Pinpoint Analysis of Software Usability

Divya K. V. Dasari, Dan E. Tamir,
Oleg V. Komogortsev, Gregory R. LaKomski
Department of Computer Science
Texas State University
San Marcos, Texas USA
{dd1290, dt19, ok11, gl1082}@txstate.edu

Carl J. Mueller
Texas A&M University Central Texas
Killeen, Texas, USA
muellercj@ct.tamus.edu

***Abstract*—The effort-based model of usability aids in evaluating user interface (UI), development of usable software, and pinpointing software usability defects. In this context, the term pinpoint analysis refers to identifying and locating software usability issues and correlating these issues with the UI software code. In this paper, the underlying theory of the effort-based model along with pattern recognition techniques are used to produce a framework for pinpointing usability deficiencies in software via automatic classification of segments of video file containing eye tracking results. This allows developers to harness their effort and focus on excessive effort segments that need attention. To verify the results of the pattern recognition procedures, the video is manually classified into excessive and non-excessive segments and the results of automatic and manual classification are compared. The paper details the theory of effort-based pinpoint analysis and reports on experiments performed to evaluate the utility of this theory. Experiment results show more than 40% reduction in time for usability testing.**

Keywords-Software Development; Software Usability; Human Computer Interaction; Pinpoint Analysis.

I. INTRODUCTION

The *Effort-based model* of usability [1-4] aids in evaluating user interface, development of usable software, and pinpointing software usability defects. It is developed using the principle that the usability is an inverse function of effort. The model is used for comparison of different implementations of the same application. The results of several experiments conducted on the effort-based model show strong relationship between effort and usability [1-4].

The underlying theory of the effort-based Model is used to produce a framework to identify usability deficiencies in the software. Identifying and locating software usability issues and correlating these issues with UI software code is referred to as Pinpoint Analysis [3,4]. For example, users who are in a state of confusion, and users that are not sure how to use the software, tend to look around the screen to figure out the best way to accomplish a task. This behavior is referred to as an excessive effort [3,4]. Identifying and pinpointing excessive effort behavior helps UI designers rectify numerous usability related issues.

This research attempts to evaluate the utility of pinpointing user interface deficiencies using pattern recognition techniques for identifying excessive effort in segments of software interaction session records. Segmentation of user's software interaction session is done using the time slice between two consecutive mouse/keyboard clicks. Automatic identification of segments showing excessive effort behavior helps the UI designers to reduce the time required for analysis and rearranging the interface at the pinpointed time snapshot.

The pattern recognition methods used in this work include feature selection, principal component analysis, K-means clustering, and threshold based classification [5-7]. Several experiments were conducted to evaluate the new framework for pinpointing software usability issues. Experiment results show more than 40% reduction in time for usability testing.

The rest of this paper is organized as follows. Section II contains background information. Section III summarizes the related work. Section IV details the experimental setup. Section V details the experiments. Section VI presents experiment results and Section VII contains results evaluation. Section VIII concludes the paper with a summary of our findings and proposals for further research.

II. BACKGROUND

A. Software Usability

According to the International Organization for Standardization/International Electro Technical Commission (ISO/IEC) 9126 standard, software usability is: "The capability of a software product to be understood, learned, used, and be attractive to the user when used under specified conditions." There are several characteristics that play an important part in defining software usability: understandability, learnability, operability, and attractiveness [8,9].

Cognitive modeling involves creating a computational model to estimate how long it takes the users to perform a given task [10-13]. It involves one or more evaluators inspecting a user interface by going through a set of tasks by which understandability and ease of learning are evaluated. The user interface is often presented in the form of a paper mock-up or a working prototype; but, it might be a fully developed interface. Cognitive models are based on

 ISBN: 978-1-61208-283-7

psychological principles and experimental studies to determine times for cognitive processing and motor movements. They are used to improve user interfaces or predict problem areas during the design process.

B. The Effort-based Usability Model

Several studies indicate that many system users associate the "physical" effort required for accomplishing tasks with the usability of the software [1-4]. The effort-based model for software usability stems from the notion that the usability is an inverse function of effort. For example, an eye tracking device is used to measure the effort expanded by the user in navigating through the user interface of software. In the case of interactive computer tasks, it is possible to calculate effort as a linear combination or a weighted sum of metrics such as the number of mouse clicks, number of keyboard clicks, eye path traversed as well as other eye activity measures, and mouse path traversed.

Eye trackers acquire eye position data and enable classifying the data into several eye movement types useful for eye related effort assessment [11,12]. The main types of eye movements are: 1) *fixation* – eye movement that keeps an eye gaze stable with regard to a stationary target providing visual pictures with high acuity, 2) *saccade* – rapid eye movement from one fixation point to another, and 3) *pursui***t** – stabilizes the retina with regard to a moving object of interest [1,11,12]. Usually, the Human Visual System (HVS) does not exhibit pursuits when dynamically moving targets are not a part of the interface [1,11,12].

In this research, the following metrics are used as a measure of the *physical* effort 1) Average fixation duration, 2) Average saccade amplitude, 3) Number of saccades, and 4) Average eye path traversed [1-4,11,12].

The effort-based software usability evaluation is divided into three phases: *Measurement, Analysis, and Assessment* [3,4]. In the *measurement phase*, a group of users executes a set of identical independent tasks, which emerge from a single scenario. These tasks differ in key parameters, which prevent the users from memorizing a sequence of interaction activities. Throughout the interaction process, certain user actions such as eye movement, time on task, keyboard activities, and mouse activities are logged.

The *analysis* phase involves accumulating data for several metrics such as the number of saccades, average saccade amplitude, number of fixations, average fixation duration, and average eye path traversed, that relate to user effort. Another metric is the time on task. The average task completion time is compared to a learning curve, which reflects users' mastery of software.

The final step is the *assessment*. Using the above steps, the learnability of software systems is assessed and the point of users' mastery of the software is identified. The same model is applied to obtain operability and understandability of various software systems or different groups of users using the same system. The effort-based metrics provides interface designers with means to evaluate their designs [1].

C. Pinpoint Analysis

Software usability testing is considered one of the most expensive, tedious, and least rewarding tests to implement [1,2]. This perception is likely to change if the usability testing is made less expensive and more rewarding. This requires accurate means through which an engineer can identify and pinpoint issues in the software or the interface. This process is called pinpoint analysis. Pinpoint analysis is one of two types; inter-pinpoint analysis deals with identifying issues with tasks performed by the users in a specific system, whereas intra-pinpoint analysis refers to identifying issues within tasks in a specific system. For example, outlier tasks might be identified through inter-pinpoint analysis and used for intra-pinpoint analysis. This analysis also helps graphical user interface (GUI) designers to make decisions about element placement on displays and determine the level of effort that is related to different widgets [3,4].

1) Inter-pinpoint Analysis

Inter-pinpoint analysis involves detecting tasks that present anomalies and identifying the reasons for these anomalies at a high level. The mouse is used as an example to understand inter-pinpoint analysis. In a particular task, the right mouse button helps users complete a task effectively; however, some of the users are unaware of it. It is possible that anomalies like this can be identified in inter-pinpoint analysis [3,4].

Inter-pinpoint analysis helps identifying alternative methods to perform a task effectively with less effort; however, it does not provide users with a hint of the alternative method. Other issues like the necessity of help facilities in software can be identified by the high level analysis of tasks that present anomalies.

2) Intra-pinpoint Analysis

A more detailed method for analyzing tasks and identifying specific issues with the software is intra-pinpoint analysis. Intra-pinpoint analysis can be done manually by watching all the video recordings of the users' interactions with software, obtained from an eye tracking device. This review helps identifying interaction issues and areas where the user has difficulty while performing tasks. For example, the analysis might reveal that most of the users go into a state of confusion in a specific part of a task and are looking around the screen to identify the best way to proceed with the task. This might prompt the designers to rearrange the interface at the relevant time snapshot. Clearly, this option is tedious and potentially expensive. An alternative is to use a semi-automatic method applying pattern recognition technique. This method eliminates the need for a person to watch the entire video in order to identify interaction issues thereby cutting down the cost and time. It enables automatic identification of areas where the user has difficulty and

 ISBN: 978-1-61208-283-7

marking these areas for further evaluation. For this reason, we refer to the process as semi-automatic.

D. Pattern Recognition

One of the applications of pattern recognition is the assignment of *label*s to a given input value, or *instance*, according to a specific algorithm. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of classes. Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value. *Supervised learning* assumes that a set of training data (the training set), consisting of a set of instances that have been properly labeled by hand with the correct output, has been provided. Next, a learning procedure generates a model that attempts to meet two sometimes conflicting objectives: Perform as well as possible on the training data, and generalize as well as possible to new data. On the other hand, *unsupervised learning* assumes the availability of training data that has not been hand-labeled and attempts to find inherent patterns that are used to determine the correct classification value for new data instances [5-7].

III. LITERATURE REVIEW

Usability is a highly researched topic with much literature available [8-15]. Nevertheless, extensive review did not reveal any research papers related to pinpointing usability issues. There are some papers on effort-based usability evaluation that are discussed below.

Tamir et al. [2] conclude that effort and usability are related but they did not address pinpointing issues. Mueller et al. [16] use effort metrics to evaluate software usability. Their method allows comparison of two or more implementations of the same application, but does not identify where exactly the problem lies. Hvannberg et al. [17] describe the design and test of a defect classification scheme that extracts information from usability problems, but is limited since it does not define the causes underlying usability problems. Nakamichi et al. [18] investigate the relations between quantitative data, viewing behavior of users, and web usability evaluation by subjects. They conclude that the moving speed of the gazing points is effective in detecting low usability. Makoto et al. [19] use a Web-Tracer to evaluate web usability. Web-Tracer is an integrated environment for web usability testing that collects the operation log of users on the Web pages. The data collected is used to determine the usability of the Web pages. However, the reasons for low usability are not identified using this approach. This paper thoroughly addresses and resolves all of the issues listed above.

IV. EXPERIMENTAL SETUP

A. Manual Input Devices

The subject performs the tasks on a computer using a standard keyboard and a mouse as input devices. An event driven logging program is used to obtain details of mouse and keystroke activities from the operating system event queue. The program saves each event along with a time stamp into a file. The logged events are: mickeys (mouse pixels), keystrokes, mouse button clicks, mouse wheel rolling, and mouse wheel clicks.

The eye tracker used for the experiments is Tobii X120 Eye Tracker, version 2.2.5 [20]. The Tobii device is a standalone eye tracking unit designed for eye tracking studies. It measures unfiltered and spontaneous human reactions, responses along with gaze and other real-time data. The data collected by the eye tracker is logged to a file, which is referred to as a *data file* in this paper. The eye tracker also records video version of the user interaction session and is referred to as a video file, which is very helpful in verifying experiment results.

B. Software Environment for Analysis

A software program developed in MATLAB is used to perform data analysis of the experiments performed in this paper [21]. The parameters of the program are detailed in the respective sections.

C. Test Procedure

Experiments conducted to evaluate the capability of pattern recognition techniques to identify software usability issues are done using the steps depicted in Fig. 1.

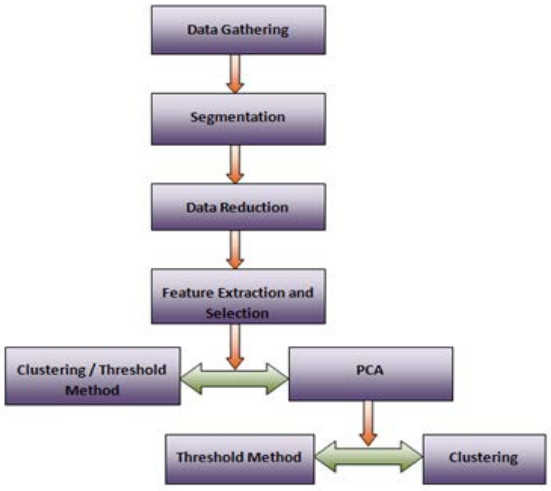

Figure 1. Experiment Procedure.

As the figure shows, the main steps are: data gathering, segmentation, data reduction, feature extraction and selection. These actions are followed by several different classification techniques. The sequence of actions depicted in the figure is further described in three subsections: data gathering, data reduction and identification of excessive effort segments.

 ISBN: 978-1-61208-283-7

1) Data Gathering

A group of five users executes a set of seven identical independent tasks, which emerge from a single scenario. Throughout the interaction process, certain user activities such as eye movement, time on task, keyboard, and mouse activities are logged using an eye tracking device. According to the learnability-based usability model, the point at which the user's effort reaches the acceptable level is called the learning point. Based on this model, it is assumed that the users' effort reaches the acceptable level by the time they perform task 5. Hence, in this paper, task 5 of each subject is used for conducting experiments.

2) Data Reduction

Phase 2 includes activities such as segmentation, data reduction, and feature extraction. The data logged throughout the user interaction session, i.e., the data file is used for event based segmentation where the events are consecutive keyboard/mouse clicks. Metrics such as: (a) segment duration (for event based segmentation), (b) the average fixation duration, (c) the average saccade amplitude, (d) the number of fixations, (e) the number of saccades , (f) the standard deviation of the fixation duration, (g) the standard deviation of the saccade amplitude, and (h) the eye path traversed are inferred for each segment. These metrics are used to generate a feature set, which is obtained by applying data reduction programs to the data file. The features data is calculated for all features within each segment and this data is useful to identify excessive effort segments.

3) Identification of Excessive Effort Segments

Pattern recognition techniques are applied to the feature set obtained from the data reduction process to identify segments that exhibit excessive effort. The techniques used and applied on the feature set are briefly explained below.

Thresholding - a threshold value is calculated for each feature in the feature set. For a given feature, all the segments that have a feature value that is less than the threshold value are classified as non-excessive segments and vice-versa.

K-means clustering - the segments are grouped into clusters. Based on the value of cluster centers, the cluster is classified as excessive or non-excessive. All segments that fall in the excessive cluster are segments exhibiting excessive effort behavior and vice versa.

Principle component analysis (PCA) - the first, the second, and the third principal components are obtained for the feature data. The threshold classification is applied on the first principal component and K-means clustering is applied on the first, second, and third components to classify the segments into excessive or non-excessive.

By the end of phase 3, the excessive effort segments are identified by the software program. To verify the results, the video file is carefully watched segment by segment and classified into excessive or non-excessive segments manually. The manual classification process of the video file is described in the following section.

D. Manual Classification

The manual classification process involves event based segmentation on the entire video file. Each segment is carefully watched and classified into the following categories:

Idle behavior segments; idle behavior is due to system response. Waiting for a progress bar to complete; or waiting for a page to load are examples of idle behavior. Segments with such behavior are classified as idle behavior segments.

Excessive effort segments; segments without any useful user actions are classified as excessive effort segments. A subject looking at different components on an interface instead of the actual target component, which help in accomplishing the task is an example for excessive effort behavior. Such behavior can be eliminated without sacrificing task completion quality.

Non-Excessive effort segments; segments with useful action that result in task completion are classified as non-excessive segments.

Off screen behavior segments: Intervals of time where the subject's view is not within the screen for more than one second, with no meaningful user action, are classified as off screen behavior segments.

Attention segments; segments with frequent off screen behavior, frequent mouse/keyboard clicks are classified as attention segments.

Once the video file is classified into one of the above five segment categories, the manual classification results are ready for comparison with the automatic classification results.

E. Result Verification

The number of Excessive vs. Excessive, Excessive vs. Non-Excessive, Non-Excessive vs. Excessive and Non-Excessive vs. Non-Excessive segments as well as related error rates are calculated for each result file and graphs are plotted to visualize the results and enable comparing the performance of different methods and features. During the verification of results, the attention segments are not considered as they are not clearly distinguished as excessive effort or non-excessive effort. Non-Excessive vs. Excessive segments are regarded as false positive or type-I error segments. It is assumed that all the segments classified as *excessive effort segments* are due for manual evaluation. Hence, in the case of type-I error, the software program is highlighting extra segments for further review, but is not missing any segments that need attention.

On a similar note, segments that show excessive effort per manual classification but identified as non-excessive effort segments by the software program are regarded as false negative or type-II error segments. These segments require extra attention as the software program has missed identified segments that require manual inspection. The total time of segments classified as excessive by the software program is also referred as inspection time. It is the sum of the time interval of each excessive effort segment. In this

 ISBN: 978-1-61208-283-7

paper, type-II errors and inspection time are considered as the most important factors for analyzing experiment results.

V. EXPERIMENTS

In this paper, the automatic part of the process is used to analyze five data files by applying the different pattern recognition techniques discussed. Each of the files is a text file that contains the entire data collected by the eye tracker throughout each experiment. The following is a list of the experiments performed:

1. Applying the threshold method
2. Applying heuristic feature selection and K-means clustering
3. Using principal component analysis
4. Applying K-means clustering on principal components.

Each experiment procedure is discussed in detail in the following sections.

A. Experiment 1: Applying the threshold method

In this experiment, event based segmentation is applied to the video and data file generated by the eye tracker. Next, a feature set is generated for the data file. All the segments are classified into excessive or non-excessive effort segments by the software program, which applies the threshold method on the following features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed.

B. Experiment 2: Applying heuristic feature selection and K-means clustering

Due to the fact that evaluating all the possible subsets of the feature set is prohibitively time consuming, we have adopted a heuristic feature selection method. The following subsets are selected: 1) Number of fixations, 2) Number of saccades, 3) Eye path traversed, 4) Number of fixations, number of saccades, eye path traversed, and 5) Number of fixations, number of saccades, eye path traversed, average fixation duration and average saccade amplitude.

C. Experiment 3: Using principal component analysis

In this experiment, the feature set is transformed into principal components by a program that implements PCA. Here, only the first principal component is considered, as it carries the most significant information related to the feature set. The first principal component is subjected to the threshold method for identifying segments exhibiting excessive effort and non-excessive effort.

D. Experiment 4: Applying K-means clustering on principal components

In this experiment, K-means clustering is applied to different combinations of principal components for identifying segments exhibiting excessive effort and non-excessive effort. The following constitute the feature set for this experiment: 1) 1st principal component, 2) 1st and 2nd principal components, and 3) 1st, 2nd, and 3rd principal components.

VI. RESULTS

In this section, the results obtained from the experiments are discussed. The results of each data file in the experiments are shown in [4]. A sample of these results, concentrating on applying the threshold method using data file 1, is presented here. For clarity, the notation used for the feature values in the graphs is presented below: 1) # Fix – denotes number of fixations, 2) Avg. Fix Dur. – denotes average fixation duration, 3) # Sacc – denotes number of saccades, 4) Sacc Amp. – denotes average saccade amplitude, 5) Eye Path denotes eye path traversed, and 6) FPC - denotes first principal component:

The video file corresponding to data file 1 is 6.09 minutes in length. Fig. 2 shows the results of an experiment using the threshold method on data file 1.

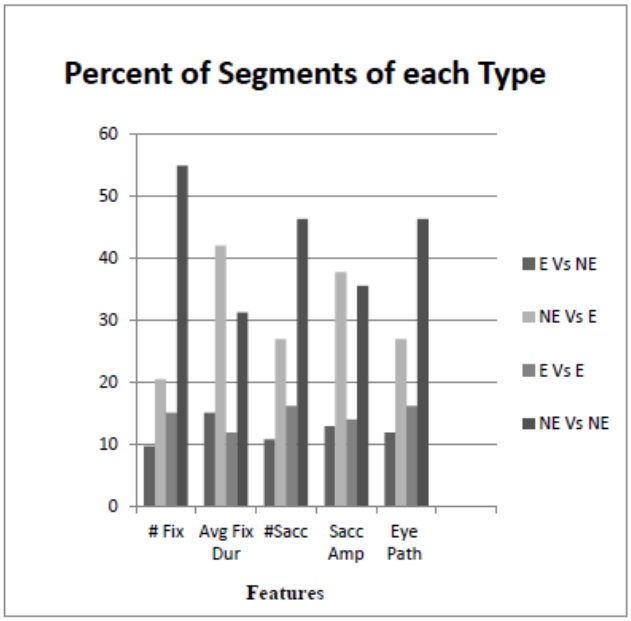

Figure 2. Percent of segments of each type.

When the graph in Fig. 2 is extrapolated and as seen from the E vs. NE bars, the feature value, number of fixations demonstrate a small percentage of E vs. NE segments. This shows that the number of fixations has the least number of type-II errors. Number of saccades and eye path traversed follow number of fixations in terms of type-II errors.

Fig. 3 shows the total time of segments classified as excessive by the software program and the manual process after the threshold method is applied on each of the following features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed.

 ISBN: 978-1-61208-283-7

The light black bars in Fig. 3 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the dark black bars, shows 1.71 minutes of excessive effort. The average fixation duration and average saccade amplitude show a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the bright bars present in the figure. From Fig. 3, it is observed that the percentage of type-II errors is 15.05% for average fixation duration and 12.9% for average saccade amplitude. However, the feature value with a reasonable type-II errors and lower percentage of time of segments classified as excessive is average saccade amplitude.

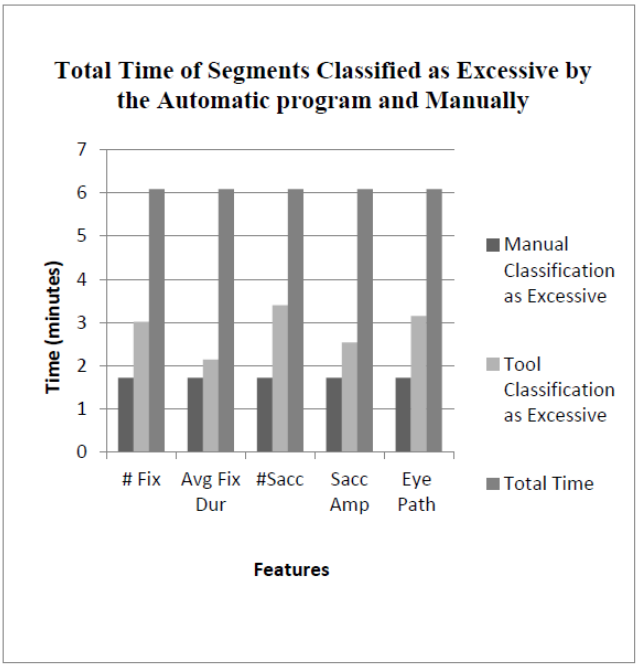

Figure 3. Total time of excessive effort segments.

The set of experiments include three more experiments the results of these experiments are detailed in [4]. The experiments are:

- Identifying excessive effort segments using heuristic feature selection and K-means clustering.
- Identifying excessive effort segments using principal component analysis
- Identifying excessive effort segments using K-means clustering on principal components.

VII. RESULT EVALUATION

In this section, we evaluate and discuss the results of the experiments conducted in this work. Our criteria for success are based on 1) The number of type-II errors and 2) A minimal time to investigate the usability issues with an acceptable level of type-II errors. Based on discussions with several engineers in the company sponsoring this work and other companies, we are assuming that 15% of error of type-II is the upper bound for being considered as acceptable. This is also consistent with a two-step approach where after a first pinpoint analysis stage, which allows for high rate of errors but provides significant reduction in evaluation time, the errors identified are fixed; leading to a more rigorous pinpoint analysis with lower error bound. The results are evaluated based on the performance of each pattern recognition method on individual features. In addition, the overall performance of each pattern recognition method is evaluated.

Tables I to IV (attached at the end of this paper) summarize the results of the experiments. An additional set of tables, which contains the entire results, can be found in [4]. The items listed in Tables I to IV are:

A. *Applying the threshold method*

The following observations are derived from Table I.

- The results of Table I show that the threshold method on the feature value, *number of fixations*, gives good results in terms of type-II errors but, the average inspection time is relatively high when compared to other feature values. The average value of type-II errors for number of fixations is 3.3%. Average saccade amplitude and eye path traversed follow the number of fixations in terms of type-II errors.
- A threshold on *average fixation duration* performs well in terms of minimal inspection time with an acceptable value of 9.8% for type-II errors.
- A feature value with minimum number of total errors is *eye path traversed*. This feature value is a good choice when inspection time is not taken into account.
- The inspection time is not completely correlated to type-I errors. In the case of average fixation duration, the inspection time is 1.67 minutes with 29.4% of type-I errors. On the other hand, the average saccade amplitude with almost the same percentage of type-I errors has higher inspection time than average fixation duration.
- The values of the average number of excessive effort segments for all features are in close proximity to each other. However, the percentage of type-I and type-II errors differs invariably. This portrays that the segments classified as excessive are different for each feature value.
- Despite the fact that the percentages of total errors for each feature value are in close proximity to each other, the inspection time varies. This delineates that the segments classified as excessive are different for each feature value.

B. *Applying heuristic feature selection and K-means clustering.*

The following observations are derived from Table II:

- The results from Table II show that the K-means clustering on the *feature subset - number of fixations,*

number of saccades, eye path traversed, average fixation duration, and average saccade amplitude, gives good results in terms of type-II errors with an average value of 5.4%. But, the average inspection time is relatively high when compared to other feature values. The number of fixations follows the above identified feature value in terms of type-II errors.

- Clustering on the *eye path traversed* performs well in terms of minimal inspection time with an acceptable value of 10.1% for type-II errors.
- A feature value with minimum number of total errors is the number of fixations. This feature value is a good choice when inspection time is not taken into account.
- The average number of excessive effort segments for number of fixations and the feature subset with the following features - number of saccades, eye path traversed, average fixation duration, average saccade amplitude are the same. However, the inspection times vary. This portrays that the segments classified as excessive are different for each feature value.
- Unlike the results of the threshold method, the percentages of total errors for each feature value vary by a wide margin when applying the K-means clustering on the feature subsets.

C. Using principal component analysis

The results summarized in Table III are compared with the results obtained from Experiment 1 to analyze the performance of the threshold method on the first principal component with other features such as: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed. Experiment 1 result evaluation shows that the feature value, number of fixations, gives good results in terms of type-II errors. The average percentage of type-II errors for number of fixations is 3.3%, whereas it is 4.1% for first principal component. Initially, average saccade amplitude and eye path traversed succeeded number of fixations in terms of performance. However, the new results place a threshold on the *first principal component* after the number of fixations with respect to type-II errors. The inspection times for the first principal component and for the average fixation duration are 2.7 and 1.6 minutes respectively. A threshold on *average fixation duration* performs better than first principal component in terms of lower inspection time and an acceptable 9.8% for type-II errors.

D. Applying K-means clustering on principal components.

Table IV shows the average values of all the features used in experiment 4 over five data files. The average type-II error is very high when using the K-means on the principal components. The average inspection time is only 1.96%. When taking type-II errors also into consideration, this method is not suitable to identify excessive effort segments.

VIII. CONCLUSIONS AND FUTURE RESEARCH

The framework presented in this research enables software developers to efficiently identify the usability issues thereby optimizing the time spent on software-usability testing. Excessive effort segments, which typically relate to the usability issues, are identified by applying pattern recognition techniques such as K-means clustering algorithm, thresholding, principal component analysis, and feature selection. The analysis of the experiments conducted in this paper shows that the time taken for software usability testing can be reduced by 40% or more.

Of all the pattern recognition methods used, a threshold on number of fixations yields the best results in terms of type-II errors and is followed by a threshold on the first principal component. The K-means clustering on feature subset with the features: number of fixations, number of saccades, average saccade amplitude, average fixation duration, and eye path traversed ranks third.

When the inspection time is taken into consideration while also confirming that type-II errors are within a reasonable limit, the K-means clustering on the number of saccades yields the best results and precedes the threshold method on average fixation duration in performance.

With time and resources at one's disposal, there is a scope to enhance the definition and implementation of pattern recognition techniques in identifying usability issues in software.

In this research, the time between two consecutive keyboard/mouse clicks by user is considered as a segment and this has served as the basic pattern for pattern recognition techniques. Equal time slicing of user's software interaction session can be used instead and the performance results can be analyzed and compared with the results from this research.

Further refinement of pattern recognition techniques can be pursued to minimize errors and inspection time. Also, more focus can be given to the criteria for manual classification of video segments thus allowing excessive effort segments to be identified more accurately.

Another direction for future research is to automate some of the manual steps in this process. This can include software that automatically log users' software interaction session data, manipulate data, and without human intervention lists the start and end times of all the excessive effort segments. This can significantly reduce time taken for the usability testing.

In this work, we have concentrated on pattern recognition techniques that do not rely on human intelligence. Hence, results are generated using non-supervised learning procedures. A surrogate approach can use supervised learning procedures to produce the output. This involves conducting experiments using training data sets to manually arrive at an archetype that can be applied on any data set to generate the output.

Another direction for further research is to consider information fusion. Information fusion combines different

 ISBN: 978-1-61208-283-7

techniques of pattern recognition classification to achieve more accurate results. An additional approach that can be researched is to arrive at a formula or function that can be applied to any data set by which the usability deficiencies can be identified.

ACKNOWLEDGMENT

This research was funded in part by Emerson Process Management [22], an Emerson business.

REFERENCES

[1] O. C. Komogortsev, C. J. Mueller, D. E. Tamir, and L. Fledman, "An Effort Based Model of Software Usability," 2009 International Conference on Software Engineering Theory and Practice, FL, 2009, pp. 75-84.

[2] D. E. Tamir, O. V. Komogortsev, and C. J. Mueller. "An Effort and Time Based Measure of Usability," 6th Workshop on Software Quality, 30th International Conference on Software Engineering, Leipzig, Germany, 2008, pp. 35-41

[3] D. E. Tamir, et al. "Detection of Software Usability Deficiencies," International Conference on Human Computer Interaction, FL, 2011, pp. 528-536.

[4] D. K. V. Dasari, Pinpoint analysis of software usability, Thesis Report, Texas State University, Computer Science, December 2012. https://digital.library.txstate.edu/handle/10877/32/browse?value=Dasari+Kali+Venkata%2C+Divya&type=author, retrieved June 2013.

[5] J. T. Tou and R. C. Gonzalez, Pattern Recognition Principles, Reading, MA: Addison-Wesley Publishing, Inc., 1974.

[6] R. O. Duda, P. E. Hart, and D. G. Stock, Pattern Classification, 2nd Ed., Indianapolis, Willey International, 2001.

[7] D. E. Tamir and A. Kandel, "The Pyramid Fuzzy C-means Algorithm," International Journal of Computational Intelligence in Control, 2 (2), 2012 pp. 65-77.

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

[9] ISO/IEC 9126-1: 2001, Software Engineering-Product Quality, Part-2, External Metrics, Geneva, Switzerland: International Standards Organization, 2001.

[10] A. Poole and L. J. Ball, Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future Prospects, Encyclopedia of Human Computer Interaction: Idea Group, 2004.

[11] M. A. Just and P. A. Carpenter, "Eye Fixation and Cognitive Processes," Cognitive Psychology, vol. 8, 1976, pp. 441-480.

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

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

[14] H. Ebbinghaus, Memory: A Contribution to Experimental Psychology, 1885, http://psychclassics.yorku.ca/Ebbinghaus/memory3.htm, retrieved July 2013.

[15] J. Nielsen, Usability Engineering, San Francisco, Boston, Academic Press, 1993.

[16] C. J. Mueller, D. E. Tamir, O. C. Komogortsev, and L. Feldman, "Using Designer's Effort for User Interface Evaluation," *IEEE International Conference on Systems, Man, and Cybernetics*, Texas, USA, October 11, 2009, pp. 480-485.

[17] E. T. Hvannberg and C. L. Lai, "Classification of Usability Problems (CUP) Scheme," Nordic conference on Human-computer Interaction, Oslo, Norway, 2006, pp. 655-662.

[18] N. Nakamichi, S. Makoto, and S. Kazuyuki, "Detecting Low Usability Web Pages using Quantitative Data of Users' Behavior," Proceedings of the 28th international conference on Software engineering, New York, NY, 2006, pp. 569-576.

[19] S. Makoto, N. Noboru, H. Jian, S. Kazuyuki, and N. Nakamichi. "Webtracer: A New Integrated Environment for Web Usability Testing," 10th Int'l Conference on Human - Computer Interaction. Crete, Greece, June 2003, pp. 289-290.

[20] Anonymous, "Tobii X60 & X120 Eye Trackers: User Manual," Tobii, 2013, http://www.tobii.com/Global/Analysis/Downloads/User_Manuals_and_Guides/Tobii_X60_X120_UserManual.pdf , retrieved June 2013.

[21] Anonymous, "MATLAB Product Help," MATLAB, 2013, http://www.mathworks.com/help/, retrieved June 2013.

[22] Anonymous, "Emerson Process Management," Emerson, 2013, http://www.emersonprocess.com, retrieved June 2013.

 ISBN: 978-1-61208-283-7

TABLE I. AVERAGE VALUES OF EXPERIMENT 1 RESULTS

Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type- I errors	avg. % type- II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
# Fix	17.2	95	28.4	3.3	31.7	2.7	62.1
Avg. Fix Dur.	18.2	95	29.5	9.9	39.4	1.6	37.4
#Sacc	32	95	21.8	10.5	32.2	2.9	64.1
Sacc Amp.	17.6	95	29.1	4.6	33.7	2.5	56.4
Eye Path	17.8	95	25.7	5.1	30.8	2.6	57.7

TABLE II . AVERAGE VALUES OF EXPERIMENT 2 RESULTS

Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type -I errors	avg. % type -II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
#fix	29.1	95	27.2	6.6	33.9	2.4	56.2
#sacc	23.5	95	17.8	8.9	26.7	2.0	45.1
eye path	19.7	95	18.0	10.1	28.1	1.6	37.5
#fix, #sacc, eye path	23.2	95	18.3	8.6	26.9	1.9	44.5
#fix, #sacc, eye path, avg. fix dur., avg. sacc amp.	29.2	95	32.6	5.4	38.0	2.5	56.3

TABLE III. AVERAGE VALUES OF EXPERIMENT 3 RESULTS

Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type -I errors	avg. % type- II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
1st principal components	16.6	95	27.5	4.1	31.6	2.7	61.2

TABLE IV. AVERAGE VALUES OF EXPERIMENT 4 RESULTS

Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type- I errors	avg. % type- II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
1st, 2nd & 3rd principal components	28.6	95	24.4	12.6	37.0	2.0	43.6

.

 ISBN: 978-1-61208-283-7