with a 2.2-fold increase at 20 hours/week. Hand/arm pain and disorders were also associated with at least two-years of computer use.

Bergqvist et al. [13, 14] studied the risk of VDT (Visual Display Terminal) usage on 535 office workers (91%) in 1981 and 341 of the remaining 353 same workers (97%) again in 1987 and compared computer to non-computer users in the group. They also compared computer users spending less than 6 hours with those spending up to 30 hours/week on the computer, and with intense users with more than 30 hours/week. Hand/wrist problems showed a dose response relationship with VDT use with a cumulative increase in incidence of 0.32 per hour increase in weekly VDT use. The relative risk comparing non-VDT users to VDT users for those introduced to VDTs during the time period was 4.04 and for the combined group of users with those introduced to computers during the time period was 2.84. Although there were elevated odds ratios for intense users, they were not statistically significant at p = 0.05, except for hand-wrist problems with intensive VDT use for at least six years. Drop-outs had a higher prevalence of hand/wrist problems than the VDT users suggesting self-selection was a factor that needed to be considered.

One of the primary recommendations to avoid RSIs is to pay special attention to pain and fatigue along with regular breaks to allow the muscles to recover from the strain imposed on them. The objective of this thesis is to evaluate a system which precisely aids in this process by monitoring fatigue and instructing the user to take micro-breaks.
1.2 Micro-breaks

It has been established that interspersing periods of continuous typing on the keyboard with frequent, brief rest periods, termed micro-breaks can reduce muscle fatigue, decrease the risk of injury and improve work performance [15]. Micro-breaks are muscle specific and hence can be targeted at specific muscle groups rather than a whole limb or even the whole body. Therefore during a micro-break all work activity need not stop, provided the work activities use muscles different to those being rested. For example, making a phone call can be considered a micro-break if the person had been involved in typing on the keyboard till then as the muscles groups employed for this task are different from those that need to be rested [1].

Thus it is justified to claim that developing and enforcing an optimum schedule for work and rest periods for keyboard users is fundamental to decreasing the risk of RSIs. It has been shown that the risks of musculoskeletal discomfort and injury during intensive computer work are significantly decreased when the users were allowed discretionary micro-breaks that totaled 30 seconds every 10 minutes [15]. In Galinsky et al. [16] the effects of a conventional schedule was compared with a supplementary schedule which contained an additional 5 minute break during each hour which otherwise did not contain a break. Results showed that breaks had a beneficial effect by reducing musculoskeletal discomfort without reduction in data-entry performance for those experiencing supplementary rest breaks.
1.3 Motivation and purpose

Several studies have been carried out to investigate the high prevalence of musculo-skeletal disorders (MSDs) among VDT (visual display terminal) work-station users [17, 19]. Studies on working environments [20], VDT peripherals [21] and working postures [22] were also carried out with objective of preventing or controlling MSDs. Although a large number of studies have been carried out in this general area it has been observed that obtaining empirical data over a large sample base has been difficult for research investigations into the cause and prevention of MSDs. This may be in part due to unavailability of a low cost, non-invasive and easily deployable means of monitoring a large study population. The common methods that were used to carry out research in this area have been through 3-D motion analysis systems, electromyogram (EMG), electronic goniometers, etc. which have been helpful in generating objective measurements of the physical conditions being experienced by the user [23]. Questionnaires were also popular and have been effectively employed to obtain valuable feedback from the users. However these techniques were inherently invasive and the quality of the data that was recorded may have been affected.

Therefore, the thrust of this thesis was aimed at using the powerful abilities and features of the PC work station itself to generate useful data. This data in turn could be used to obtain the metrics that are needed to analyze the physical condition of the user. The objective was to develop a non-invasive, transparent and light-weight software application running on the PC which could perform two major tasks. The first task was to interface with the resident operating system and log all the key-press and key-release
activities that are occurring on the keyboard along with time stamps with a resolution of 1 millisecond at the minimum. The second task was to analyze the log files in real-time and estimate the fatigue level experienced by the user based on the key holding time. The metric key holding time as explained previously is the time lag between the instances at which the keys are pressed to the points at which the keys are released. An example of how the software application could be trained based on empirical data obtained in the initial phase of experiments using a small sample set has been shown. In this way the system could eventually be capable of reliably indicating to the user when a micro-break is to be taken if a large sample set could be used to train the system. Hence using such a system any potential threat of a musculoskeletal disorder could be eliminated at the incipient stage itself.

1.4 Scope

There were four distinct milestones that needed to be achieved during the implementation phase of this thesis. The first milestone was to design, develop, test and validate the software component that was responsible for logging the time of occurrence of each key-press and key-release event that was occurring at the keyboard. The scope of this component was limited to simply generating log files of raw keystroke information. The second milestone was to develop and test the software component that could extrapolate the keystroke timing data from the log files in real-time and analyze it to obtain statistical information with respect to metrics such as key holding time. Once the integration of the “logger” and “analyzer” component was complete the resulting application called the “Latency Monitor” was tested using predetermined input to exercise all the significant
conditional and data paths of the software. After sufficient confidence was gained on the accuracy of the information generated by the “Latency Monitor” the third milestone was achieved wherein testing was carried out on three volunteers to collect adequate empirical data. During the experiment trials the fatigue level of the users was also monitored through self-reporting with the help of suitable questionnaires. The fourth milestone was to analyze the data collected with a statistical perspective and suggest a mechanism through which the software application could be trained with a sufficiently large sample set, to generate an alarm for the user whenever there was a need for a micro-break.

1.5 Thesis Organization

In the second chapter the overall design and software architecture of the application is presented. The third chapter describes the human aspect of the software. In the fourth chapter the results of the beta testing phase of experiments are statistically analyzed. In the fifth chapter the final results obtained from human trials is presented along with the advantages and limitations of the software. Suggestions for future work are also discussed.
CHAPTER 2

SOFTWARE ARCHITECTURE

The software application that was developed as part of this research effort could be thought of as comprising of two major components. One component was responsible for the hard-coupled interfacing with the operating system in order to obtain the most accurate timing information related to the keystroke activity that occurred at the keyboard. This component, henceforth referred to as the "logger" generated random access files containing all the raw keyboard parameters related to each keystroke event that occurred at the keyboard. The other software component referred to as the "analyzer" extracted the keystroke timing information from the log files and estimated the average, maximum and minimum latencies experienced by the user in the time period of interest. Based on the latency information threshold conditions could be established with a sufficiently large sample set which could then be used to advice the user to take a micro-break at appropriate instances.

2.1 Choice of Development Environment

A number of languages were considered at the beginning of this thesis effort to develop the required software application. After several trials and short scale feasibility studies using C#, Visual C++ and a combination of VB6.0 & VC++ it was determined that the
simplest, most elegant and the most efficient software solution could be developed using Visual Basic 6.0. Microsoft Visual Basic 6.0 is a rapid application development language in an environment that gives the users fast, easy, and intuitive tools to quickly develop Windows applications. Using Visual Basic, users can develop simple utilities or sophisticated applications with relative ease. Data access features allows them to create databases, front-end applications, and scalable server-side components for most popular database formats. ActiveX technologies allows them to use the functionality provided by other applications, and even automate applications and objects created using the Professional or Enterprise editions of Visual Basic [24].

Even though only a few of the rich feature set offered by VB 6.0 were used in this project, the flexibility and ease of use of this development environment was found to be invaluable. Some of the API's (Application Program Interface), UI (User Interface) tools, graphical tools and mathematical operators available in this environment made several design objectives very simple to implement and test. Hence all the software development in this thesis was successfully carried out using the language Visual Basic 6.0 in the Microsoft Visual Studio IDE (Integrated Development Environment) with relative ease and comfort.

2.2 Architecture Overview

The software application that was developed consisted of two major components as illustrated in Figure 1. The “back-end” which comprised of the

- Logger
• Analyzer

These two sub-components can be considered the engine of the application and addressed the primary requirements of the application. The other component that also required considerable design effort and testing was the front end which comprised of the

• UI (User Interface)
• Graphical Display Procedures

Figure 1: Overview of the Software Architecture

The front end served as the primary interface to the user of the application. It offered the user certain options to control the application and also displayed graphical results to the users.

2.3 Logger

The logger was the module responsible for capturing the keyboard event information over time and storing it reliably in a file using easily accessible data structures that lend itself to analysis.
2.3.1 Requirements

The major requirements of the logger component could be listed as follows:

**Transparent**: The logger should not interfere with the normal work activities of the user.

**Non-invasive**: The logger should not require any extra hardware peripherals that may interfere with the user’s work activities.

**Lightweight**: The logger was expected to be a simple utility that did not load the operating system too heavily in order to ensure the accuracy of the timing information that was obtained. Most operating systems installed on PC work stations are not real time in nature and hence are not capable of assuring the time-constrained execution of certain events. Hence if there were several tasks running concurrently or if there was a single “resource-greedy” task running then the priority awarded to our logger task might have caused undesirable effects to the timing information that was recorded by the operating system.

**Output**: The logger was expected to generate files containing raw keystroke information in which the time stamps related to the key-press and key-release events were of special interest to us at the analysis stage.
2.3.2 Program Flow

**Install Hook Routine**

- Start
- Declare the Required Windows APIs
- Install keyboard hook using API
- Stop

**Keyboard Callback Routine**

- Start
- Declare Required APIs, Data structures and file handlers
- Open a random access file
- If E, T, O, A, I
  - YES: Store keypress information in File
  - NO:
- Close the random access file
- Stop

Figure 2: Flowchart describing the logger component
Data Format: The logger component was responsible for storing the keystroke information in a format that aided the analysis of the raw data recorded at a later time. Hence, a particular data structure was adopted that encompassed all the required pieces of information as a single logical chunk that could be addressed and used as a unique entity. In other words, a user defined type was required which in turn could be used to access information such as the key manipulated, press or release event and time instance of the keyboard activity that was recorded.

2.3.3 Algorithm / Pseudo-code

The first step in the Logger task set was to define and declare low-level keyboard API methods which tied in closely with the Windows event management system. It is a known fact that a number of complex features that are available as part of Microsoft Visual Studio development environment requires some knowledge of Windows API’s. This is for the simple reason that VB6 (Visual Basic 6.0) runs on top of the Windows OS and there is no specific framework capturing this information. The Windows API’s that are used in conjunction with VB6 is procedural in nature. Hence in order to access and use their powerful features in Windows we have to load a library containing the procedures of interest and invoke those operations. Fortunately VB6 supports implicit library loading and API invocation by simply declaring the API methods we want to use. Hence to “Hook” the keyboard we had to declare and invoke the SetWindowsHookEx API. The SetWindowsHookEx function installs an application-defined hook procedure into a hook chain. A hook procedure would be installed to monitor the system for certain types of events which in our case were related to keyboard activity. These events are associated either with a specific thread or with all threads in the same desktop as the
calling thread depending on the parameters that are passed into the associated function call [25].

The declaration for SetWindowsHookEx is as follows

```vbnet
Public Declare Function SetWindowsHookEx Lib "user32" _
Alias "SetWindowsHookExA" (ByVal idHook As Long, _
    ByVal lpfn As Long, _
    ByVal hmod As Long, _
    ByVal dwThreadId As Long) As Long
```

The first argument is of the data-type "Long" [24] and represents an instruction to the API describing the kind of hook operation to perform. The second argument is actually a function pointer, called a callback. The third argument is the handle to the application instance, and the fourth argument is the application's thread ID. For our implementation purposes we used the following declaration to suit our requirements.

```vba
Private Const WH_KEYBOARD_LL = 13&
KeyboardHandle = SetWindowsHookEx(_
    WH_KEYBOARD_LL, AddressOf KeyboardCallback, _
    App.hInstance, 0&)
```

The constant WH_KEYBOARD_LL defined the kind of hook that was to be made: a low-level keyboard hook. The second argument was the AddressOf the KeyboardCallback function. The third argument was the application Windows handle,
and the value 0 was used for the thread ID. The value 0 indicated that the hook was associated with all threads on the desktop. By doing this the objective of hooking the keyboard and effectively trapping all keys effecting all applications was achieved. Thus far a mechanism was implemented through which the KeyboardCallback method was called every time a key was pressed, whether the application had the focus or not.

The second step in implementing the “Logger” component was to declare and invoke the KeyboardCallback procedure which was an application-defined callback function used with the SetWindowsHookEx function. The system called this function every time a new keyboard input event was about to be posted into a thread input queue. The keyboard input could come from the local keyboard driver or from calls to the keybd_event function. If the input came from a call to keybd_event, the input was "injected". However, the WH_KEYBOARD_LL hook is not injected into another process. Instead, the context switches back to the process that installed the hook and it is called in its original context. Then the context switches back to the application that generated the event. The syntax that was used is as follows:

```vbnet
Public Function KeyboardCallback(ByVal Code As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
```

As a rule in VB6, call back methods should be placed in modules, which in the case of this implementation was Keyboardhandler.bas. The signature of the callback that was used was a function that accepted three “Long” parameters and returned a “Long”. Hence
the callback can be thought of a generic windows message handler in which code has to be written to interpret and use the information sent to it by the operating system. Of the arguments that are sent to the callback, wParam represents the actual windows message constant and lParam plays the role of the pointer to the keyboard data. The argument “Code” is used to determine if the message posted to the thread queue is meant for the application defined callback.

After this, code was written within to do a quick and coarse filtration of the keyboard event data after which the information of interest was assigned to an appropriate data structure and recorded in a file. Two types of data structures were used in this implementation, KBDLLHOOKSTRUCT and FILEINFO. The former was used in conjunction with the function CopyMemory to store all the pertinent keyboard information provided by the pointer lParam. In other words CopyMemory was used to get the keyboard data from the address pointed to by lParam into a local static variable for coarse filtration process. At this stage only the keyboard events related to the characters of interest E, T, O, A and I were processed and all other events were discarded. After that the ASCII code associated with each keyboard event, the type of event (press/release) and the operating system time instance were assigned to the latter data structure FILEINFO and recorded in a random access file.

At this point, the operational flow of the “logger” component of the software is complete. This process continued in a loop as long as the associated un-hook procedure was not called. Thus the logger component continued to store all the information associated with
the keyboard activity in a large random access files till the user closed the application or explicitly used the “Stop Logging” option on the user interface.

2.4 Analyzer

The “analyzer” was the module responsible for reading back the relevant raw keyboard activity data from the files created by the logger component, analyzing the data on a statistical basis and storing the processed information in a form that lends itself to display in a graphical format.

2.4.1 Requirements

The major requirements of the analyzer component could be listed as follows:

Efficient: The analyzer component needed to be efficient in the use of MIPS (Millions of Instructions per seconds). This was because at certain instances the analyzer component could be invoked by the user while the logging component was till active. Hence to make sure that the timing information recorded remained accurate the implementation efficiency of the analyzer component had to be substantial.
2.4.2 Program Flow

Analyzer Routine

Start

Declare all the required Data structures, counters & Process Variables

Open the Random access file

Retrieve the next records to be processed

If Keypress Event

Identify the character and assign the index value

Update the current hour/day counters and latency variables

If "Hour of Day/Day of Week Switch" event In Next Record

Store all current hour/day information in the file

Reset all "current hour/day" variables and counters

If EOF

Stop

Figure 3: Flowchart describing the "analyzer" component
**File Operations:** The analyzer component needed to be capable of opening, maintaining and accessing both an input and an output file thread. The two subsequent data members of the current member of the file data structure pointed to needed to be kept track of all times. This was because the data was processed and segregated at one level based on the time and the switch-point from one collection of data to another needed to be known in advance by the algorithm. Due to the complications associated with having to open file handles within the same sub-routine the write operation needed to be carried out by another sub-routine which was passed the appropriate data structure that needed to be stored.

**Output:** The analyzer was expected to generate files containing information that represented the keyboard activity with a statistical perspective. The information needed to depict the minimum, maximum and average latency for each key used during, for example, every hour of day during the duration of the data collection.

**Data Format:** The analyzer component was responsible for storing the statistically significant information regarding the keyboard latency recorded in a format that aided the easy display of that information in a graphical format. Hence in this case too a particular data structure was adopted that would help in collecting and storing all the required and related information in a single logically related unit. Hence a user defined data type was used in this component to achieve this objective.

**2.4.3 Algorithm / Pseudo-code**

The first step in the analyzer task was to declare the various instances of the required data structures that were to be used for analysis. Four types of data structures were used in this component. The first one, called FILEINFO was the vehicle through which all the
relevant raw data regarding keystroke event information was stored and retrieved from the random access files. The second one called CHARINFO was used to store information regarding the current calculation being carried out. A user defined type was required for this since at any time instance information regarding each of the characters of interest needed to be stored and updated till a “hour of day/day of week switch” event occurred. The data of interest were total latency, total number of occurrences, and average, maximum and minimum latencies respectively. An “hour of time event switch” was when data related to the next hour of data collection was encountered and all the information related to the current hour stored using CHARINFO was dumped into the files and relevant counters were cleared for processing the next hour of keyboard activity. In the same vein, “day of week event switch” was when data related to the next day of data collection was encountered. Hence at that point all the information related to the current day was stored and the relevant counters were cleared.

The third type used was DAYSTOREINFO which was required to store the output obtained from the “analyzer” procedure. The graphical display required, stipulated that the data format allowed by DAYSTOREINFO needed to provide the average, maximum and minimum latency information for each hour of the data collection carried out when the “per day” view was chosen. In a similar fashion, the fourth type called the WEEKSTOREINFO was required to provide the average, maximum and minimum latency information over each day of data collection when the “per week” view was chosen.
The algorithm used to process the keyboard log information was quite straightforward. It consisted of the following sequence of steps:

Step 1: Open the file containing the raw information related to the key stroke activity recorded by the logger component.

Step 2: Retrieve the first record if this is the first pass through the data structure.

Step 3: Retrieve the next two records in order to monitor if “hour of day or day of week” is to occur when accessing the next record.

Step 4: In the first condition only key press events are considered and key release events are discarded.

Step 5: The character group associated with the particular keyboard event is identified.

Step 6: The current “hour of day or day of week” data is updated based on the character index identified in the previous step.

Step 7: Check the next record to see if the “hour of day or day of week” switch event is to take place. If so then all the relevant information such as total number of occurrences, total latency and average, maximum and minimum latencies associated with all the characters for the current “hour of day” or “day of week” is recorded in a file. Also, reset all the current “hour of day” or “day of week” parameters to get ready for the next calculation and analysis cycle.

Step 8: Repeat the whole process till all the records are processed.

At this point the operational flow of “analyzer” component was complete. This algorithm was repeated till all the records present in the raw “logger” generated file were analyzed and processed. The output of the analyzer component was a file that contained all the
information such as average, maximum and minimum latencies for the period of interest which was in turn used for display purposes.

The implementation of the "back-end" of this application was accomplished at this stage. All the design requirements were met to the best pragmatic extent possible to develop the back-end or the engine of the software application.
CHAPTER 3

HUMAN ASPECT

The human aspect of the software that was developed to aid in this research is discussed in this section. The human interface is significant in the sense that it is the primary means through which the logger is controlled and the statistical information is displayed with respect to the user.

3.1 Overview

From the user's perspective the software was represented by a friendly and intuitive GUI (Graphical User Interface). The UI consisted of two components, a control panel and a display window. The control panel contained buttons that were used to control the hooking and unhooking processes intrinsic to the software application. In effect the software application began logging all keyboard activity at the instance the user clicked on the “Start Logging” option on control panel. In the same way the logging process was terminated when the “Stop Logging” option was exercised. In the background the "logger" engine stored all the relevant information in large random access files. The moment the user clicked on the “Analyze & Display” button, the analyzer engine went to work and processed the raw data file to generate statistically significant latency
information for the keyboard activity during the period of interest. This information was displayed on an “easy to read” bar graph on the display panel.

3.2 Testing Logistics

In order to collect useful data during the experimental trials volunteers were required who could type continuously for four hours with a nominal degree of accuracy. For this, an email advertisement was sent to all the undergraduate and graduate students within the “Department of Electrical & Computer Engineering” at UNH. Three graduate students offered to help with this research effort and schedules were drawn and agreed upon for each testing session. All scientific research that requires testing using human volunteers needs to be approved by the Institutional Review Board (IRB). Hence an approval was sought for and obtained from IRB to carry out these human trials. During the course of these trials all the rules and regulations that were laid down by board were explicitly honored and adhered to.

3.3 Testing Procedure

All the testing sessions were preceded by a brief tutorial to the volunteers on the objective, breadth and scope of the research effort. They were also given a brief overview of how the software worked and instructions on how to use it effectively. In effect the following set of points was discussed with each of the volunteers:

- The objective is to develop a reliable and non-invasive method of detecting and monitoring fatigue experienced by people when using keyboards on PC workstations.
• A software application has been developed which would allow us to monitor keyboard fatigue as a function of the latency between keyboard key-press and key-release events.

• The application to be employed is highly efficient, non-intrusive and will not affect any of the user's applications.

• The application is to be launched and allowed to run as a background task.

• The volunteers are expected to carry out their typing assignments just as any typical data entry operator would function.

• The application is designed to filter and process information only related to the five most used characters in the English alphabet [26, 27, 28, 29, 30, 31] and hence will not record any private information such as user-names, passwords, etc.

• The trials do not pose any danger to the volunteers as there are no electrical, pneumatic or electro-mechanical devices that need to be used.

• Volunteers could cease typing at any point if they feel any discomfort, strain or pain during testing. The only request that was made was that they record their observations clearly in the questionnaires that were handed out to them.

This was followed by seating the volunteers at their designated work-stations, ensuring that they were comfortable and then launching the application on their respective systems. All of them were provided with the same material to type and were asked to continue typing in a loop using that material till the stipulated end of the experiment period was reached. At suitable intervals the volunteers were asked to use the "hand-dynamometer" and fill out a questionnaire describing their observations. The "hand-
dynamometer”, was a device used in the experiments to induce fatigue more quickly in the volunteers. This allowed collection of more pertinent data over the shorter periods of time over which the experiments were conducted.

3.4 Graphical User Interface

The GUI that was used as a part of the software application provided control and display functionality to the users. The interface was designed to be simple and intuitive from the user’s perspective. When the application was launched the user or test volunteer was shown the window illustrated in Figure 4.

The users were trained to use the simple buttons provided in the control panel to control the operation of the application. The user was typically expected to begin by clicking on the “Start Logging” button. Internally the associated “on-click” routine would initiate the logging procedure for capturing all the relevant key stroke information from that point in time onwards. The users were expected to minimize the application and continue with their normal work or typing assignments. The “latency monitor” application would continue to run in the background and record all the keyboard event information in large random access files. Once the test session came to an end or when the user was done with his or her work on the specific work-station, they were expected to restore the “latency monitor” window. The “Stop Logging” command button was to be clicked to stop the application from collecting any further information. Finally the user could optionally choose to analyze all the keyboard event data to extrapolate all the statistically significant information. This process could be initiated by clicking on the “Analyze” button. In other
words, every time this button was used, an analyze cycle was carried out. After this the "Refresh Display" button needed to be clicked after choosing the particular "view" of choice to ensure that the information graphically represented on the front-end reflected the results of the latest analysis.

Figure 4: User Interface of the Latency Monitor Application
The display panel was the primary means by which all the information that was recorded, analyzed and interpreted could be shown to the outside world. Even though, the figures themselves were generated from the data stored in some results files; those files were essentially of VB 6.0 random access format which could not be read using other utilities such as “notepad”, “MS Word”, etc. Hence the application was completely dependent, in not so uncertain terms on the display panel to effectively represent the statistics in a succinct, clear and useful format to both the researcher and the volunteers.

The fundamental component of the display panel was the bar graph which was centered on the “latency monitor” window. This graph compared the varying magnitudes of the keyboard latencies recorded for each of the character of interest against the time period of interest. Internally the analyzer would have extracted and analyzed the data recorded to estimate the average, maximum and minimum latencies associated with each of the five most used characters in the English alphabet. This information was then stored in such a way that they could be retrieved to display the different latencies over a period of a day or over a period of a week graphically. Thus the “view” or the duration of the test whose results need to be observed, whether over a week or over a day, was controlled by a list box in the upper right hand corner of the application window. In addition to that, the user under the “Per Day” view could also optionally choose the day of interest using the second list box. The ability to evaluate the measurements over longer periods of time permitted more confidence in the consistency expected from these empirical results.
A typical example of the appearance of the “latency monitor” when a “analyze and display” cycle has been exercised is shown in Figure 5. Here each of the colored bars represents the average latency measured for a specific character over the specified one hour duration represented. The red, green, magenta, blue and yellow colors represent the average latency of ‘e’, ‘t’, ‘o’, ‘a’ and ‘i’ respectively which are the five most used
characters in the English alphabet. In effect, the variation of the average latency experienced while typing each of these characters over every hour of the duration of the test is clear from this graphical display.

Figure 6: The “Per Week” view of the display for a three day period

Figure 6 represents the appearance of the monitor when the “Per Week” view is chosen using the combo box in the upper right hand corner. The results of any set of
Experimental trials are constrained by a limited sample set of data over durations which are far less than the real life situations that they try to mimic or simulate. However a rough approximation of the results that can be expected in real life situations could be estimated by averaging. This should then be followed by analysis and comparison of the statistical nuances of these results over increasing durations of the tests. In this context, the interpretations that can be made using the "Per Week" view is significant. It allows one to observe and predict the general distribution of the latencies that can be expected in real life situations.

Another point to be considered when discussing the human aspect of these research experiments is the importance awarded to the information from the questionnaires handed out to test volunteers. This valuable feedback gives us an additional point of reference to correlate with the conclusions drawn from the bar graphs. In addition to answering the specific questions that had been asked in the questionnaires, the volunteers were encouraged to record in detail any discomfort, pain or strain they felt during the typing process. They were also requested to record the time and specific points in their limbs or body where they felt these conditions.

Thus from the different views of the bar graphs and the information collated and processed from the questionnaires a scientific threshold was established. This was done as an example exercise with the small sample set of data available. This was then used to estimate and establish an alarm condition at which point the continued keyboard activity could be harmful to the well being of the work-station user. In conclusion, the initial data
that was gathered from the experimental trials was used to train the system to recognize the symptoms of keyboard fatigue. This was done using latency as a metric and the future users needed to be advised to take relaxing breaks at appropriate points in time to avoid and minimize the risk of developing RSIs, musculo-skeletal disorders, carpals tunnel syndrome, etc. during the course of their professional life.
CHAPTER 4

RESULTS

This chapter focuses on the analysis of the data and the results that were obtained from the experiments carried out using the "Latency Monitor" application. The testing sessions were conducted over two days for each of the volunteers over four hour periods. In terms of raw data collected this was equivalent to three random access files containing approximately twenty thousand records each, related to the keypress/keyrelease events that took place over the eight hours of testing for each of the volunteers. These raw data files were processed using the "Analyzer" component of the "Latency Monitor" application. The final observations and trends were inferred from graphical views that were presented on the display panel of the application.

Each of the testing sessions was preceded by a short tutorial to the volunteer educating him or her about the objective of the study and how the "Latency Monitor" functioned. Any concerns or questions that the volunteers had regarding their privacy and potential risks involved in the testing were addressed. Once the volunteers were mentally and physically ready to begin testing the software was installed on the work station of their choice. A standard document was used for all the testing sessions and all the volunteers were requested to observe the following directions during the testing.
• Type the material provided in the document continuously for as long as possible up to a maximum of four hours.

• Perform 15 compressions of the hand dynamometer using each hand every thirty minutes to induce fatigue.

• Co-operate with the researcher to complete the “fatigue” questionnaire every 30 minutes during the test session.

• Mention any additional information such as breaks taken, the specific need for the break, any other symptom of fatigue, etc. that were in turn recorded as notes on the questionnaire.

The three sets of results that were obtained are critically analyzed in the following sections.

4.1 Volunteer 1 Observations

The first volunteer was a healthy male graduate student who had moderate typing abilities. The software was installed on his laptop and the logging process was initiated. All the standard test procedures mentioned were carried out over the two testing sessions that he participated in. The results obtained for Volunteer 1 are represented by the latency graphs in Figure 7, 8 & 9 and numerical latency values (in ms) provided in Tables 1, 2 & 3 respectively.

From the graph in figure 7 which depicts the average, minimum and maximum latencies experienced by the volunteer on the first day of testing it is clear that in general there was a gradual increase in the average latency as fatigue set in. Going from the first to the
second hour of testing the average latency increased by 2 ms for “e”, 1 ms for “t”, 5 ms for “o” and 3 ms for “i” respectively. In a similar fashion, from the second to the third hour the average latency increased by 5 ms for “o”, 3 ms for “a” and 3 ms for “i”. However from the third to the fourth hour the average latencies decreased by 4 ms for “e”, 2 ms for “t”, 1 ms for “o” and 1 ms for “i” respectively. The drop in the latency during the last hour could be as a consequence of the break that Volunteer 1 had availed during the third hour. Hence these results do support the notion that the average latencies appreciated for the majority of the character set during the first three hours of testing. It is interesting to note that the average latency recorded for “e” and “t” showed a depreciating trend from the second hour onwards. It could be argued that Volunteer 1 could have gained experience and familiarity with the location of the most frequently used keys on the keyboard during the initial two hours of typing. Hence from that point onwards he would have been able to type those keys with increased efficiency and consequently reduced latency.

From the questionnaire (Appendix C) in which Volunteer 1 reported the symptoms of fatigue during the first day of testing it is evident that he began experiencing the first signs of fatigue in the wrist about 1.5 hours into the testing session. From the graph the maximum latencies for “o” and “i” reached the highest levels of 340 ms and 401 ms respectively during the second hour. Hence this would lead us to believe that Volunteer 1 experienced the increased latency in his keystrokes during the second hour specifically due to the onset of fatigue as reported in questionnaire.
Figure 7: Day 1 “Per Day” View for Volunteer 1

Expt Date: 8/9/2007
Expt Time: 1:00 PM to 4:00 PM

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>1st Hour</th>
<th>2nd Hour</th>
<th>3rd Hour</th>
<th>4th Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>1</td>
<td>“e”</td>
<td>107</td>
<td>271</td>
<td>10</td>
<td>109</td>
</tr>
<tr>
<td>2</td>
<td>“t”</td>
<td>98</td>
<td>220</td>
<td>20</td>
<td>99</td>
</tr>
<tr>
<td>3</td>
<td>“o”</td>
<td>116</td>
<td>180</td>
<td>30</td>
<td>121</td>
</tr>
<tr>
<td>4</td>
<td>“a”</td>
<td>108</td>
<td>230</td>
<td>10</td>
<td>106</td>
</tr>
<tr>
<td>5</td>
<td>“i”</td>
<td>118</td>
<td>301</td>
<td>10</td>
<td>121</td>
</tr>
</tbody>
</table>

Table 1: Numerical Representation of Day 1 Results for Volunteer 1
Figure 8: Day 2 “Per Day” View for Volunteer 1

Expt Date: 8/10/2007
Expt Time: 1:00 PM to 4:00 PM

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>1st Hour</th>
<th>2nd Hour</th>
<th>3rd Hour</th>
<th>4th Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
</tr>
<tr>
<td>1</td>
<td>“e”</td>
<td>111</td>
<td>301</td>
<td>20</td>
<td>108</td>
</tr>
<tr>
<td>2</td>
<td>“f”</td>
<td>100</td>
<td>210</td>
<td>20</td>
<td>98</td>
</tr>
<tr>
<td>3</td>
<td>“o”</td>
<td>116</td>
<td>240</td>
<td>50</td>
<td>122</td>
</tr>
<tr>
<td>4</td>
<td>“a”</td>
<td>112</td>
<td>310</td>
<td>10</td>
<td>106</td>
</tr>
<tr>
<td>5</td>
<td>“i”</td>
<td>120</td>
<td>281</td>
<td>40</td>
<td>121</td>
</tr>
</tbody>
</table>

Table 2: Numerical Representation of Day 2 Results for Volunteer 1
Figure 9: “Per Week” View for Volunteer 1

Expt Time: 1:00 PM to 4:00 PM on both days

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>First Day</th>
<th>Second Day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>1</td>
<td>“e”</td>
<td>121</td>
<td>401</td>
</tr>
<tr>
<td>2</td>
<td>“i”</td>
<td>107</td>
<td>401</td>
</tr>
<tr>
<td>3</td>
<td>“o”</td>
<td>97</td>
<td>231</td>
</tr>
<tr>
<td>4</td>
<td>“a”</td>
<td>122</td>
<td>340</td>
</tr>
<tr>
<td>5</td>
<td>“i”</td>
<td>107</td>
<td>251</td>
</tr>
</tbody>
</table>

Table 3: Numerical Representation of the Work Week Results for Volunteer 1
On the second day of testing the same procedure was followed and the application was restarted. The average latencies calculated for the second day showed a steady increase in magnitude proportional to the period of time spent in typing. Going from the second to the third hour of testing the average latency increased by 1 ms for “e”, 3 ms for “t”, 10 ms for “o”, 1 ms for “a” and 7 ms for “i” respectively. In a similar fashion, from the third to the fourth hour the average latency increased by 4 ms for “e”, 1 ms for “t”, 6 ms for “o”, 2 ms for “a” and 2 ms for “i”. However from the first to the second hour the average latencies decreased by 3 ms for “e”, 2 ms for “t” and 6 ms for “a” respectively. Even though the average latencies for all the character sets displayed an appreciating trend in general there were subtle differences in the trends observed for the character groups “o”, “i” and “e”, “t”, “a” respectively. The drop in the latency during the second hour of testing may be attributed to the break that Volunteer 1 took during the very first hour of testing. Since Volunteer 1 required a break in the very first hour one could argue that typing for less than an hour by itself could not be responsible for such a high level of fatigue experienced. In this case, a more plausible explanation would be that volunteer 1 was suffering from the cumulative effect of the testing combined with the previous activity in which the volunteer was involved in before the session commenced. The increased levels of fatigue experienced by the user during the very first hour was further supported by the high levels of instantaneous latencies recorded during that period for “e”, “t” and “i” which were 390 ms, 280 ms and 290 ms respectively.

From the questionnaire (Appendix C) it is also clear that Volunteer 1 had experienced high levels of fatigue during the last hour of testing on the second day. The maximum
latencies recorded during the period in question were 460 ms, 330 ms and 341 ms for “e”, “t” and “o” respectively. Hence once again a physiological symptom of fatigue felt by the volunteer co-related well with maximum instantaneous latencies recorded by the application.

Figure 9 represents the average latencies measured over the two days of testing carried out for Volunteer 1. This graph showed that the average latencies recorded in the second day of testing were marginally higher than the latencies recorded on the first day. Hence one could argue that the effect of fatigue was cumulative in the case of Volunteer 1 which caused him to experience increased level of latency during the second day of testing.

4.2 Volunteer 2 Observations

The second volunteer was a healthy female candidate who had good typing skills. Just as in the case of the first volunteer the software was installed on the second volunteers workstation and the application was launched. All attempts were made to help the volunteer set up as ergonomically and comfortably as possible. The results obtained for Volunteer 2 are represented by the latency graphs in Figure 10, 11 & 12 and numerical latency values (in ms) provided in Tables 4, 5 & 6 respectively.
Figure 10: Day 1 “Per Day” View for Volunteer 2
Expt Date: 8/1/2007
Expt Time: 1:00 PM to 4:00 PM

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>1st Hour</th>
<th>2nd Hour</th>
<th>3rd Hour</th>
<th>4th Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>1.</td>
<td>“e”</td>
<td>122</td>
<td>390</td>
<td>40</td>
<td>116</td>
</tr>
<tr>
<td>2.</td>
<td>“t”</td>
<td>111</td>
<td>280</td>
<td>40</td>
<td>116</td>
</tr>
<tr>
<td>3.</td>
<td>“o”</td>
<td>132</td>
<td>240</td>
<td>50</td>
<td>127</td>
</tr>
<tr>
<td>4.</td>
<td>“a”</td>
<td>116</td>
<td>260</td>
<td>50</td>
<td>113</td>
</tr>
<tr>
<td>5.</td>
<td>“i”</td>
<td>127</td>
<td>290</td>
<td>60</td>
<td>122</td>
</tr>
</tbody>
</table>

Table 4: Numerical Representation of Day 1 Results for Volunteer 2
Figure 11: Day 2 “Per Day” View for Volunteer 2

Expt Date: 8/2/2007
Expt Time: 8:00 AM to 12:00 PM

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>1st Hour</th>
<th>2nd Hour</th>
<th>3rd Hour</th>
<th>4th Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
</tr>
<tr>
<td>1</td>
<td>“e”</td>
<td>100</td>
<td>300</td>
<td>20</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>“t”</td>
<td>88</td>
<td>300</td>
<td>10</td>
<td>89</td>
</tr>
<tr>
<td>3</td>
<td>“o”</td>
<td>110</td>
<td>300</td>
<td>40</td>
<td>118</td>
</tr>
<tr>
<td>4</td>
<td>“a”</td>
<td>105</td>
<td>301</td>
<td>10</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>“i”</td>
<td>110</td>
<td>301</td>
<td>20</td>
<td>116</td>
</tr>
</tbody>
</table>

Table 5: Numerical Representation of Day 2 Results for Volunteer 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Expt Time: 1:00 PM to 4:00 PM on Day1 & 8:00 AM to 12:00 PM on Day2

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>First Day</th>
<th>Second Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“e”</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>2</td>
<td>“t”</td>
<td>120</td>
<td>390</td>
</tr>
<tr>
<td>3</td>
<td>“o”</td>
<td>116</td>
<td>320</td>
</tr>
<tr>
<td>4</td>
<td>“a”</td>
<td>133</td>
<td>310</td>
</tr>
<tr>
<td>5</td>
<td>“i”</td>
<td>116</td>
<td>300</td>
</tr>
<tr>
<td>1</td>
<td>“e”</td>
<td>127</td>
<td>601</td>
</tr>
</tbody>
</table>

Table 6: Numerical Representation of the Work Week Results for Volunteer 2
Figure 10 which depicts the latency distribution for Volunteer 2 on day 1 shows that the average latencies in general had a gradual increasing trend from the second hour onwards. Going from the second to the third hour of testing the average latency increased by 8 ms for “e”, 2 ms for “t”, 10 ms for “o”, 2 ms for “a” and 8 ms for “i” respectively. From the third to the fourth hour though the average latency increased by 5 ms only for “a” the values for “t”, “o” and “i” remained relatively steady at the levels that were recorded during the third hour. However from the first to the second hour the average latencies decreased by 6 ms for “e”, 5 ms for “o”, 3 ms for “a” and 5 ms for “i” respectively. In the case of volunteer 2 a break was taken during the very first hour of the first testing session that she was involved in. She had experienced very high levels of latencies such as 390 ms for “e”, 280 ms for “t” and 290 ms for “i” respectively during the first hour of testing.

Hence it was obvious that volunteer 2 was affected by the cumulative fatigue from her activity prior to the testing session and that combined with the typing in the first hour had ultimately forced her to take a break. High levels of instantaneous latencies such as 301 ms for “e”, 300 ms for “t”, 310 ms for “o”, 300 ms for “a” and 601 ms for “i” respectively were also recorded during the last hour of that session. This correlated well with the physiological symptoms of fatigue that the volunteer had reported in the questionnaire (Appendix D) during the same period.

The results obtained from the second day of testing for Volunteer 2 is represented by Figure 11 from which it is clear that in general there was a gradual increase in the
average latency as fatigue set in. Going from the first to the second hour of testing the average latency increased by 1 ms for “e”, 1 ms for “t”, 8 ms for “o” and 6 ms for “i” respectively. In a similar fashion, from the second to the third hour the average latency had increased significantly across the board by 13 ms for “e”, 33 ms for “t”, 29 ms for “o”, 41 ms for “a” and 15 ms for “i”. However from the third to the fourth hour, average latencies decreased by 3 ms for “e”, 11 ms for “t”, 6 ms for “o”, 17 ms for “a” and 2 ms for “i” respectively. On the second day volunteer 2 did not take a break until the end of the third hour of testing even though she exhibited signs of fatigue through very high instantaneous latencies during the first three hours of testing itself. Hence the drop in average latencies during the last hour may be considered a direct consequence of the break taken from typing during the end of the third hour. This particular set of observations was special in the sense that the user appeared to have carried over some fatigue from typing on the previous day. This assertion was further supported by the fact that in the case of volunteer 2 the testing session on the second day was scheduled from 8:00 AM to 12:00 PM while the session on the first day was from 1:00 PM to 4:00 PM. In addition to that latency instances over 300 ms which were recorded during the first two hours of typing proved that volunteer 2 began experiencing fatigue from the very first hour of testing on the second day. There was also a relatively sharp increase in the average latency values recorded in the third hour of the session. The volunteer had also reported considerable pain in her right hand at the end of the third hour. This implied that the relatively large instantaneous latencies such as 421 ms for “e”, 300 ms for “t”, 301 ms for “o” and 650 ms for “i” respectively reported during the third hour of testing were undoubtedly due to the onset of fatigue in the hand, wrist and forearms of the user.
The "Per Week" view of the latencies over the two sessions of testing for Volunteer 2 is shown in Figure 12. For Volunteer 2 unlike Volunteer 1 the latency trend over the two days of testing was marginally depreciating. This could be related to the typing style adopted by the user on each of the sessions. On the first day the user displayed a steady and almost stable latency distribution. However on the second day of testing the average latency was comparatively less for the first two hours.

4.3 Volunteer 3 Observations

The third candidate was a young graduate student with moderate typing abilities. The volunteer requested that the software be installed on his lab workstation on which the testing sessions were carried out. All the procedures that were followed for the previous two volunteers were repeated for the third volunteer during the two testing sessions. The results obtained for volunteer 3 are represented by the latency graphs in Figures 13, 14 & 15 and numerical latency values (in ms) provided in Tables 7, 8 & 9 respectively.

The latency distribution exhibited by volunteer 3 on the first day of testing is represented by the "Per Day" view latency graph in Figure 13. In general the average latencies showed a gradual and steady increasing trend just as in the case of each of the previous volunteers.
Expt Date: 8/12/2007
Expt Time: 8:00 AM to 12:00 PM

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>1st Hour</th>
<th>2nd Hour</th>
<th>3rd Hour</th>
<th>4th Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
</tr>
<tr>
<td>1.</td>
<td>“e”</td>
<td>105</td>
<td>221</td>
<td>10</td>
<td>106</td>
</tr>
<tr>
<td>2.</td>
<td>“t”</td>
<td>93</td>
<td>260</td>
<td>10</td>
<td>95</td>
</tr>
<tr>
<td>3.</td>
<td>“o”</td>
<td>114</td>
<td>191</td>
<td>30</td>
<td>115</td>
</tr>
<tr>
<td>4.</td>
<td>“a”</td>
<td>107</td>
<td>211</td>
<td>20</td>
<td>107</td>
</tr>
<tr>
<td>5.</td>
<td>“i”</td>
<td>112</td>
<td>270</td>
<td>30</td>
<td>110</td>
</tr>
</tbody>
</table>

Table 7: Numerical Representation of Day 1 Results for Volunteer 3
Figure 14: Day 2 “Per Day” View for Volunteer 3

Expt Date: 8/13/2007  
Expt Time: 1:00 PM to 4:00 PM

<table>
<thead>
<tr>
<th>#</th>
<th>Char</th>
<th>1\textsuperscript{st} Hour</th>
<th>2\textsuperscript{nd} Hour</th>
<th>3\textsuperscript{rd} Hour</th>
<th>4\textsuperscript{th} Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“e”</td>
<td>109 551 10</td>
<td>101 351 10</td>
<td>104 211 10</td>
<td>113 291 30</td>
</tr>
<tr>
<td>2</td>
<td>“t”</td>
<td>97 601 10</td>
<td>91 241 30</td>
<td>95 191 20</td>
<td>103 260 20</td>
</tr>
<tr>
<td>3</td>
<td>“o”</td>
<td>118 301 10</td>
<td>118 251 50</td>
<td>124 200 40</td>
<td>137 381 20</td>
</tr>
<tr>
<td>4</td>
<td>“a”</td>
<td>107 301 10</td>
<td>103 220 40</td>
<td>104 211 40</td>
<td>119 311 40</td>
</tr>
<tr>
<td>5</td>
<td>“i”</td>
<td>123 420 20</td>
<td>122 250 10</td>
<td>121 230 50</td>
<td>133 310 310</td>
</tr>
</tbody>
</table>

Table 8: Numerical Representation of Day 2 Results for Volunteer 3
Figure 15: “Per Week” View for Volunteer 3

Expt Time: 8:00 AM to 12:00 PM on Day1 & 1:00 PM to 4:00 PM on Day2

<table>
<thead>
<tr>
<th></th>
<th>Char</th>
<th>First Day</th>
<th>Second Day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>1.</td>
<td>“e”</td>
<td>105</td>
<td>361</td>
</tr>
<tr>
<td>2.</td>
<td>“t”</td>
<td>95</td>
<td>310</td>
</tr>
<tr>
<td>3.</td>
<td>“o”</td>
<td>118</td>
<td>310</td>
</tr>
<tr>
<td>4.</td>
<td>“a”</td>
<td>106</td>
<td>301</td>
</tr>
<tr>
<td>5.</td>
<td>“i”</td>
<td>114</td>
<td>301</td>
</tr>
</tbody>
</table>

Table 9: Numerical Representation of the Work Week Results for Volunteer 3
Going from the first to the second hour of testing the average latency increased by 1 ms for “e”, 2 ms for “t” and 1 ms for “o” respectively. In a similar fashion, from the second to the third hour the average latency increased by 10 ms for “o”, 8 ms for “i” and remained at a constant level of 95 ms for “t”. From the third to the fourth hour, average latencies increased by 2 ms for “e”, 1 ms for “t” and 1 ms for “a” respectively. Though the differences in the average latencies recorded for each of the character groups changed by a comparatively smaller margin from hour to hour, the general trend suggested a proportional relationship between latency and the fatigue experienced.

One interesting aspect of the first days’ results for volunteer 3 was that the maximum latencies were recorded at and around a consistent high level of 300 ms. In addition to that, volunteer 3 also recorded comparatively high instantaneous latency levels such as 361 ms for “e”, 300 ms for “t”, 301 ms for “o”, 301 ms for “a” and 301 ms for “i” respectively during the second hour of testing itself. This observation coupled with the information from the questionnaire supports the notion that the latencies associated with individual keystrokes reach maximum levels with the onset of fatigue.

The results for the second day of testing for volunteer 3 are displayed though Figure 14 and Table 8 which show a slightly different latency distribution from day 1. In the case of the second day the average latencies decreased initially in the second hour, then increased gradually till the third hour and spiked to the highest levels in the last hour. To be specific, going from the second to the third hour of testing the average latency increased by 3 ms for “e”, 4 ms for “t”, 6 ms for “o” and 1 ms for “a” respectively. From
the third to the fourth hour the average latency increased significantly for the entire character set which was by 9 ms for “a”, 8 ms for “t”, 13 ms for “o”, 15 ms for “a” and 12 ms for “i” respectively. However from the first to the second hour the average latencies decreased by 8 ms for “e”, 6 ms for “o”, 4 ms for “a” and 1 ms for “i” respectively. From the notes in questionnaire filled out for Volunteer 3 it is clear that he did request and take a break during the first hour of the second testing session. He had complained of significant pain and strain in his knuckles, fingers and wrist three-fourth into the first hour. During the same period he had also experienced very high levels of instantaneous latencies such as 551 ms for “e”, 601 ms for “t”, 301 ms for “o”, 301 ms for “a” and 420 ms for “i” respectively which were obviously caused by the high levels of fatigue that he was afflicted with. Hence it could be argued that volunteer 3 was affected by the cumulative fatigue from his activity from just before the testing session and that combined with the typing in the first hour caused an extreme level of fatigue to set in. High levels of instantaneous latencies such as 291 ms for “e”, 381 ms for “o”, 311 ms for “a” and 310 ms for “i” respectively was also recorded during the last hour of that session. This correlated well with the physiological symptoms of fatigue that the volunteer had reported in the questionnaire (Appendix E) during the same period.

The unexpected pain that volunteer 3 experienced at the end of the very first hour of testing could be attributed to two causes. One is the cumulative fatigue that could have been contracted during the previous days’ testing. The second likely explanation is the exertion that might have been caused by the user performing some labor intensive work using his hands shortly before the second session of testing. However following the
discussion so far it was seen that even a fifteen minute break had a considerable impact on the fatigue level experienced by the user and the latencies observed thereafter. Hence in the case of volunteer 3's second day results it would be a more sound argument to suggest that he might have been involved in some activity prior to the testing session which involved the strenuous use of his arms, wrists and hands. This in turn could have caused him to exhibit the instances of uncharacteristically high levels of fatigue observed during the first hour of testing itself.

The "Per Week View" for volunteer 3 was shown on Figure 15 which represented the latencies experienced by the user over the two days of testing calculated on a "per-day" basis. The average latencies recorded seem to have increased from the first session to the next, more so for the characters "o" and "i" than for the others. The user had also experienced heightened physiological symptoms of fatigue, twice during the second session of testing. Though the pain experienced during the first hour of testing could be reasonably attributed to activity prior to the testing session, in general it looked like volunteer 3 did seem to experience increased average levels of latency on the second day of testing. Hence one can assume that in the case of volunteer 3 the fatigue experienced was cumulative over the two sessions and the user could be expected to experience higher fatigue levels if he continues to carry out the same typing activity on subsequent days.

Based on the testing sessions that were conducted in general it was observed that there was an unmistakable positive association between time spent at typing and the fatigue experienced. This result was true for all the volunteers irrespective of the testing sessions.
in which they had participated. Some of the common trends that were observed for all the volunteers are as follows:

- Most of the testing sessions were marked by physiological symptoms of fatigue experienced on or after the second or third hour of testing.

- In general the average latencies increased in a manner proportional to the amount of time spent typing continuously at the testing station.

- In the questionnaires the volunteers reported palpable fatigue symptoms only when the pain exceeded a certain psychological threshold. In other words the volunteers complained of fatigue only when they experienced a very acute level of pain at which point they simply had to stop and could not continue typing.

- Even short breaks were seen to have a profound effect on the fatigue level experienced thereafter for all volunteers. In other words the brief rest provided, rejuvenated the related muscles in the arms, wrist, etc. allowing the users to continue typing with lower levels of latency from that point onwards.

- The volunteers were observed to slow down involuntarily during the testing sessions in response to fatigue. Hence even though they continued typing they were, in a way, actually resting since the reduced typing speed resulted in a lower level of effort required from the user during that period.
CHAPTER 5

DISCUSSION

When an individual is involved in the typing process the sequence of steps involved in that activity can be broken down as follows:

- Reading the subject material that is to be typed
- Retaining that small amount of information accurately during the typing
- Locating and depressing the appropriate keys
- Verifying the accuracy of the typed information
- Repeat the process for the subsequent material

This is the process that would be typically expected to be followed by most computer users all over the world. However, professional typists are capable of reliable data entry through a more coordinated and efficient process by carrying some of these steps in parallel. In either case, it can be seen that several parts of the human body are involved in the typing process and contribute in varying degrees to the general fatigue experienced by the person. Of these, the eyes and the hands can be considered the direct interface between the human operator and the machine. Hence, the effects of fatigue can be expected to be most pronounced in these body parts.

The focus of this thesis has been to identify and validate a metric that would allow us
quantify in a reliable way the amount of fatigue experienced by computer users in their arms, wrists and hands. Towards this end, keyboard latency, defined as the elapsed time between the key-press and key-release operations was adopted as the metric of choice for this project. It was hypothesized that as fatigue set in the effective time taken to complete a succession of key-press and key-release operations would increase gradually. This would be caused due to the inherent sluggishness that would affect any muscle of the human body when required to carry out a repetitive activity without adequate rest in-between.

5.1 Post Development Analysis

The first phase of the project involved the identification of a suitable mechanism through which all the keyboard events such as key-presses and key-releases could be recorded reliably. Towards this end several development platforms were evaluated in terms of the features and the hooks they provided into the Windows operating system. Once a framework was identified which allowed reliable logging of keyboard messages the primary software component called the “Logger” was successfully implemented. The “Logger” was subject to a certain amount of unit testing where it was determined that all the keyboard events were being logged with a high level of accuracy as long as the system was lightly loaded.

The second phase was aimed at developing the set of algorithms that were required to extract and statistically analyze all the logged keyboard event information. This component referred to as the “Analyzer” was in essence responsible for processing the
sizeable amount of raw data and provide them in a form that could be interpreted visually through the use of bar charts. Two variants of the algorithm were developed, one of which was responsible for estimating and recording the processed information on a “per day” basis while the other one was for doing the same on a “per week” basis. Both the algorithms were verified and tested using known data sets in addition to using the comparatively large data sets generated by the logger component. Steps were taken to ensure that the data generated from the logger component for this purpose followed a specified pattern in order to be able to debug the analyzer algorithms. The implementation of the logic related to this component had to be revisited and refined during the “beta-testing” phase due to the challenges posed by the inherently random nature of the keyboard events. In the end several qualification filters were implemented to only process the keyboard data that made logical sense and were statistically significant. The final implementation was determined to be a robust and reliable component of the “Latency Monitor” application which was able to successfully process all the logger data and generate the required latency statistics on a “per-day” and “per-week” basis.

The third phase consisted of developing the front end of the “Latency Monitor” application. The control panel of the user interface was comparatively easier to develop and test when compared to the display panel which had to be resized and refined several times farther down the development cycle. An attempt was made initially to automate the display refresh process whereby the bar charts could be continuously updated without requiring the user to initiate it. However this uncovered a limitation of the application where the logger component continued execution as a higher priority process blocking
out the display task indefinitely. Hence the application was re-designed so that the display was refreshed only when the user initiated the process using the command panel. The front end performed flawlessly as all the volunteers could use it to interact with application with ease and in addition to that were also able to display their latency dynamics at the end of their respective testing sessions without any issues.

The final phase of the project involved the six testing sessions that were carried out with the enthusiastic and committed participation of three volunteers. All the recommended procedures for carrying out the testing sessions on human subjects were followed to the letter. At the end of the testing sessions all the raw data files were preserved for later analysis and display. All the testing sessions were a success and were carried out without any notable problems or issues. Most of the volunteers were curious to know the results of the testing session and were encouraged to initiate the analysis and display process to view their latency distribution across 4 hours of the testing at the end of their sessions. Due to the intuitive and simple nature of the bar chart display that was used the volunteers were able to interpret the instances at which they had experienced the maximum latencies during the typing process. They were also able to correlate the latency peaks in the bar charts to the physiological symptoms of fatigue they had mentioned and subsequently recorded in the questionnaire. In this way they were able to appreciate the relevance and utility of the “Latency Monitor” by themselves and actually experience in real life, all the features that were mentioned previously related to the software application that was used throughout the testing sessions.
5.2 Results Overview

The keyboard event data collected for each of the volunteers was analyzed in terms of the average latency distribution across each hour on a “per-day” basis and across each day on a “per-week” basis. The “per-day” view was used to determine the onset of fatigue though each of the four hour testing sessions while the “per-week” view was aimed at estimating the cumulative effect of fatigue over two days of testing.

Three volunteers among which two were male were recruited for testing the “Latency Monitor” application. The inclusion of the female volunteer was very useful in increasing the relevance of the study by improving the diversity of the sample set. From Figure 7 it was clear that during the first day of testing volunteer 1 experienced steadily increasing keystroke latencies for the majority of the character set during the first three hours of testing. Specifically going from the second to the third hour the increase in average latencies recorded for “o”, “a” and “i” were 5 ms, 3 ms, and 3 ms respectively. On the other hand for volunteer 2 the average keystroke latencies for the majority of the character set dropped going from the first to the second hour, increased from the second to the third hour and remained relatively constant going from the third to the fourth hour. The increase in the average latencies from the second to the third hour in this volunteer’s case was 8 ms, 2 ms, 10 ms, 2 ms and 8 ms for “e”, “t”, “o”, “a” and “i” respectively. For volunteer 3 the average latencies increased gradually through each of the four hours of the testing session. In particular the average latencies appreciated by 10 ms and 8 ms for ”o” and “i” respectively going from the second to the third hour of testing.
From a comparative analysis of the first days' testing session for all the three volunteers it can be inferred that there is a definite, directly-proportional relationship between the latency experienced by the user and time spent in continuously operating the keyboard irrespective of the user involved. In particular, the period of typing spanning the second and third hours is of particular interest since the trends observed across all the volunteers' results for this particular time period are very similar. Another point to be noted is that many complaints of extreme discomfort and pain were also reported in the second hour of testing. In addition to that several instances of very high instantaneous latencies were also recorded in the second hour of the testing session.

In the same way when considering the average latency distribution for all the volunteers on the second day we can identify some trends that are common across all of them. From Figure 7 it was clear that the average latencies for Volunteer 1 increased steadily from the second through the fourth hour during the second testing session. In particular, going from the second to the third hour the average latencies for “e”, “t”, “o”, “a” and “i” increased by 1 ms, 3 ms, 10 ms, 1 ms and 7 ms respectively. However for Volunteer 2 the average latencies increased steadily from the first to the third hour and then decreased in the last hour of the testing session. In this case the average latencies going from the second to the third hour exhibited a significant spike when compared to all other hour to hour transitions. This held true even when considering all the other testing sessions irrespective of the volunteers involved. The average latencies for “e”, “t”, “o”, “a” and “i” increased by 13 ms, 33 ms, 29 ms, 41 ms and 15 ms respectively. For Volunteer 3 the average latencies decreased in the second hour, but increased steadily from the second
hour onwards to the testing session. When considering the second to third hour transition the average latencies for “e”, “t”, “o” and “a” increased by 3 ms, 4 ms, 6 ms and 1 ms respectively.

When a comparative analysis is carried out for the second day of testing across all the volunteers, common trends that were noted and highlighted previously for the first days’ results are observed in this case too. It is clear that as the duration of the time using the keyboard increases, fatigue sets in which in turn leads to a steady and gradual increase in the latencies experienced. Even though the trends going from the first to the second hour or the third to the fourth hour varied from one volunteer to the other, the results observed for the second to third hour transition remained consistent for all the volunteers.

At this point, one could begin the discussion related to an ideal threshold value for an alarm condition that could be setup within the “Latency Monitor” application to warn the users of impending fatigue. The objective would be to continuously monitor the average latencies on an hourly basis. When the percentage increase in the total aggregate average latency during an hour to hour transition exceeds a certain threshold a dialog box would open up and inform the user that they are experiencing the first stages of fatigue and need to take micro-break. Hence in this way the users could prevent the onset of fatigue by following the timely suggestions of the latency monitor application. The best choice for the ideal threshold condition could be chosen based on two primary requirements:

- Should have occurred in a time period in which the trend remained uniform across all volunteers and sessions.
Based on these guidelines and from the previous discussion on the trends of average latencies for the various volunteers we could show how a possible threshold condition could be estimated with a sufficiently large sample set. In the case of the small sample set of data that was collected in the experimental trials of this thesis it could be stated that an ideal threshold would be 3 percentage increase in the total aggregate average latency calculated for the majority of the character set that is representative of the trend. This calculation is based on the aggregate average latency increase that was estimated going from the second to the third hour for volunteer 1 which was the lowest magnitude change when compared to all the other volunteers over different testing sessions. Hence by setting up the threshold value based on Volunteer 1’s results which is at lower end of the spectrum we could ensure that alarms are generated for the occurrences of even mild symptoms of fatigue. Since a micro-break by definition lasts for a short duration the overhead incurred by adopting a conservative threshold condition could be justified in the interest of the well being of the end users. However since this threshold condition is based on the observations recorded for only three volunteers this exercise should be considered as an example of how a suitable threshold could be estimated. This in turn could be used in a more comprehensive system to warn the users of the need for a micro-break whenever the average latency exceeds the threshold condition.
Yet another method through which the latency distribution could be analyzed was by calculating the respective averages across all the volunteers. In essence the objective was to estimate the average of all the average latencies across the first sessions of testing for all the volunteers with respect to each character of interest.

Figure 16: Average of all average latencies across all volunteers on Day 1
Figure 17: Average of all average latencies across all volunteers on Day 2

Figure 16 represents the distribution of the averages of the average latencies recorded for all the volunteers across each of the testing sessions during the first day. In the same way Figure 17 represents the same distribution of the averages across all the volunteers during the second day of testing sessions. From Figures 16 and 17 it is clear that by averaging the average latencies across all volunteers for a particular day of testing we are able to
emphasize the trends represented by this metric. It is clear, from the analysis done so far that the percentage increase in the average latencies for each of the volunteers during their individual testing sessions could be marginal in some cases. However, it follows from this discussion that the same average latencies, when averaged together across all volunteers depict a sharper progression and are more representative of the appreciating trend observed so far on all the latency distributions.

Apart from the distributions of the average latencies it was also interesting to look at the instantaneous latency distribution for a single character. In essence all the latencies with respect to character “o” for Volunteer 1 during the first session of testing were plotted across time. Figure 17 and Figure 18 represent the instantaneous latencies experienced by Volunteer 1 for character “o” across four hours of testing on the first day.

![Figure 18: Volunteer1 Latency distribution for “o” during 1st & 2nd hours](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
From Figure 18 it was clear that Volunteer 1 experienced considerably high latencies during the initial part of the first hour of testing. However the latencies settled down to a moderate level from the middle of the first hour to the middle of the second hour. During the latter half of the last hour the latencies were observed to have increased to a higher level. Figure 19 showed that Volunteer 1 had experienced moderate levels of latency during the third hour and relatively higher levels of latency during the last hour of testing. In general the distribution of the instantaneous latencies were seen to correlate well with the average latency distributions discussed earlier further supporting the theory that there is a positive association between the latencies experienced and the fatigue level of the users.
5.3 Advantages

The technique adopted in this thesis to estimate the onset of fatigue in keyboard users using the “Latency Monitor” application has the following advantages.

- The “Latency Monitor” does not require any hardware components in addition to the PC peripherals to operate.

- Since there are no electronic, electrical or electro-mechanical devices required for data collection there is no risk of electric shock or injury to the users.

- For the same reason as above, the cost of the system is minimal as the only components required for collecting data, analyzing it and generating user alarms are self-contained within the software application called “Latency Monitor”.

- Due to the low cost of installation and ease of usage of the software, gathering data for very large sample populations is very much in the realm of possibility for continuing this research effort.

- Since the software has been implemented using Visual Basic 6.0 it can be run very easily on any system running Windows 98, Windows 2000 or Windows XP by simply double clicking on the executable.

- Installing and initiating the data collection process typically takes a few minutes at the most. In the same way initiating and running an analysis/display cycle takes less than a second to complete thereby enriching the user experience of the software.

- The availability of a simple and intuitive graphical user interface permitted easy comprehension and usage for the volunteers.
• The application allowed the users to both initiate the logging process and also view how they had fared in terms of average latencies over the four hours of testing through easy options provided on the user friendly GUI.

• The software was intentionally limited in its capability of logging sensitive personal information related to the users by implementing a filter in the logger task itself. This in turn allows only keystroke information related to a set five characters thereby protecting the privacy of the users.

5.4 Limitations

During the course of the development of this thesis several limitations and consequently opportunities for future improvement were identified.

• The software can be run only on workstations that had Windows operating systems installed on them. Hence this would prove to be a limitation in settings where open source operating systems such as Linux have been deployed.

• The data collection exercise could have benefited if the determination of the onset of fatigue could have been corroborated with evidence from an alternative means such as an electromyogram (EMG), 3-D Motion Analysis, etc.

• In terms of implementation, having the ability to keep track of the number of occurrences of high levels of instantaneous latencies would have been helpful in the data analysis. This is because from the results it was clear that the instances of high levels of latencies were very closely related to the physical manifestation of fatigue in the volunteers.
• It was generally observed that sometimes there was a gradual decrease in the speed of typing for all the volunteers towards the end of the testing sessions. Since by reducing speed the volunteers were in a way resting their hands, wrists, etc. the average latencies could have been impacted as a side effect of this process.

• There were several instances where the activity involved in by the user during the hour immediately preceding the testing session had a profound effect on the results obtained.

• The sample population was quite limited as only three volunteers could be recruited for the study.

• All the volunteers were in the age group of 24 to 26 years of age which limited the relevance of these results in some ways when considering the age diversity of keyboard users all over the world.

5.5 Future Work

Several steps could be taken to improve the feature set of the “Latency Monitor” application and the quality of the latency data collected through it. They are as follows:

• The software could be developed so that it can be used on any PC workstation agnostic to the native operative system employed.

• Fatigue measurements using an alternative methodology such as EMG could be used to determine an ideal threshold condition for efficient operation.
• An additional feature could be added to the software to ensure that a count is maintained of all the high latency instantaneous events that occur in any given hour.

• A means to evaluate the speed of typing could be included in the software as this data could provide additional valuable information regarding typing dynamics when overlaid on the latency distribution graphs.

• Any future study with an objective to carry the work done in this thesis forward should have some means of either controlling or determining the nature of activity the volunteer study population was involved in just before the beta phase testing sessions. This information would be vital in resolving and understanding the trends that show up in the analysis phase.

• The “Latency Monitor” application could be extended to have a web interface through which all the raw keyboard event data files could be send though email to a central collection and analysis center. In this way large study populations could be used to generate valuable data for diverse sample sets ultimately improving the quality of the latency statistics obtained.

5.6 Summary and Conclusion

In this thesis an attempt was made to develop a low cost, efficient and easily deployable system that could benefit the multitudes of keyboard users all over the world. The software that was required to achieve this objective was successfully developed and tested using a small study population. The results that were obtained supported the initial hypothesis that there exists a positive correlation between the onset of fatigue and
keyboard latency exhibited by the users. Upon a statistical analysis of the results, trends could be observed which were further supported by the physiological symptoms of fatigue that the users had recorded in the questionnaires. Based on these observations a threshold alarm condition was added to the system so that future users of the system would get feedback instructing them to take micro-breaks based on the latencies exhibited.

Even in its present form the software application developed would prove invaluable to several organizations all over the world by enabling them to proactively control the incidences of RSI and other musculoskeletal disorders among their employees. Keyboard users in general would experience higher efficiency in their output and overall improvement in their sense of well being by using this system. If some of the recommendations towards future work were implemented, the resulting tool would be a powerful, versatile and effective weapon against most common forms of upper-extremity ailments that keyboard users are affected by today.
BIBLIOGRAPHY


75

APPENDICES
APPENDIX A
Questionnaire for Volunteer1

Questionnaire

Volunteer # :  

Date : 

Instructions

Please answer the following questions and enter the option chosen in the table provided in the next page, at every thirty minute interval. The space for entries may be left empty to indicate that typing activity was not carried out during that period.

A. Can you characterize the state of your right hand at this point as

B. Can you characterize the state of your right wrist at this point as

C. Can you characterize the state of your right forearm at this point as

D. Can you characterize the state of your right upper arm at this point as

E. Can you characterize the state of your right shoulder at this point as

F. Can you characterize the state of your left hand at this point as

G. Can you characterize the state of your left wrist at this point as

H. Can you characterize the state of your left forearm at this point as

I. Can you characterize the state of your left upper arm at this point as

J. Can you characterize the state of your left shoulder at this point as
Questionnaire

Volunteer # : 1
Date : 8/10/04

Instructions

Please answer the following questions and enter the option chosen in the table provided in the next page, at every thirty minute interval. The space for entries may be left empty to indicate that typing activity was not carried out during that period.

A. Can you characterize the state of your right hand at this point as

B. Can you characterize the state of your right wrist at this point as

C. Can you characterize the state of your right forearm at this point as

D. Can you characterize the state of your right upper arm at this point as

E. Can you characterize the state of your right shoulder at this point as

F. Can you characterize the state of your left hand at this point as

G. Can you characterize the state of your left wrist at this point as

H. Can you characterize the state of your left forearm at this point as

I. Can you characterize the state of your left upper arm at this point as

J. Can you characterize the state of your left shoulder at this point as
<table>
<thead>
<tr>
<th>Time</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0830</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0900</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0930</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1330</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1430</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1530</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1630</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1700</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

Volunteer experienced some fatigue in the 1st hour itself and took a break at the end of the first hour.
APPENDIX B

Questionnaire for Volunteer 2

Questionnaire

Volunteer # : 2

Date : 1/1/07

Instructions

Please answer the following questions and enter the option chosen in the table provided in the next page, at every thirty minute interval. The space for entries may be left empty to indicate that typing activity was not carried out during that period.

A. Can you characterize the state of your right hand at this point as

B. Can you characterize the state of your right wrist at this point as

C. Can you characterize the state of your right forearm at this point as

D. Can you characterize the state of your right upper arm at this point as

E. Can you characterize the state of your right shoulder at this point as

F. Can you characterize the state of your left hand at this point as

G. Can you characterize the state of your left wrist at this point as

H. Can you characterize the state of your left forearm at this point as

I. Can you characterize the state of your left upper arm at this point as

J. Can you characterize the state of your left shoulder at this point as
<table>
<thead>
<tr>
<th>Time</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0830</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0900</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0930</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1330</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1430</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1500</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1530</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1600</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1630</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1700</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Breaks are taken in the final hour.
Questionnaire

Volunteer # : 2
Date : 8/18/007

Instructions

Please answer the following questions and enter the option chosen in the table provided in the next page, at every thirty minute interval. The space for entries may be left empty to indicate that typing activity was not carried out during that period.

A. Can you characterize the state of your right hand at this point as

B. Can you characterize the state of your right wrist at this point as

C. Can you characterize the state of your right forearm at this point as

D. Can you characterize the state of your right upper arm at this point as

E. Can you characterize the state of your right shoulder at this point as

F. Can you characterize the state of your left hand at this point as

G. Can you characterize the state of your left wrist at this point as

H. Can you characterize the state of your left forearm at this point as

I. Can you characterize the state of your left upper arm at this point as

J. Can you characterize the state of your left shoulder at this point as
<table>
<thead>
<tr>
<th>Time</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0830</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0900</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0930</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1330</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1430</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1530</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1630</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1700</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

Data was taken in the Grand Guild Room.
APPENDIX C
Questionnaire for Volunteer3

Questionnaire

Instructions

Please answer the following questions and enter the option chosen in the table provided in the next page, at every thirty minute interval. The space for entries may be left empty to indicate that typing activity was not carried out during that period.

A. Can you characterize the state of your right hand at this point as

B. Can you characterize the state of your right wrist at this point as

C. Can you characterize the state of your right forearm at this point as

D. Can you characterize the state of your right upper arm at this point as

E. Can you characterize the state of your right shoulder at this point as

F. Can you characterize the state of your left hand at this point as

G. Can you characterize the state of your left wrist at this point as

H. Can you characterize the state of your left forearm at this point as

I. Can you characterize the state of your left upper arm at this point as

J. Can you characterize the state of your left shoulder at this point as

Volunteer #: 8
Date: 8/18/07
Questionnaire

Volunteer #:

Date:

Instructions

Please answer the following questions and enter the option chosen in the table provided in the next page, at every thirty minute interval. The space for entries may be left empty to indicate that typing activity was not carried out during that period.

A. Can you characterize the state of your right hand at this point as

B. Can you characterize the state of your right wrist at this point as

C. Can you characterize the state of your right forearm at this point as

D. Can you characterize the state of your right upper arm at this point as

E. Can you characterize the state of your right shoulder at this point as

F. Can you characterize the state of your left hand at this point as

G. Can you characterize the state of your left wrist at this point as

H. Can you characterize the state of your left forearm at this point as

I. Can you characterize the state of your left upper arm at this point as

J. Can you characterize the state of your left shoulder at this point as
<table>
<thead>
<tr>
<th>Time</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0830</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0900</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0930</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1330</td>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1400</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1430</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1500</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1530</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1600</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1630</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1700</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Feel relaxed towards the end of the 1st hr.
' Form1.frm
Option Explicit
Private Sub cboViewCtrl1_Click()
    Dim i As Integer
    If (cboViewCtrl1.Text = "Per Week") Then
        lblTimePnt(0).Caption = "Day1"
        lblTimePnt(1).Caption = "Day2"
        lblTimePnt(2).Caption = "Day3"
        lblTimePnt(3).Caption = "Day4"
        lblTimePnt(4).Caption = "Day5"
        lblTimePnt(5).Caption = "Day6"
        lblTimePnt(6).Caption = "Day7"
        lblTimePnt(7).Visible = False
        lblTimePnt(8).Visible = False
        lblTimePnt(9).Visible = False
        lblTimePnt(10).Visible = False
        cboViewCtrl2.Visible = False
        lblDayOfWeek.Visible = False
        For i = 0 To 49
            If (i > 34) Then
                lblTmPnt(i).Visible = False
                lblTmPnt2(i).Visible = False
                lblTmPnt3(i).Visible = False
            Else
                lblTmPnt(i).Top = 8280
                lblTmPnt(i).Height = 10
                lblTmPnt2(i).Top = 8280
                lblTmPnt2(i).Height = 10
                lblTmPnt3(i).Top = 8280
                lblTmPnt3(i).Height = 10
            End If
        Next i
        Label2.Caption = "DAY OF WORK WEEK"
    Else
        For i = 0 To 49
            If (i > 34) Then
                lblTmPnt(i).Visible = False
                lblTmPnt2(i).Visible = False
                lblTmPnt3(i).Visible = False
            Else
                lblTmPnt(i).Top = 8280
                lblTmPnt(i).Height = 10
                lblTmPnt2(i).Top = 8280
                lblTmPnt2(i).Height = 10
                lblTmPnt3(i).Top = 8280
                lblTmPnt3(i).Height = 10
            End If
        Next i
        Label2.Caption = "DAY OF WORK WEEK"
        For i = 0 To 49
            If (i > 34) Then
                lblTmPnt(i).Visible = False
                lblTmPnt2(i).Visible = False
                lblTmPnt3(i).Visible = False
            Else
                lblTmPnt(i).Top = 8280
                lblTmPnt(i).Height = 10
                lblTmPnt2(i).Top = 8280
                lblTmPnt2(i).Height = 10
                lblTmPnt3(i).Top = 8280
                lblTmPnt3(i).Height = 10
            End If
        Next i
        Label2.Caption = "DAY OF WORK WEEK"
    End If
Next i
Label2.Caption = "DAY OF WORK WEEK"
Else
lblTimePnt(0).Caption = "8am"
lblTimePnt(1).Caption = "9am"

lblTimePnt(2).Caption = "10am"
lblTimePnt(3).Caption = "11am"

lblTimePnt(4).Caption = "12pm"
lblTimePnt(5).Caption = "1pm"
lblTimePnt(6).Caption = "2pm"
lblTimePnt(7).Visible = True
lblTimePnt(8).Visible = True
lblTimePnt(9).Visible = True
lblTimePnt(10).Visible = True
cboViewCtrl2.Visible = True
lblDayOfWeek.Visible = True

For i = 0 To 49
    If (i > 34) Then
        lblTmPnt(i).Visible = True
        lblTmPnt2(i).Visible = True
        lblTmPnt3(i).Visible = True
    Else
        lblTmPnt(i).Top = 8280
        lblTmPnt(i).Height = 10
        lblTmPnt2(i).Top = 8280
        lblTmPnt2(i).Height = 10
        lblTmPnt3(i).Top = 8280
        lblTmPnt3(i).Height = 10
    End If
Next i
Label2.Caption = "HOUR OF WORK DAY"
End If
End Sub

Private Sub Form_Load()
cboViewCtrl1.AddItem "Per Day"
cboViewCtrl1.AddItem "Per Week"
cboViewCtrl2.AddItem "Day 1"
cboViewCtrl2.AddItem "Day 2"
cboViewCtrl2.AddItem "Day 3"
End Sub

Private Sub Command1_Click()
HookKeyboard
End Sub

Private Sub Command2_Click()
    UnhookKeyboard
End Sub

Private Sub Command4_Click()
    AnalyzeData
    'ModifyData
End Sub

Private Sub Command5_Click()
    If (cboViewCtrl1.Text = "Per Day") Then
        If (cboViewCtrl2.Text = "Day 1") Then
            ClearGraph
            DisplayData (1)
        End If
        If (cboViewCtrl2.Text = "Day 2") Then
            ClearGraph
            DisplayData (2)
        End If
        If (cboViewCtrl2.Text = "Day 3") Then
            ClearGraph
            DisplayData (3)
        End If
    Else
        ClearGraph
        DispPerWeekGraph
        End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    UnhookKeyboard
End Sub

Private Sub ClearGraph()
    Dim i As Integer
    For i = 0 To 49
        lblTmPnt(i).Top = 8280
        lblTmPnt(i).Height = 10
        lblTmPnt2(i).Top = 8280
        lblTmPnt2(i).Height = 10
        lblTmPnt3(i).Top = 8280
    Next i
End Sub
lblTmPnt3(i).Height = 10
Next i
End Sub

Keyboardhandler.bas

Option Explicit

Public Declare Function UnhookWindowsHookEx Lib "user32" _
(ByVal hHook As Long) As Long

Public Declare Function SetWindowsHookEx Lib "user32" _
Alias "SetWindowsHookExA" (ByVal idHook As Long, _
ByVal lpfn As Long, _
ByVal hmod As Long, _
ByVal dwThreadId As Long) As Long

Private Declare Sub CopyMemory Lib "kernel32" _
Alias "RtlMoveMemory" _
(pDest As Any, _
pSource As Any, _
ByVal cb As Long)

Private Declare Function CallNextHookEx Lib "user32" _
(ByVal hHook As Long, _
ByVal nCode As Long, _
ByVal wParam As Long, _
ByVal lParam As Long) As Long

Private Type KBDLHOOKSTRUCT
    vkCode As Long
    scanCode As Long
    flags As Long
    time As Long
    dwExtraInfo As Long
End Type

Private Type CHARINFO
    totNum As Long
End Type
totLatency As Long
avgLatency As Long
maxLatency As Long
minLatency As Long
numOfLatAboveStd As Long
End Type

Private Type FILEINFO
flevkCode As Long
intTimeWrite As Integer
intDayWrite As Integer
flewparam As Long
fletime As Long
End Type

' Type for storing relevant information in the Day Store Info file
Private Type DAYSTOREINFO
dayOfWeek As Integer ' We store this here cause we 'll use it to navigate to
' the next day in the per day view
hourOfDay As Integer
lngChrAvgLatPerHr(4) As Integer
lngChrMaxLatPerHr(4) As Integer
lngChrMinLatPerHr(4) As Integer
End Type

' Type for storing relevant information in the Week Store Info file
Private Type WEEKSTOREINFO
dayOfWeek As Integer
lngChrAvgLatPerDay(4) As Integer
lngChrMaxLatPerDay(4) As Integer
lngChrMinLatPerDay(4) As Integer
End Type

' Low-Level Keyboard Constants
Private Const HC_ACTION = 0
Private Const WH_KEYBOARD_LL = 13&
' Function declarations
Public KeyboardHandle As Long
' Integer variables used
Dim strTimeWrite As String * 2
Dim wParamRead As Long
Dim intTimeRead As Integer
Dim intDayStrTotRecs As Integer
Dim intWeekStrTotRecs As Integer

' Analysis variables
Dim cdIdx As Integer
Dim tmIdx As Integer
'Debug variables
Dim tmpDbug As Integer

Public Function KeyboardCallback(ByVal Code As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

Static Hookstruct As KBDLLHOOKSTRUCT
Static fleInf As FILEINFO
Static totNumOfRecs As Long
Static firstTimeAccess As Integer
Dim intRfNum As Integer
Dim intCtr As Integer ' Loop counter
'Debug constants

intRfNum = FreeFile()

If (Code = HC_ACTION) Then
    ' Copy the keyboard data out of the lParam (which is a pointer)
    Call CopyMemory(Hookstruct, ByVal lParam, Len(Hookstruct))
    If ((Hookstruct.vkCode = 69) Or (Hookstruct.vkCode = 84) Or (Hookstruct.vkCode = 79) Or (Hookstruct.vkCode = 65) Or (Hookstruct.vkCode = 73)) Then
        ' Update the totNumofRecs variable to keep track of the EOF of the raw file
        ' Store it always in the first fletime variable of the first record
        Open "C:\logs\Volunteer1.txt" For Random As #intRfNum Len = 24
        ' Check if this is the first time the call back function is being executed
        ' This piece of code is especially useful when we are carrying out data collection
        ' over several days or sessions. We would like the data collection to resume from
        ' the last record that was recorded
        ' So we check if this is the first time KeyboardCallback is being called. If this
        ' is the first time then we get the first record and check if the totNumOfRecs is
        ' 0 to see if this is the first session. If so we intialize the totRecs to 1 to
        ' account for the first record always holding the totNumOfRecs. Otherwise we
        ' intialize the last known totNumOfRecs to resume data collection from where
        ' we left off in the previous session.
        If (firstTimeAccess = 0) Then
            Get #intRfNum, 1, fleInf
            If (fleInf.fletime = 0) Then
                totNumofRecs = 1 'Accounting for the double increment being done later
            Else
                totNumofRecs = fleInf.fletime
                'totNumofRecs = 13333
            End If
            firstTimeAccess = firstTimeAccess + 1
End If

95
End If

totNumofRecs = totNumofRecs + 1
fleInf.fletime = totNumofRecs
fleInf.flevkCode = 0
fleInf.flewparam = 0
Put #intRfNum, 1, fleInf ' tot number of records stored in first record

strTimeWrite = Format(time, "hh")
'Time is adjusted to account for the difference in the time that is to be intended to be recorded

fleInf.intTimeWrite = CInt(strTimeWrite)
fleInf.intDayWrite = Day(Now)
fleInf.flevkCode = Hookstruct.vkCode
fleInf.flewparam = wParam
fleInf.fletime = Hookstruct.time
'Store the record according to the number calculated above
Put #intRfNum, totNumofRecs, fleInf
Close #intRfNum
End If

End If

KeyboardCallback = CallNextHookEx(KeyboardHandle, Code, wParam, lParam)

End Function

Public Sub HookKeyboard()
KeyboardHandle = SetWindowsHookEx(WH_KEYBOARD_LL, AddressOf KeyboardCallback, App.hInstance, 0&)

Call CheckHooked

End Sub

Public Sub CheckHooked()
If (Hooked) Then
    Debug.Print "Keyboard hooked"
Else
    Debug.Print "Keyboard hook failed: " & Err.LastDllError
End If
End Sub

96
Private Function Hooked()
  Hooked = KeyboardHandle <> 0
End Function

Public Sub UnhookKeyboard()
  Static intAlreadyDone As Integer

  If (intAlreadyDone) Then
    Exit Sub
  Else
    If (Hooked) Then
      Call UnhookWindowsHookEx(KeyboardHandle)
      Debug.Print "Keyboard Unhooked"
    End If
    intAlreadyDone = 1
  End If
End Sub

Public Sub ModifyData()
  Dim intRNo As Integer
  Dim upprLmt As Long
  Dim intRecNo As Integer
  Dim fleChng As FILEINFO

  intRNo = FreeFile()
  upprLmt = 13333
  Open "C:\logs\Volunteer.txt" For Random As #intRNo Len = 24
  For intRecNo = 2 To upprLmt
    Get #intRNo, intRecNo, fleChng
    If (fleChng.intDayWrite = 5) Then
      fleChng.intDayWrite = 1
      Put #intRNo, intRecNo, fleChng
    End If
  Next intRecNo
  Close #intRNo
End Sub

Public Sub AnalyzeData()
  Dim charDataPerHr(4) As CHARINFO
  Dim charDataPerDay(4) As CHARINFO
  Dim fleInfRetrieve As FILEINFO
  Dim fleInfRetrivNxt As FILEINFO
  Dim fleInfRetrivNxtNxt As FILEINFO
  Dim fieWeekStore As WEEKSTOREINFO
  Dim totNumofRecs As Long
Dim prevLatTimeRead As Long
Dim intRfNum As Integer
Dim intCtr As Long ' Record counter for the reading case
Dim intCmmnLoopCtr As Integer 'Common Loop ctr used for the various assignments
Dim resetCtrLoopIdx As Integer 'For resetting the running counters

Dim intVal As Integer ' Read value
Dim lngNxtChrTme As Long
Dim intCurChrInstLat As Long
Dim intPrHrMinMxInitCtrl(4) As Integer
Dim intPrDayMinMxInitCtrl(4) As Integer

' Dim intRfNum3 As Integer

intRfNum = FreeFile()
Open "C:\logsVolunteer.txt" For Random As #intRfNum Len = 24
'Retrieving first record to get the totNumofRecs for EOF
Get #intRfNum, 1, flelnfRetrieve
totNumofRecs = flelnfRetrieve.fletime
'Since the next record is being accessed also we got to keep the
counter value pointing to the last record
For intCtr = 2 To (totNumofRecs - 1)
  Get #intRfNum, intCtr, flelnfRetrieve
  Get #intRfNum, (intCtr + 1), flelnfRetrivNxt
  
  'Update event info only if the next event was a key release also related to the same
  key event
  'This is to ensure that no funny sequences are processed and are simply abandoned.
  If ((flelnfRetrieve.flewparam = 256) And (flelnfRetrivNxt.flewparam = 257) _
    And (flelnfRetrieve.flevkCode = flelnfRetrivNxt.flevkCode) _
    And (flelnfRetrieve.fletime <> flelnfRetrivNxt.fletime)) Then
    Select Case flelnfRetrieve.flevkCode
      Case 69
        cdldx = 0
      Case 84
        cdldx = 1
      Case 79
        cdldx = 2
      Case 65
        cdldx = 3
      Case 73
        cdldx = 4
    End Select

98

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
charDataPerHr(cdIdx).totNum = charDataPerHr(cdIdx).totNum + 1
intCurChrInstLat = fleInfRetrieveNxt.fletime - fleInfRetrieve.fletime
'If (intCurChrInstLat = 0) Then
'  tmpDbg = tmpDbg + 1
'End If

'If (fleInfRetrieve.intTimeWrite = 15) Then
'  tmpDbg = 1
'End If
If (intCurChrInstLat > 130) Then
  charDataPerHr(cdIdx).numOfLatAbveStd = charDataPerHr(cdIdx).numOfLatAbveStd + 1
End If

charDataPerHr(cdIdx).totLatency = charDataPerHr(cdIdx).totLatency +
intCurChrInstLat
charDataPerHr(cdIdx).avgLatency = charDataPerHr(cdIdx).totLatency /
charDataPerHr(cdIdx).totNum

'Running counters maintained to hold max and min latencies too
If (intPrHrMinMxInitCtrl(cdIdx) = 0) Then
  charDataPerHr(cdIdx).maxLatency = intCurChrInstLat 'initialize the vals
  charDataPerHr(cdIdx).minLatency = intCurChrInstLat 'for the very first
  intPrHrMinMxInitCtrl(cdIdx) = intPrHrMinMxInitCtrl(cdIdx) + 1
End If

If (charDataPerHr(cdIdx).maxLatency < intCurChrInstLat) Then
  charDataPerHr(cdIdx).maxLatency = intCurChrInstLat
End If
If (charDataPerHr(cdIdx).minLatency > intCurChrInstLat) Then
  charDataPerHr(cdIdx).minLatency = intCurChrInstLat
End If

charDataPerDay(cdIdx).totNum = charDataPerDay(cdIdx).totNum + 1
charDataPerDay(cdIdx).totLatency = charDataPerDay(cdIdx).totLatency +
intCurChrInstLat
charDataPerDay(cdIdx).avgLatency = charDataPerDay(cdIdx).totLatency /
charDataPerDay(cdIdx).totNum
If (intPrDayMinMxInitCtrl(cdIdx) = 0) Then
  charDataPerDay(cdIdx).maxLatency = intCurChrInstLat 'initialize the vals
  charDataPerDay(cdIdx).minLatency = intCurChrInstLat 'for the very first
  intPrDayMinMxInitCtrl(cdIdx) = intPrDayMinMxInitCtrl(cdIdx) + 1
End If

If (charDataPerDay(cdIdx).maxLatency < intCurChrInstLat) Then
  charDataPerDay(cdIdx).maxLatency = intCurChrInstLat
End If
End If

If (charDataPerDay(cdIdx).minLatency > intCurChrinstLat) Then
  charDataPerDay(cdIdx).minLatency = intCurChrInstLat
'If (charDataPerDay(cdIdx).minLatency = 0) Then
  tmpDbug = 1
'End If

End If

End If’ For wparam = 257 condn.To check and process only press events

'We need a third counter to access the next "key press event. We also should
'have the following condition to make sure we end the inspection of the
'next-next event before we reach the end of data string. We need to inspect
'the next-next event for dumping the current running counters to memory and
'resetting the current running counters to get ready for the next hour
'recording and analyzing data
If (((intCtr + 2) < totNumofRecs) Then
  'Get #intRfNum, (intCtr + 2), fleInfRetrivNxtNxt
  'Using "not equal to" (<> ) cause we may have a day jump in between. So
  'the next record may contain 8am after 12 pm being processed here
  If (fleInfRetrieve.intTimeWrite <> fleInfRetrivNxt.intTimeWrite) Then
    tmpDbug = tmpDbug + 1
  fleDayStore.hourOfDay = fleInfRetrieve.intTimeWrite
  fleDayStore.dayOfWeek = fleInfRetrieve.intDayWrite
  For intCmmnLoopCtr = 0 To 4
    fleDayStore.lngChrAvgLatPerHr(intCmmnLoopCtr) = _
    charDataPerHr(intCmmnLoopCtr).avgLatency
    fleDayStore.lngChrMaxLatPerHr(intCmmnLoopCtr) = _
    charDataPerHr(intCmmnLoopCtr).maxLatency
    fleDayStore.lngChrMinLatPerHr(intCmmnLoopCtr) = _
    charDataPerHr(intCmmnLoopCtr).minLatency
  Next intCmmnLoopCtr
  Call StoreDaylnfo(fleDayStore)
  'Resetting counters
  For resetCtrLoopIdx = 0 To 4
    charDataPerHr(resetCtrLoopIdx).avgLatency = 0
    charDataPerHr(resetCtrLoopIdx).maxLatency = 0
    charDataPerHr(resetCtrLoopIdx).minLatency = 0
    charDataPerHr(resetCtrLoopIdx).totLatency = 0
    charDataPerHr(resetCtrLoopIdx).totNum = 0
    charDataPerHr(resetCtrLoopIdx).numOfLatAbveStd = 0
    intPrHrMinMxInitCtrl(resetCtrLoopIdx) = 0 'This should ensure that the
    min/max

100

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Next resetCtrLoopIdx 'variables are reinitialized for next hour
End If

If (fleInfRetrieve.intDayWrite <> fleInfRetrieveNxt.intDayWrite) Then
fleWeekStore.dayOfWeek = fleInfRetrieve.intDayWrite
For intCmmnLoopCtr = 0 To 4
fleWeekStore.lngChrAvgLatPerDay(intCmmnLoopCtr) = _
charDataPerDay(intCmmnLoopCtr).avgLatency
fleWeekStore.lngChrMaxLatPerDay(intCmmnLoopCtr) = _
charDataPerDay(intCmmnLoopCtr).maxLatency
fleWeekStore.lngChrMinLatPerDay(intCmmnLoopCtr) = _
charDataPerDay(intCmmnLoopCtr).minLatency
Next intCmmnLoopCtr
Call StoreWeeklnfo(fleWeekStore)
'Resetting counters
For resetCtrLoopIdx = 0 To 4
charDataPerDay(resetCtrLoopIdx).avgLatency = 0
charDataPerDay(resetCtrLoopIdx).maxLatency = 0
charDataPerDay(resetCtrLoopIdx).minLatency = 0
charDataPerDay(resetCtrLoopIdx).totLatency = 0
charDataPerDay(resetCtrLoopIdx).totNum = 0
intPrDayMinMxInitCtrl(resetCtrLoopIdx) = 0 'This should ensure that
the min/max
Next resetCtrLoopIdx 'variables are reinitialized for next day
End If

End If 'End of Hour of Time & Day of Week switch event code

Next intCtr
Close #intRfNum
'Call DisplayData
End Sub

Public Sub StoreDayInfo(strctDayStrInfo As DAYSTOREINFO)
Dim intRfNum2 As Integer 'Used to open the Day Info Store File

intRfNum2 = FreeFile()
'Store all relevant day / hr info for all the char in the file
' adressed by Rfnum2
'Form1.pmtStoreTxt = Form1.pmtStoreTxt & strctDayStrlnfo.hourOfDay _
'& "." & strctDayStrlnfo.lngChrAvgLatPerHr(0) & "." & strctDayStrlnfo.lngChrAvgLatPerHr(1) _
'"." & strctDayStrlnfo.lngChrAvgLatPerHr(2) & "." & strctDayStrlnfo.lngChrAvgLatPerHr(3) _
'& "." & strctDayStrlnfo.lngChrAvgLatPerHr(4) & " "

101

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Open "C:\logs\DayStore.txt" For Random As #intRfNum2 Len = 68
intDayStrTotRecs = intDayStrTotRecs + 1 'Updating the tot num of day info store recs
'Writing all day/hour avgLatency into day store file
Put #intRfNum2, intDayStrTotRecs, strctDayStrInfo
Close #intRfNum2
End Sub

Public Sub StoreWeekInfo(strctWeekStrInfo As WEEKSTOREINFO)
Dim intRfNum3 As Integer 'Used to open the Week Info Store File

intRfNum3 = FreeFile()
'Store all relevant day/hr info for all the char in the file
'addressed by Rfnum2
Open "C:\logs\WeekStore.txt" For Random As #intRfNum3 Len = 68
intWeekStrTotRecs = intWeekStrTotRecs + 1 'Updating the tot num of day info store recs
'Writing all day/hour avgLatency into day store file
Put #intRfNum3, intWeekStrTotRecs, strctWeekStrInfo
Close #intRfNum3
End Sub

Public Sub DisplayData(dayOfWeek As Integer)
Static intCurStepVal As Integer
Static intDfltTop As Integer
Static intDfltHt As Integer
Dim intRfNum4 As Integer 'Used to open the Day Info Store File
Dim intDayStrRecdlndx As Integer
Dim intLoopDiff As Integer
Dim strctDayStrInfoDisp As DAYSTOREINFO
Dim intVarAssgnlndx As Integer
Dim intDispStartRecIdx As Integer
Dim intDispEndRecIdx As Integer

intCurStepVal = 24 'Used to control the step increment for the latency rectangles
intDfltTop = 8280
intDfltHt = 10

'Get an argument passed in from the form to display the per day view on a day
to day basis. Number 1 will be passed in for Day 1, 2 for Day 2 etc
'Based on the day per week the record access index pointers can be set to access
'all the records relate to a particular day.
'Here the assumption that there are only going to be five records in a day is made here
by assigning
'the record start and end indices here.
'The indices were modified to match up with the requirement for four hour testing
sessions
If (dayOfWeek = 1) Then
intDispStartRecIdx = 1
intDispEndRecIdx = 4
ElseIf (dayOfWeek = 2) Then
    intDispStartRecIdx = 5
    intDispEndRecIdx = 8
Else
    intDispStartRecIdx = 9
    intDispEndRecIdx = 12
End If

intRfNum4 = FreeFile()
'Read all relevant day / hr info for all the char in the file
'addressed by RfNum3
Open "C:\logs\DayStore.txt" For Random As #intRfNum4 Len = 68
For intDayStrRecdlndx = intDispStartRecIdx To intDispEndRecIdx
    Get #intRfNum4, intDayStrRecdlndx, strctDayStrlnfoDisp
    tmpDbug = tmpDbug + 1
    'Assign the correct avg latency values to the associated display time group
    'eg. for timeofhour 8 this would be (8-8)*5 = 0 which is dispTmpnt (0) grp
    'Basically these variables identifies the display time grp for these set
    'of values read back from the day store file
    intLoopDiff = (strctDayStrInfoDisp.hourOfDay - 8) * 5
    For intVarAssgnlndx = 0 To 4
        If (strctDayStrlnfoDisp.lngChrAvgLatPerHr(intVarAssgnlndx) < 301) Then
            Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Height = intDfltHt + _
               (intCurStepVal)
            Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Top = intDfltTop - _
               (intCurStepVal)
        End If
        If (strctDayStrlnfoDisp.lngChrAvgLatPerHr(intVarAssgnlndx) > 300) Then
            Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Height = (intCurStepVal * 300)
            Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Top = 1092
        End If
    End For
End If
End If
If (strctDayStrInfoDisp.lngChrMinLatPerHr(intVarAssignlndx) < 301) Then
    Form1.lblTmPnt3(intLoopDiff + intVarAssignIndx).Height = (intDfltHt * 6)
    Form1.lblTmPnt3(intLoopDiff + intVarAssignIndx).Top = intDfltTop - _
        (strctDayStrInfoDisp.lngChrMinLatPerHr(intVarAssignlndx) * intCurStepVal)
End If

If (strctDayStrInfoDisp.lngChrMinLatPerHr(intVarAssignlndx) > 300) Then
    Form1.lblTmPnt3(intLoopDiff + intVarAssignIndx).Height = (intDfltHt * 6)
    Form1.lblTmPnt3(intLoopDiff + intVarAssignIndx).Top = 1092
End If

Next intVarAssignlndx
Next intDayStrRecdlndx
Close #intRfNum4
End Sub

Public Sub DispPerWeekGraph()
    Static intCurStepVal As Integer
    Static intDfltTop As Integer
    Static intDfltHt As Integer
    Dim intRfNum5 As Integer 'Used to open the Week Info Store File
    Dim intWeekStrRecdIndx As Integer
    Dim intLoopDiff As Integer
    Dim strctWeekStrlnfoDisp As WEEKSTOREINFO
    Dim intVarAssignIndx As Integer
    Dim intLoopIntializeDecVar As Integer

    intCurStepVal = 24 'Used to control the step increment for the latency rectangles
    intDfltTop = 8280
    intDfltHt = 10

    intRfNum5 = FreeFile()
    'Read all relevant day / hr info for all the char in the file
    'addressed by Rfhum3
    Open "C:\logs\WeekStore.txt" For Random As #intRfNum5 Len = 68
    For intWeekStrRecdlndx = 1 To intWeekStrTotRecs
        Get #intRfNum5, intWeekStrRecdlndx, strctWeekStrlnfoDisp
        strctWeekStrlnfoDisp.dayOfWeek = strctWeekStrlnfoDisp.dayOfWeek
        intLoopDiff = (strctWeekStrlnfoDisp.dayOfWeek - intLoopIntializeDecVar) * 5
        If (intWeekStrRecdlndx = 1) Then
            intLoopIntializeDecVar = strctWeekStrlnfoDisp.dayOfWeek
            tmpDbg = tmpDbg + 1
            If (intWeekStrRecdlndx = 1) Then
                intLoopIntializeDecVar = strctWeekStrlnfoDisp.dayOfWeek
            End If
        'Assign the correct avg latency values to the associated display time group
        'eg. for timeofhour 8 this would be (8-8)*5 = 0 which is dispTmpnt (0) grp
        'Basically this variables identifies the display time grp for these set
        'of values read back from the day store file
        intLoopDiff = (strctWeekStrlnfoDisp.dayOfWeek - intLoopIntializeDecVar) * 5

        intLoopDiff = (strctWeekStrlnfoDisp.dayOfWeek - intLoopIntializeDecVar) * 5
    Next intWeekStrRecdIndx
Close #intRfNum5
For intVarAssgnlndx = 0 To 4
    If (strctWeekStrInfoDisp.lngChrAvgLatPerDay(intVarAssgnlndx) < 301) Then
        Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Height = intDfltHt + (strctWeekStrInfoDisp.lngChrAvgLatPerDay(intVarAssgnlndx) * intCurStepVal)
        Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Top = intDfltTop - (strctWeekStrInfoDisp.lngChrAvgLatPerDay(intVarAssgnlndx) * intCurStepVal)
    End If
    If (strctWeekStrInfoDisp.lngChrAvgLatPerDay(intVarAssgnlndx) > 300) Then
        Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Height = (intCurStepVal * 300)
        Form1.lblTmPnt(intLoopDiff + intVarAssgnlndx).Top = 1092
    End If
If (strctWeekStrInfoDisp.lngChrMaxLatPerDay(intVarAssgnlndx) < 301) Then
    Form1.lblTmPnt2(intLoopDiff + intVarAssgnlndx).Height = (intDfltHt * 6)
    Form1.lblTmPnt2(intLoopDiff + intVarAssgnlndx).Top = intDfltTop - (strctWeekStrInfoDisp.lngChrMaxLatPerDay(intVarAssgnlndx) * intCurStepVal)
    Debug.Print Form1.lblTmPnt2(0).FillColor
    Debug.Print Form1.lblTmPnt(0).FillColor
End If
    If (strctWeekStrInfoDisp.lngChrMaxLatPerDay(intVarAssgnlndx) > 300) Then
        Form1.lblTmPnt2(intLoopDiff + intVarAssgnlndx).Height = (intCurStepVal * 6)
        Form1.lblTmPnt2(intLoopDiff + intVarAssgnlndx).Top = 1092
    End If
    If (strctWeekStrInfoDisp.lngChrMinLatPerDay(intVarAssgnlndx) < 301) Then
        Form1.lblTmPnt3(intLoopDiff + intVarAssgnlndx).Height = (intDfltHt * 6)
        Form1.lblTmPnt3(intLoopDiff + intVarAssgnlndx).Top = intDfltTop - (strctWeekStrInfoDisp.lngChrMinLatPerDay(intVarAssgnlndx) * intCurStepVal)
    End If
    If (strctWeekStrInfoDisp.lngChrMinLatPerDay(intVarAssgnlndx) > 300) Then
        Form1.lblTmPnt3(intLoopDiff + intVarAssgnlndx).Height = (intCurStepVal * 6)
        Form1.lblTmPnt3(intLoopDiff + intVarAssgnlndx).Top = 1092
    End If
Next intVarAssgnlndx
Next intWeekStrRecdIndx
Close #intRfNum5

'For intLoop = 0 To 49
    'Form1.lblTmPnt(intLoop).Height = intDfltHt + ((90 - 60) * intCurStepVal)
    'Form1.lblTmPnt(intLoop).Top = intDfltTop - ((90 - 60) * intCurStepVal)
'Next intLoop
'Form1.lblTmPnt(0).Height = intDfltHt + ((90 - 60) * intCurStepVal)
'Form1.lblTmPnt(0).Top = intDfltTop - ((90 - 60) * intCurStepVal)

End Sub
APPENDIX E

IRB Approval

University of New Hampshire
Research Conduct and Compliance Services, Office of Sponsored Research
Service Building, 51 College Road, Durham, NH 03824-3585
Fax: 603-862-3564

03-Nov-2006

Jayakumar, Satheesh
Electrical and Computer Engineering
317 Forest Park
5 Lakeview Ave, Reading, MA 01867
Durham, NH 03824

IRB #: 3810
Study: Non-invasive Keyboard Fatigue Monitoring System for Improving User Performance
and Reducing Incidences of Repetitive Strain Injuries
Approval Date: 03-Nov-2006

The Institutional Review Board for the Protection of Human Subjects in Research (IRB) has
reviewed and approved the protocol for your study as Expedited as described in Title 45,

Approval is granted to conduct your study as described in your protocol for one
year from the approval date above. At the end of the approval period, you will be
asked to submit a report with regard to the involvement of human subjects in this study. If
your study is still active, you may request an extension of IRB approval.

Researchers who conduct studies involving human subjects have responsibilities as outlined
in the attached document, Responsibilities of Directors of Research Studies Involving
Human Subjects. (This document is also available at
http://www.unh.edu/osr/compliance/irb.html.) Please read this document carefully before
commencing your work involving human subjects.

If you have questions or concerns about your study or this approval, please feel free to
contact me at 603-862-2003 or julie.simpson@unh.edu. Please refer to the IRB # above in
all correspondence related to this study. The IRB wishes you success with your research.

For the IRB,

Julie F. Simpson
Manager

cc: File
   Lacourse, John