Metrics for Event Driven Software

Neha Chaudhary
Ph.D. Scholar, Gautam Buddha University,
Greater Noida, India

O.P. Sangwan
Guru Jambheshwer University of Science & Technology,
Hisar, India

***Abstract*—The evaluation of Graphical User Interface has significant role to improve its quality. Very few metrics exists for the evaluation of Graphical User Interface. The purpose of metrics is to obtain better measurements in terms of risk management, reliability forecast, project scheduling, and cost repression. In this paper structural complexity metrics is proposed for the evaluation of Graphical User Interface. Structural complexity of Graphical User Interface is considered as an indicator of complexity. The goal of identifying structural complexity is to measure the GUI testability. In this testability evaluation the process of measuring the complexity of the user interface from testing perspective is proposed. For the GUI evaluation and calculating structural complexity an assessment process is designed which is based on types of events. A fuzzy model is developed to evaluate the structural complexity of GUI. This model takes five types of events as input and return structural complexity of GUI as output. Further a relationship is established between structural complexity and testability of event driven software. Proposed model is evaluated with four different applications. It is evident from the results that higher the complexities lower the testability of application.**

Keywords—Graphical User Interface; Structural Complexity; Testability; Fuzzy model

I. Introduction

Graphical User interfaces have special characteristics that differentiate them from the rest of the software code. These applications have many challenges due to its event driven nature and infinite input domain. This event driven nature presents a challenge to testing because there are a large number of possible event sequences that users can invoke through a user interface. It is important to assess the quality of software. Software testability is one of the quality metric for software applications and ISO has defined software testability as a functionality and it defines functionality as “the collection of characteristics of software that bear on the effort required to authenticate the software produced”.

The testability of software is determined by factors such as:

- Controllability
- Observability
- Built in Test Capability
- Understandability
- Complexity

Complexity is one of the important factors to assess the testability of software. It is concluded from the literature survey that typical software metrics for complexity dose not identify complex GUI. Various structural complexity metrics are reported in literature survey. Most of the papers defined structural complexity in terms of visual objects, size, distribution and position and tree structure they are as follows:

1) *Number of controls in an interface [1]*

2) *The longest sequence of different controls that is defined to perform a specific task. In terms of the GUI XML tree, it is depth of the tree.*

3) *Maximum number of choices for a user at any moment while using that interface.*

4) *Time required performing certain events in a GUI.*

5) *Tree structure metric that defines complex GUI when majority of its controls is toward the top and less complex GUI when most of the control is at bottom [2].*

6) *Tree path count calculates the number of leaf nodes in a tree [3].*

7) *Tree depth is calculated by the total number of choices in the tree is divided by the tree depth [3].*

Users interact with a GUI by performing events on some widgets, like a button click, menu open, and dragging an icon. All interaction of GUI and a user is with the help of these events [4][6]. As defined in above metrics all choices presented to the user will be in the form of events. Maximum tree depth will be longest sequences of events. Therefore a new metric can be defined in terms of all types of events a GUI is consisting. It was shown from the Alsmadi I. (2011) research that LOC metric is irrelevant for GUI because it cannot be identified that how much code is GUI oriented. GUI events are classified on the basis of their response to the system on selection. Their classification is as follows [11].:-

Restricted-focus events,

Restricted-focus events open a modal window of GUI. For example, “Set Language” is a restricted focus event.

Unrestricted-focus events

Unrestricted-focus events open modeless windows. For example, Replace in MS WordPad is an unrestricted focus event.

Termination events

Termination events are used to close modal windows. Examples of Termination events include Ok, Exit and Cancel. There are other types of events that GUI contains, and they do not open or close windows but make other GUI events available. Menus that contain several events are open by these events.

Menu-open

Menus can be open by using Menu-open events. They expand the menus so that the set of GUI events would be available to the user. Menu-open events need not interact with the underlying software. Unrestricted-focus events open a window that has to be explicitly terminated, whereas menu open events has no such restriction. The most common example of menu-open events is generated by menus that open pull-down menus. Common example includes File, Edit, and Format. All other remaining events in the GUI are used to interact with the underlying software [8].

System-interaction

System-interaction events interact with the underlying software to perform some action; common examples include the Copy event used for copying objects to the clipboard [13]. This classification of events is used to compute structural complexity of GUI based software. To compute the structural complexity an assessment process is defined. An overview of Assessment Process is presented in the next section.

II. Structural Complexity Assessment Process

Based on types of events an assessment process for structural complexity is designed. This process is outlined next:

1) Input to the process is GUI application. The Application under Test (AUT) is identified which essentially means identification of the locations of source files and any library modules needed to compile/build the AUT. This is known as baseline AUT.

2) To extract GUI components GUITAR (GUI Testing framework) is used, which generates. GUI Structure (using a module called the GUI ripper) by automatically traversing all the windows of the GUI.

3) Event calculator computes count of each type of events. It is a kind of parser which takes GUI structure as input and return count of type of event.

4) Each event is assigned a weight value.

5) Normalization of data: Softmax scaling method is used to map all values between 0 and 1. Softmax scaling is based on the logistic function given in equ. (1):

$$y = 1 / (1 + e^{-x}) \qquad (1.1)$$

Where y is the normalized value and x is the original value.

The logistic function transforms the original range of [-∞,∞] to [0,1] and also has a linear part on the transform. The values of the variables must be modified before using the logistic function in order to get a desired response.

This is achieved by using the following transform

$$x' = (x - \underline{x})/(\lambda(\sigma /2\pi)) \qquad (1.2)$$

where $\underline{x}$ is the mean of x , σ is the standard deviation, and λ is the size of the desired linear response.

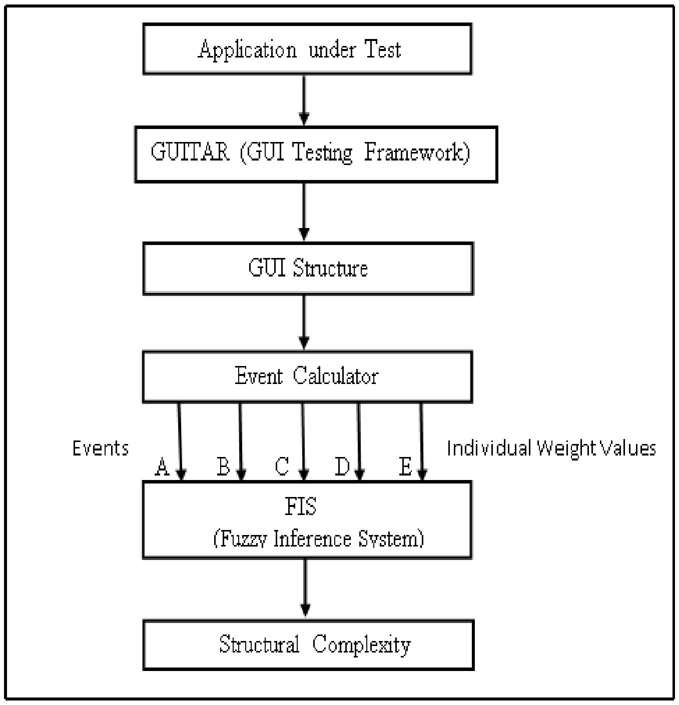


Fig. 1. Structural Complexity Assessment Process

Once the number of events is calculated their weight value is multiplied to each count.

6) Fuzzy Inference System (FIS) is the process of formulating the mapping from an input space to output space [13]. A fuzzy model is proposed with five inputs, namely Unrestricted Focus Events, Menu Open Events, Termination Events, System Interaction Events and Restricted Focus Events. Figure1.2 shows the fuzzy model. The proposed model consists of five inputs and provides a crisp value of structural complexity using Rule Base. In this model Mamdani's fuzzy inference method is used as shown in figure 3.

After the fuzzification process, there is a fuzzy set for each output variable that needs defuzzification. The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is singleton number. Further centroid method is used for defuzzification.

Events are classified in five types and further they are divided into three states (linguistic variables) i.e. low, medium and high.

The input variable Unrestricted Focus Events has been divided into three levels i.e. Low, medium and high.

Similarly other four inputs are divided into three states i.e. low, medium and high. The output variable complexity is also is classified as low, medium and high.

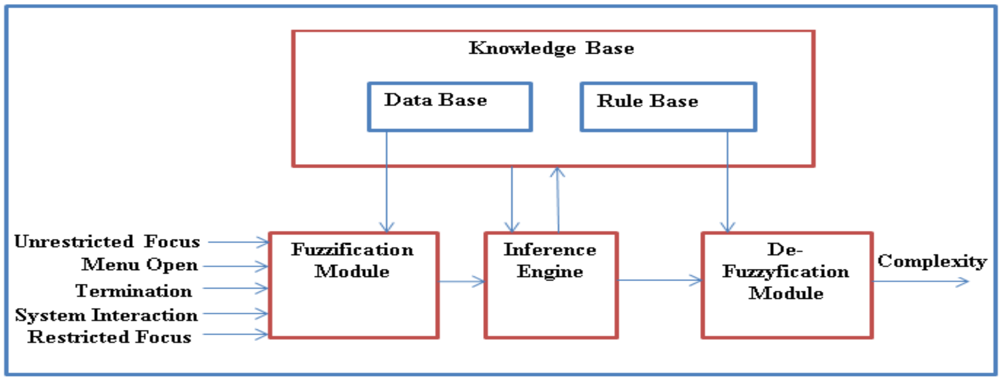


Fig. 2. Fuzzy model for Structural Complexity Assessment

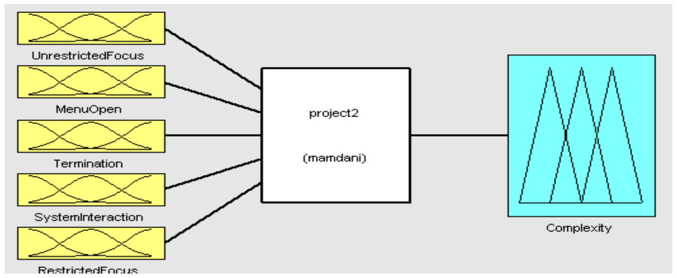


Fig. 3. Fuzzy Inference System: Complexity Assessment Model

A. Rule base and evaluation process

When input data is fuzzified, processing is carried out in fuzzy domain. The model integrates the effects of multiple factors Unrestricted Focus Events, Menu Open Events, Termination Events, System Interaction Events and Restricted Focus Events into a single measurable parameter that will define the structural complexity of test case, based on the following knowledge/rule base. The rule base can further be advanced by creating more ranges (fuzzy sets) for the input variables. All inputs and outputs are fuzzified as shown in figure 1.4. All possible combinations of inputs were considered that will create 3^5 i.e. 243 sets. The structural complexity for all 243 combinations is classified as low, medium and high by expert judgment. This indicates to formulation of 243 rules for the fuzzy model and some of the rules are presented below:

1) If value assigned for Unrestricted Focus Events is high, Menu Open Events is high, Termination Events is high, System Interaction Events is high and Restricted Focus Events is high then structural complexity is high.

2) If value assigned for Unrestricted Focus Events is high, Menu Open Events is high, Termination Events is high, System Interaction Events is high and Restricted Focus Events is medium then complexity is high.

3) If value assigned for Unrestricted Focus Events is high, Menu Open Events is high, Termination Events is high, System Interaction Events is high and Restricted Focus Events is low then complexity is high.

.

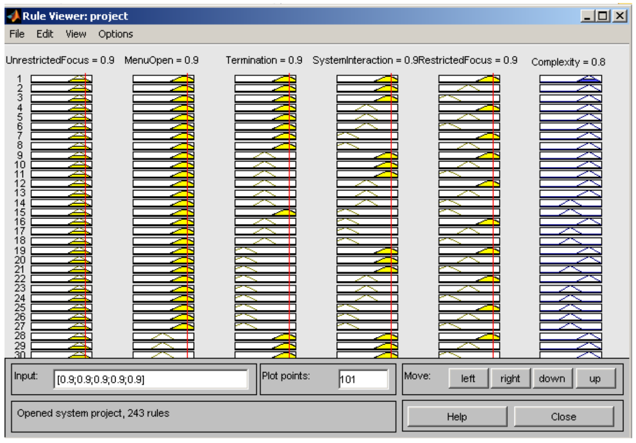
.

.

? If value assigned for Unrestricted Focus Events high, Menu Open Events is medium, Termination Events is high, System Interaction Events is high and Restricted Focus Events is high then complexity is high.

…..

243. If value assigned for Unrestricted Focus Events very low, Menu Open Events is low, Termination Events is very low, System Interaction Events is low and Restricted Focus Events is low then complexity is low.



Opened system project, 243 rules

Fig. 4. Rule Viewer for the Complexity Assessment Model

All 243 rules are inserted and rule base is created. A rule is fired based on the particular set of inputs. In this model Mamdani style inference has been used.

The output of GUI complexity has been observed using rule viewer for particular set of inputs using MATLAB fuzzy Tool Box as shown in figure 4.

Defuzzification

After getting the fuzzified output as specified in previous section, these values are defuzzified to get the crisp value of the output variable priority. Transformation of the output from fuzzy domain to crisp domain is called defuzzification. In this model defuzzification is done using centre of gravity (COG) method. The final crisp value for COG is 0.8. Final value of complexity assessment for given input is 0.8, which lies in a complex GUI category.

For example if following values are considered as inputs to the model: Unrestricted Focus Events =0.9, Menu Open Events=0.9, Termination Events=0.9, System Interaction Events=0.9 and Restricted Focus Events=0.9.

III. Evaluation of Fuzzy Based Complexity Assessment Method

For the evaluation of fuzzy based test case prioritization method software artifacts are taken from Event Driven Software Lab (Dept. of Computer Science, University of Maryland, USA) established in 2005. These applications are part of an opensource office suite which has has been considered in many research. This application suite is known as TerpOffice3 and includes TerpWord (a small word-processor), TerpSpreadSheet (a spreadsheet application), TerpPaint (an image editing/ manipulation program) and TerpPresent (a presentation tool). These applications are implemented using Java. These applications are fairly large with complex GUIs nearly as the size of MS WordPad. These applications do not have any complex underlying "business logic." This property of applications makes them perfect subject applications for GUI research.

TABLE I. Events and Normalized Values for TerpPaint 3.0

Event type	TerpPaint 3.0	Weighted Value	Value Transformation	Logistic Normalization
System Interaction	589	58.9	6.0607	0.9977
Termination	0	0	-8.8630	0.0001
Non Restricted Focus	1	10	-6.3293	0.0018
Restricted Focus	11	55	5.0725	0.9938
Menu Open	51	51	4.0590	0.9830
Mean	-	34.98	-	-
Standard deviation	-	24.80809545	-	-

For all these application value of termination event is zero. So this value is not considered. Count of all events is coming in different range so different weight value is assigned to make these values come in same range. Softmax scaling is used to map all values in the range of 0 to 1. For all applications these values are calculated and they are shown in the table 1 to 4 for all applications. Table 1 shows events and normalized values for TerpPaint 3.0.

Table 2 shows event and normalized value for TerpPaint 3.0. For this application number of menu open events is very high as compared to other applications.

Table 3 represents Events and Normalized values for Terp SpreadSheet 1.0. In this application total numbers of events are small as compared to other application.

TABLE II. Events and Normalized Values for TerpPresent 3.0

Event type	Terp Present 3.0	Weighted Value	Value Transformation	Logistic Normalization
SYSTEM INTERACTION	682	68.2	3.1606	0.9593
TERMINATION	0	0	-6.7453	0.0012
NON RESTRICTED FOCUS	0	0	-6.7453	0.0012
RESTRICTED FOCUS	10	50	0.5171	0.6265
MENU OPEN	114	114	9.8130	0.9999
Mean	-	46.44	-	-
Standard deviation	-	43.27556354	-	-

TABLE III. Events and Normalized Values for TerpSpreadSheet 3.0

Event type	TerpSpread Sheet 3.0	Weighted Value	Value Transformation	Logistic Normalization
SYSTEM INTERACTION	401	40.1	5.2760	0.9949
TERMINATION	0	0	-7.4079	0.0006
NON RESTRICTED FOCUS	0	0	-7.4079	0.0006
RESTRICTED FOCUS	6	30	2.0813	0.8891
MENU OPEN	47	47	7.4585	0.9994
Mean	-	23.42	-	-
Standard deviation	-	19.87223188	-	-

Table 4 shows events and normalized values for TerpWord 3.0. this application is fairly large application in terms of number of events as compared to other applications.

TABLE IV. Events and Normalized Values for TerpWord 3.0

Event type	TerpWord 3.0	Weighted Value	Value Transformation	Logistic Normalization
SYSTEM INTERACTION	1628	162.8	9.7407	0.9999
TERMINATION	0	0	-5.6195	0.0036
NON RESTRICTED FOCUS	1	10	-4.6760	0.0092
RESTRICTED FOCUS	23	115	5.2308	0.9947
MENU OPEN	10	10	-4.6760	0.0092
Mean	-	59.56	-	-
Standard deviation	-	66.62112578	-	-

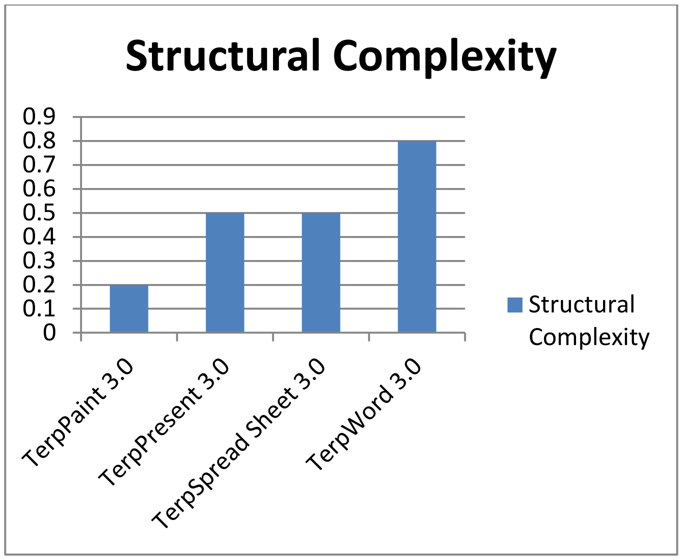


Fig. 5. Structural complexities of Applications under test

All normalized values shown in table 1 to table 4 are given as input to fuzzy model and their structural complexity is calculated and shown in figure 5.

Further fault seeding has been done on all applications and 200 faults are injected in each applications and different version of applications are created and tested with GUITAR[5].

Number of faults identified by each application is shown in figure 6.

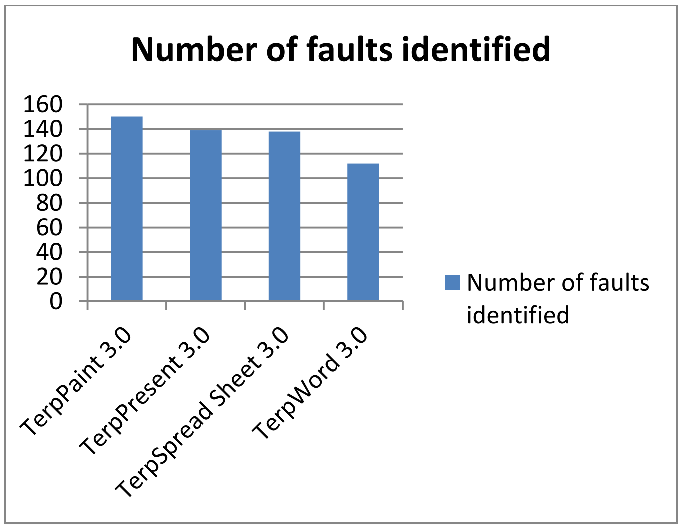


Fig. 6. Number of faults identified by each application

IV. CONCLUSION

Number of identified faults represents teastability of software, where TerpPaint shows highest Testability and its structural complexity is lowest. Testability of TerpWord is lowest and its complexity is highest. Structural complexity of TerpPresent and TerpSpreadsheet is same and their fault revealing capability is almost same. Hence it is evedent from the results that higher the complexity lower the testability of application.

For evaluation of GUI structural complexity metric is defined which is based on types of events. A fuzzy model is developed for GUI evaluation. Its results are evaluated with 4 different applications. A relationship is further established with testability with this metric.

ACKNOWLEDGMENT

For the evaluation software artifacts are taken from Event Driven Software Lab (Dept. of Computer Science, University of Maryland, USA). This repository is a huge help for the researchers in the field.

REFERENCES

[1] Magel, K. and Izzat, A. , "GUI Structural Metrics and Testability Testing", in Proceedings of IASTED SEA, USA, 2007, pp. 159-163.

[2] Alsmadi Izzat and Al-Kabi Mohammed, "GUI Structural Metrics", Published in the International Arab Journal of Information Technology, Vol. 8, No. 2, 2011, pp. 124-129.

[3] Alsmadi Izzat and Al-Kabi Mohammed, "The Introduction of Several User Interface Structural Metrics to Make Test Automation More Effective", Published in the Open Software Engineering Journal, 2009, pp. 72-77.

[4] Memon Atif, " Automatically Repairing Event Sequence-Based GUI Test Suites for Regression Testing", ACM Transaction on Software Engineering and Method. Volume 18, Issue 2, 2008.

[5] Hackner R., Daniel, Memon Atif, " Test Case Generator for GUITAR", International Conference on Software Engineering, (Washington, DC, USA), 2008

[6] Memon Atif, Soffa Lou Mary, Martha E. Pollack, " Coverage Criteria for GUI Testing" , Proc. of the 8th European Software Engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of Software Engineering, 2001, pp. 256-267.

[7] Memon Atif, McMaster Scott, " Call-Stack Coverage for GUI Test Suite Reduction", IEEE Transaction on Software Engineering, Volume 34, 2008.

[8] Memon Atif, Strecker Jaymie, " Relationships Between Test Suites, Faults, and Fault Detection in GUI Testing" , In ICST'08 Proc. of the First international conference on Software Testing, Verification, and Validation, (Washington, DC, USA),2008.

[9] Memon Atif, Nagarajan A. and Xie Q., "Automating Regression Testing for Evolving GUI Software", Journal of Software Maintenance and Evolution: Research and Practice, Volume 17, no. 1, 2005, pp. 27-64.

[10] Memon Atif , Xie Qing, "Studying the Fault-Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software", IEEE Transaction on Software Engineering, Volume 31, no. 10, 2005, pp. 884-896.

[11] Huang Chin-Yu, Chang Jun-Ru and Chang Yung-Hsin, "Design and analysis of GUI test-case prioritization using weight-based methods", The journal of Systems and Software vol. 83, 2 010, pp. 646-659.

[12] Chaudhary Neha, Sangwan O.P., Singh Yogesh, "Test Case Prioritization Using Fuzzy Logic for GUI based Software", International Journal of Advanced Computer Science and Applications, 2012.

[13] Memon A., Soffa Lou M., "Regression testing of GUIs", In the Proc. of the 9th European software engineering conference New York, NY, USA, 2003, pp. 118-127.