GUIEvaluator: A Metric-tool for Evaluating the Complexity of Graphical User Interfaces

Khalid Alemerien and Kenneth Magel
Computer Science Department
North Dakota State University
Fargo, North Dakota, USA
khalid.alemerien@my.ndsu.edu, kenneth.magel@ndsu.edu

Abstract—User interface design metrics assist developers to evaluate interface designs in early phase before delivering the software to end users. This paper presents a metric-based tool called GUIEvaluator for evaluating the complexity of the user interface based on its structure. We defined a metrics model that includes five modified structural measures of interface complexity: Alignment, grouping, size, density, and balance. The results of our tool are discussed in comparison with the subjective evaluations of interface layouts. This paper demonstrates the potential benefits of incorporating automated complexity metrics into the user interface design process. Our findings show that the means of the interface complexity values for both the subjective evaluation and the GUIEvaluator are equal at a significance level 0.01.

Keywords—User Interface Design, GUI, Complexity Metrics, Screen Aesthetics, Usability, Automated Tools.

I. INTRODUCTION

Despite the abundance of usability and interface design principles and guidelines, interface complexity continues to be a pressing Human Computer Interaction (HCI) issue. With the tremendous advances in technology, the user interfaces become complex. Therefore, interface design is a difficult process that needs effective tools and techniques to evaluate interface layouts. One of these techniques is the software metrics. With the rapid advance of software, software metrics have become one of the foundations of software management and essential to the success of software development. Nowadays, software metrics are integrated into programming languages that provide developers statistical data about their code. One of the metrics suites is structural metrics. Of the structural metrics, particularly important are complexity metrics, which are used widely with code. Thus, complexity metrics have played a significant role in software development. Therefore, the growth of interface complexity calls for the growth of the complexity metrics in the interface design process. Although people realize the importance of interface metrics, the complexity metrics field still needs to grow quickly to meet the requirement of interface design. This encourages us to investigate the importance of applying complexity metrics in interface design.

We need to study and merge the complexity metrics into the interface design environment for two reasons: First, there are several applications of the complexity measures in the software development process such as project size estimation, cost and time estimation, effort estimation, and software maintenance. In order to improve the user interface quality and the project controllability, it is necessary to control the interface complexity by measuring the related aspects. Second, user interface is a key component of any software application. Moreover, good interfaces make the interaction between the human and the software seamless and as effortless as possible. Previous empirical studies have proven that the layout complexity is important to aesthetic perception of user interfaces [11, 12].

Quantitative tools can help developers identify better solutions to user interface design problems and make better decisions when faced with alternative designs [14]. In order to enhance the user satisfaction through well-designed interfaces in early stages, we present five design factors with their metrics based on the structural aspects of the interface layout. Those design factors are: Alignment, grouping, size, density, and balance, which are considered significant influences on interface usability [8]. This metrics suite is supported by a complementary tool called GUIEvaluator, which provides automatic metrics calculation. This tool is used to measure the complexity of an interface layout in order to judge the quality of the interface design.

The ultimate goal of this research is to develop an effective metric-based tool to evaluate the complexity of interface layouts. This quantitative tool may mitigate the cost and time of the evaluation of user interfaces in early stages. The rest of this paper is organized as follows. Section II describes the GUIEvaluator tool and the metrics model in further detail. Section III presents the related work. Section IV shows the collected data and procedure. Section V shows the results of our user study. Section VI discusses the results. Section VII concludes the paper.
II. RELATED WORK

Several research studies run in the interface design field focus on developing and calculating metrics models manually or semi-automatically. Stickel et al. [2] introduced a method to calculate the visual complexity, which is based on the number of possible interactions or functional elements, number of higher level of structures or organizational elements, and summed entropy of RGB values. Sears [3] developed a layout metric called Layout Appropriateness (LA). Each task needs a sequence of actions and LA metric attempts to calculate the costs of each sequence of actions. Kang and Seong [5] proposed three measures to determine the complexity of UI design: Operation, transition, and screen complexity. In addition, Xing [7] developed metrics which utilize three factors associated with complexity: numeric size, variety of elements, and relation between elements.

Ngo et al. [11, 12] introduced a new aesthetic model which consists of 14 aesthetic measures for screen layouts. Fongling et al. [9] developed a metrics model to evaluate the complexity of web pages, including four design factors: density, alignment, grouping, and size. In addition, Parush et al. [10] developed a numerical model to evaluate the GUI. This model consists of four screen factors: element size, grouping, alignment, and local density. All of the above metrics were calculated manually.

A number of researchers developed semi-automated tools that help them to calculate some aspects of user interface designs. Comber and Maltby [1] developed an application called Launcher. Zheng et al. [4] developed a Matlab tool to calculate image vectors. Sears [6] introduced a metric-based tool called AIDE, which partially automates the designing and evaluating user interface layouts. Miyoshi and Murata [8] developed a numerical tool for evaluating the usability of a screen design based on its complexity measures. González et al. [13] developed a tool called BGLayout which helps to automate the metrics calculations using image processing techniques.

GUIDeveloper extends the ideas presented in previous work in two directions: First, GUIDeveloper is a full automated tool that calculates and analyzes the five structural interface measures. This helps designers to determine which design factor has a high complexity rating and allows designers to evaluate, redesign, and reevaluate their interface designs from within a single interface design environment. The previous tools are semi-automated tools. Furthermore, these tools calculate four design factors. Second, GUIDeveloper incorporates the weight for each design factor using the user preferences. This helps us to consider the human factors as a part of our metrics.

The previous studies do not take into account the importance of design factors.

III. GUIDeveloper TOOL AND METRICS MODEL

A. The GUIDeveloper Tool

GUIDeveloper is a tool for supporting the effectiveness evaluation of our metrics model. GUIDeveloper calculates the complexity of the interface layout. By using this tool we can evaluate user interface layouts whether the interface is under development or a part of running applications. GUIDeveloper was developed in Visual Basic 2012. This tool extracts the interface layout information using reflection techniques that are supported by Visual Basic 2012. The extraction and analysis window of GUIDeveloper is shown in Fig. 1.

![Figure 1. The extraction and analysis window of the GUIDeveloper](image)

B. Metrics Model

The purpose of the metrics-model is to reduce the time and costs of subjective evaluation of user interfaces early in the software development. This model consists of five metrics for measuring five design factors: Alignment, grouping, density, balance, and size. The values of these metrics will be used to calculate the interface design complexity. These metrics are the following:

1. Alignment

Alignment metric measures the vertical and horizontal alignment of objects in two levels: A group level (Local Alignment) and screen level (Global Alignment). These two alignment levels are combined to calculate the Total Screen Alignment Complexity.

1.1 Local Alignment

Equation 1 calculates the alignment for each group. Where Vp is the number of vertical alignment points and Hp is the number of horizontal alignment points. K is the number of grouped objects on the
screen. The range of values of GA is [0, 1]. Equation 2 calculates the Alignment Complexity (AC) for all groups on the screen. Where the Weight is the number of objects in a group (i) divided by the total number of grouped objects. And m is the number of groups on the screen.

\[ \text{Group Alignment (GA)} = \sum_{i=1}^{K} \frac{Vp+Hp}{2K} \]  
(1)

\[ AC = \sum_{i=1}^{m} GAi \times Weight(i) \]  
(2)

1.2 Global Alignment

The global alignment is calculated as shown in Eq. 3, where Vp is the number of the vertical alignment points and Hp is the number of the horizontal alignment points. N is the number of ungrouped objects on the screen. The range of values of the SA is [0, 1].

\[ \text{Screen Alignment (SA)} = \sum_{i=1}^{N} \frac{Vp+Hp}{2N} \]  
(3)

1.3 Total Screen Alignment Complexity

Equation 4 shows the Total Alignment Complexity (TAC). Where weight1 is the ratio of the number of grouped-objects to the total number of objects on the screen. Whilst the weight2 is the ratio of the number of ungrouped-objects to the total number of objects on the screen.

\[ TAC = AC \times weight1 + SA \times weight2 \]  
(4)

2. Balance

Balance metric uses the number and size of objects from Eq. 5 and Eq. 6, respectively, for each quarter of screen to calculate the Total Balance Complexity (TBC). The BQni and BQnj represent the number of objects in ith and jth quarters. The range of ratio of BQni and BQnj is [0, 1] and the range of values of BQn overall is [0, 1], where 0 means unbalanced and 1 means fully balanced in terms of number of objects. The BQsi and BQsj represent the sum of sizes of objects in ith and jth quarters. The range of ratio of BQsi and BQsj is [0, 1] and the range of values of BQs overall is [0, 1], where 0 means unbalanced, 1 means fully balanced in terms of object size. Equation 7 is the Total Balance Complexity (TBC).

\[ BQn = \frac{\sum_{k=1}^{6} BQni}{6} \]  
(5)

\[ BQs = \frac{\sum_{k=1}^{6} BQsi}{6} \]  
(6)

\[ TBC = 1 - (0.5 \times BQn + 0.5 \times BQs) \]  
(7)

3. Density

Density metric measures the screen occupation by objects, where Eq. 8 calculates the Local Density (LDj) for the group jth and Eq. 9 calculates the Global Density (GD). In Eq.10, the Density-Complexity (DC) is calculated taking into account the W1 that is the ratio of the area of groups to the screen area, and W2 is the ratio of ungrouped area to the screen area.

\[ LDj = \frac{\sum_{i=1}^{ungroupedobjects} \text{Size of object } i \text{ in a group } j}{\text{Area of group } j} \]  
(8)

\[ GD = \frac{\sum_{k=1}^{ungroupedobjects} \text{Size of object } k}{\text{Area of ungrouped}} \]  
(9)

\[ DC = \frac{\sum_{j=1}^{LDj}}{n} \times W1 + GD \times W2 \]  
(10)

4. Size

Size metric measures the object sizes complexity of two levels: The object size complexity (SCk) as in Eq. 11 and the size complexity overall (SC) as in Eq. 12, where N is the number of objects in type kth and Sj is the number of different sizes, where Sj is 1 if the object size is not counted before and 0 if the object size is counted. Wi is the number of objects types and Weight (j) is the number of objects in category jth divided by the total number of objects on the screen.

\[ \text{Object Size Complexity (SCk)} = \frac{\sum_{j=1}^{N} Sj}{N} \]  
(11)

\[ SC = \sum_{k=1}^{Wi} SCk \times Weight(k) \times Wi \]  
(12)

5. Grouping

Grouping metric measures the number of objects that have a clear boundary by line, background, color, space, or size. Equation 13 calculates the percentage of ungrouped objects (UG), where the GW is the object that is grouped. The value of GW equals 1, if the object exists in a group, otherwise GW equals 0. N is the total number of objects on the screen. Equation 14 calculates the ratio of the number of different object types (G) to the total number of objects (M) in all groups, where the Weight is the ratio of total number of grouped objects to total number of objects on the screen. To calculate the grouping complexity (GT), we use Eq. 15.

\[ UG = 1 - \frac{\sum_{k=1}^{N} GW}{N} \]  
(13)
\[ GC = \frac{G}{M} \cdot \text{Weight} \quad (14) \]
\[ GT = UG + GC \quad (15) \]

6. Overall Screen Layout_Complexity (LC)

Equation 16 calculates the Overall Screen Layout_Complexity (LC), which includes the values of TAC, TBC, DC, SC, and GT metrics. Each metric has a weight \((w1, w2, w3, w4,\) and \(w5\), respectively), which are calculated based on the participant rating. The values of those weights are 0.84, 0.76, 0.80, 0.72, and 0.88, respectively.

\[
LC = \left( TAC \cdot w1 + TBC \cdot w2 + DC \cdot w3 + SC \cdot w4 + GT \cdot w5 \right) / 5 \times 100\% 
\quad (16)
\]

IV. METHODOLOGY

A. Participants

The study was conducted at North Dakota State University. Participants included 50 student volunteers (17 females, 33 males), where 80% of participants were graduate students and 20% were undergraduate students. The participants who are computer science or related fields were 50% and 50% were from other majors. The data that were collected about the participants show 46% of participants have no experience in software development. But 24%, 16%, and 14% of participants have one or two years, three to six years, and six years and above, respectively.

B. Data Collection

This section presents a brief description of data collected during the study. We developed an application to collect our data. The study took approximately 35 minutes. First, we showed the participants examples that explain the design factors and how to rate them. Second, we asked participants to provide background information. Third, the participants were asked to rate the five design factors (alignment, grouping, density, balance, and size) and the user interface design overall using a 7-point Likert scale. Finally, to obtain the weight for each of the five complexity measures in Eq.16, the participants rated the importance of each measure.

V. RESULTS AND ANALYSIS

The major objective of this study is to investigate the usefulness and effectiveness of using the GUIEvaluator and its metrics in evaluating the user interface complexity. Therefore, our investigation addresses the following two research questions:

**RQ1**: Is the GUIEvaluator effective to measure the complexity of user interface?

**RQ2**: How do the structural measures of an interface affect the interface complexity rating?

Our hypotheses associated with RQ1 are:

- **H1**: given a specific user interface, the means of interface complexity for the user rating and the GUIEvaluator are equal.
- **H2**: given a specific user interface, there is a strong positive correlation between the user rating and the GUIEvaluator in terms of interface complexity value.

To test these two hypotheses, we performed the t-test on (H1) and the Pearson correlation on test (H2) for both the user rating and the GUIEvaluator, at a significance level of 0.01, on 18 interface layouts. Table 1 shows that the means of the user rating and the GUIEvaluator are 0.561 and 0.575, respectively. Furthermore, to test the hypothesis (H1), we performed the t-test for (H1) as shown in Table 1, df=32.39, for a significance level of 0.01. It shows there is no difference between the means of interface complexity of the user rating and the GUIEvaluator. In addition, to test the hypothesis (H2), we performed the Pearson correlation test. In Table 1, the R value (0.804) shows a strong positive correlation between the user rating and the GUIEvaluator at a significance level of 0.01.

Figure 2 presents the complexity values of 18 interface layouts, which are rated by both the participants and the GUIEvaluator. We have observed strong similarities between the complexity values for both the user rating and the GUIEvaluator. Therefore, our tool can be used to accurately evaluate the complexity of user interfaces.

Our hypothesis associated with RQ2 is:

- **H3**: given a specific user interface, the values of the metrics which exist in our metrics-model are strongly correlated with interface complexity values given by both the users and the GUIEvaluator.

To test this hypothesis, we performed the Pearson correlation test and the t-test for both the user rating and the GUIEvaluator with the five complexity measures, at a significance level of 0.01, on 18 screen layouts. Table 2 shows the results of the Pearson correlation test and t-test for the values of the five complexity measures and the values of interface complexity given by both the user rating and the GUIEvaluator. On the one hand, Table 2 shows that there is a strong correlation among the user rating and the GUIEvaluator and the following design factors at a significance level of 0.01: Size, alignment, density, and balance. On the other hand, the grouping factor has a weak correlation with the user rating and the GUIEvaluator with R values 0.462 and 0.095, respectively. Therefore, we can accept the hypothesis.
As just outlined, the interface complexity values, which are rated by both the participants and the GUIEvaluator, are consistent. Furthermore, our findings show that our complexity metrics of interface layouts are useful, but these metrics do not have equal magnitude of importance. We investigated that the grouping measure has the less correlation with complexity values for both the user rating and the GUIEvaluator. Perhaps the reasons behind this are: (1) Misunderstanding the objects grouping on the screens, (2) 46% of participants have no experience in software development, or (3) our grouping metric may be insufficient to measure the objects grouping.

In Table 1, our findings show that the R value is 0.804 between the user rating and the GUIEvaluator for 18 interface layouts. However, the interface layouts 3, 8, and 14, we observed that there is a non-trivial difference between the interface complexity values for both the user rating and the GUIEvaluator. The complexity measure that causes this difference is the objects grouping given by user rating. The values of grouping are less than the average of the values of other complexity measures for the interface layouts 3, 8, and 14. This encourages us to focus on grouping measure to investigate the causes behind the variance of the values between the user rating and the GUIEvaluator. In summary, whether we use GUIEvaluator or user rating to evaluate the interface complexity, we reach the same conclusion. Therefore, our findings confirm the effectiveness of our tool and its metrics-model to be used during the early stage software development.

**VI. DISCUSSION**

As just outlined, the interface complexity values, which are rated by both the participants and the GUIEvaluator, are consistent. Furthermore, our findings show that our complexity metrics of interface layouts are useful, but these metrics do not have equal magnitude of importance. We investigated that the grouping measure has the less correlation with complexity values for both the user rating and the GUIEvaluator. Perhaps the reasons behind this are: (1) Misunderstanding the objects grouping on the screens, (2) 46% of participants have no experience in software development, or (3) our grouping metric may be insufficient to measure the objects grouping.

In Table 1, our findings show that the R value is 0.804 between the user rating and the GUIEvaluator for 18 interface layouts. However, the interface layouts 3, 8, and 14, we observed that there is a non-trivial difference between the interface complexity values for both the user rating and the GUIEvaluator. The complexity measure that causes this difference is the objects grouping given by user rating. The values of grouping are less than the average of the values of other complexity measures for the interface layouts 3, 8, and 14. This encourages us to focus on grouping measure to investigate the causes behind the variance of the values between the user rating and the GUIEvaluator. In summary, whether we use GUIEvaluator or user rating to evaluate the interface complexity, we reach the same conclusion. Therefore, our findings confirm the effectiveness of our tool and its metrics-model to be used during the early stage software development.

**VII. CONCLUSION**

In this paper, we identify a set of structural measures of user interfaces. This metrics-model consists of five metrics: Alignment, grouping, size, density, and balance. This model is supported by a complementary tool called GUIEvaluator, which automatically calculates the values for the metrics. Our findings confirm the effectiveness of our metrics model and GUIEvaluator to measure the complexity of interface layouts. Some issues still need investigating. Our future work will include conducting an experiment to compare our metrics with other existing metrics and extend our model with new structural metrics. Therefore, this paper is the start point of developing our framework that consists of tools and metrics to automate the evaluation of user interfaces in early stages. To sum up, this work addresses the lack of effective tools and techniques to evaluate the complexity of user interfaces during software development. So, automated tools can play a significant role in evaluating of the complexity of user interfaces.
Table 2. Pearson correlation test and t-test results for each user interface design factor with the user rating and GUIEvaluator rating

<table>
<thead>
<tr>
<th>Interface Design Factor</th>
<th>User Rating</th>
<th></th>
<th></th>
<th></th>
<th>GUIEvaluator Rating</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pearson Correlation Test</td>
<td>T-test</td>
<td></td>
<td></td>
<td>Pearson Correlation Test</td>
<td>T-test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alignment</td>
<td>0.836</td>
<td>&lt;0.00001</td>
<td>-0.780</td>
<td>0.442</td>
<td>0.662</td>
<td>0.0028</td>
<td>-1.005</td>
<td>0.323</td>
</tr>
<tr>
<td>Balance</td>
<td>0.825</td>
<td>&lt;0.00001</td>
<td>-0.560</td>
<td>0.579</td>
<td>0.705</td>
<td>0.0011</td>
<td>-0.857</td>
<td>0.398</td>
</tr>
<tr>
<td>Density</td>
<td>0.683</td>
<td>0.0018</td>
<td>-0.772</td>
<td>0.445</td>
<td>0.585</td>
<td>0.0098</td>
<td>-1.050</td>
<td>0.302</td>
</tr>
<tr>
<td>Size</td>
<td>0.943</td>
<td>&lt;0.00001</td>
<td>-0.731</td>
<td>0.470</td>
<td>0.767</td>
<td>&lt;0.001</td>
<td>-1.013</td>
<td>0.319</td>
</tr>
<tr>
<td>Grouping</td>
<td>0.462</td>
<td>0.0537</td>
<td>-2.608</td>
<td>0.014</td>
<td>0.095</td>
<td>0.708</td>
<td>-2.657</td>
<td>0.012</td>
</tr>
</tbody>
</table>

Figure 3. Comparison of the values of five complexity measures with the interface complexity values of the user rating

Figure 4. Comparison of the values of five complexity measures with the interface complexity values of the GUIEvaluator rating

REFERENCES