International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol. 8, No. 1 (2015), pp. 225-240
http://dx.doi.org/10.14257/ijsip.2015.8.1.20

GUI Reliability Assessment based on Bayesian Network and Structural Profile

Zhi-Fang Yang, Zhong-Xing Yu, Bei-Bei Yin and Cheng-Gang Bai

School of Automation Science and Electrical Engineering,
Beijing University of Aeronautics and Astronautics
qwyzf@163.com

Abstract

Graphical User Interfaces (GUI) is becoming increasingly important in the software field as it builds a friendly way between users and software through continuous interactions. A well-developed GUI is therefore an important factor of software quality. In particular, the reliability of GUIs is still on the way of development. Existing software reliability assessment techniques attempt to statistically describe the software testing process and predict the reliability of the system. However, those techniques are not suitable for GUI as quality of GUI is challenged by immense number of event interactions and complex structural profile etc. Furthermore, GUI has a wealth of information about GUI architecture, components, windows and their interactions with each other, which can be adopted to guide the testing process and establish confidence assessment of GUI. In this paper, a Bayesian network model of GUI reliability is introduced to discuss the reliability model topology and its issues encountered in the modeling process. A case is also presented to verify the validity of the model during the GUI reliability assessment process.

Keywords: *GUI; reliability model; Bayesian network; structural profile*

1. Introduction

Graphical User Interfaces (GUI) is becoming increasingly important in improving the software usability since it is a friendly way to interact between the users and the software system. Many researchers have already conducted a lot of meaningful work on GUI testing that eventually improves the reliability of GUI. Even that, the reliability of GUIs is still at the beginning stage of its development [1].

Software reliability is defined as the probability of failure-free software operation for a specified period in a specified environment [15]. Several software reliability models have been developed since 1970 [15, 16, 17]. In particular, in testing phase, software reliability models are used to assess reliability through statistically describe the software testing process and predict the reliability of the system. However, it is hard to achieve the predefined objectives due to the high complexity of software[18]. Compared with traditional software, GUI is more complex and thus more difficult to measure. In this regard, the current software reliability models may be totally infeasible to be adopted in GUI.

To overcome the drawbacks above, Belli et al., [1,19] introduced event-based GUI testing and reliability assessment techniques, which proposed an experimental insight and preliminary results, and analyzed the software reliability assessment techniques in event-based GUI testing. Consequently, the reliability of the GUI depends on the approaches and reliability model used for GUI. Thus, it is important to establish the model of GUI according to its characteristic. Compared with traditional software, GUI reliability assessment encounters several problems as follow:

ISSN: 2005-4254 IJSIP

Firstly, generation and execution of GUI test cases is so complicated that test data is incomplete due to missing of some failure data. It requires reliability assessment techniques of GUI to be more flexible and effective.

Secondly, based on the interaction of GUI event and state, the profile of GUI is so complicated that a new structural profile is introduced in the work of B.B. Yin[20]. Considering the state of GUI event, two structural profiles of GUI are defined in section 3. More importantly, how to predict the impact of testing profile on the GUI is an important issue in GUI reliability assessment.

Thirdly, compared with traditional software, the failure of the GUI is difficult to described quantitatively. How to analyze the influence of different failures on GUI reliability assessment is quite complicated.

Finally, the most important problem is that traditional reliability models treat a specific software as a monolithic whole and only consider its interactions with external environment [19]. GUI testing can benefit from information including GUI structure, component and window. These information can be used to guide the testing process and improve the confidence level of GUI reliability assessment.

Bayesian network is proposed to deal with the above problems. It has been used to solve many complex scientific problems [9, 10, 11], especially the considerable uncertainties. In this paper, a Bayesian Network Topology will be built according to the information of GUI structure, test path and structural profile. First, the learning function of Bayesian Network model will be gained a wealth of information based on prior knowledge of the posterior distribution, which can be adopted to guide the testing process or establish confidence assessment of GUI. Second, compared with traditional neural network method, the network structure of the Bayesian approach is more flexible and suitable for GUI testing as GUI intricate interaction of the testing process even some failure data is missing. Third, Bayesian methods show higher sensitivity of a small sample than the other models. This is extremely important for GUI testing since exhaustive test for GUI is proven to be impossible.

The aim of this paper is to provide an model of the GUI reliability assessment based on Bayesian network. The rest of this paper is organized as follow. The background of GUI testing is introduced in Section 2. The GUI structural profile is presented in Section 3. The Bayesian Network of GUI reliability assessment is described in detail in Section 4. The parameter learning and data updating process are described in Section 5. The analysis, construction process and a case study are demonstrated in Section 6. Finally, the conclusion is drawn in Section 7.

2. Ralated Work

2.1 GUI testing

Several different model-based approaches have been proposed for GUI testing. White et al. [2, 3] and Belli [4] pointed out that various responsibilities of a user can be specified as a complete interaction sequence (CIS) between the user and the GUI application. A.T.Memon et al [4,5,6] proposed the event flow graph (EFG) models in the GUI testing and several approaches guiding the GUI testing. In the work of L.Zhao[7], he introduced a new model of event handler based on EFG, which took advantage of relationship of events and code and developed a corresponding testing framework of GUI. Z.F.Yang[8] presented an approach of GUI testing Guided by Bayesian model that can guide the GUI testing and find more defects as soon as possible.

2.2 GUI and Event

A GUI is a graphical user interface for users. The basic input is an event in GUI, which is normally generated by action of a user. This action is sent to GUI by the operating system and GUI responds to the event in order to accomplish a specific function[4].

2.3 EFG

A.T.Memon proposed that Event-Flow Graph (*EFG*) instead of all the events interactions in the GUI, which was defined as follows:

$$<V, E, B> \quad (1)$$

where V is the set of all vertices in the EFG, in which each vertex represents one event; E is the set of edges which are event interactions; B is a set of initial events, which can be immediately executed after GUI starts.

2.4 Event Handler and HIG

When an event is triggered, all possible responses to the event will be named event handler (*EH*). Event handler interaction graph (HIG) is defined as follow [7]:

$$<H, RHI> \quad (2)$$

where H is the set of all vertices in the EFG, in which each vertex is an event handler; RHI is the set of edges which are event handler interactions.

2.5 Test Cases

The test case generation is the most important and difficult task of the GUI testing. Indeed, the sequence of events in the GUI test needs to be used as a test input. In this paper, the test cases are generated by utilize EFG for path searching and HIG interaction as the coverage criteria.

2.6 Main window, modal window and modeless window

A main window is a parent window which is presented after GUI starts [12]. It launches a parent window that stays active on the user's screen until GUI closes. Main windows can be controlled and entered any information and will hide behind modal windows while modal windows are invoked.

A modal window is any window that is a child (secondary window) to a parent window and usurps the parent's control. A user may not press any controls or enter any information on the parent window (the original window which opens the modal window) until the modal has been closed. A modal window is commonly used when the GUI wants to retain the user's focus on the information in the modal as it is impossible for the user to interact with the other windows.

Similar to a modal window, a modeless window is a feature that was first introduced in Internet Explorer 5. It launches a secondary (child) window that stays active on the user's screen until dismissed. Modeless windows can be minimized or hidden behind other windows. Unlike a modal window, a modeless window will allow the user to continue working with the GUI when the modeless window is open.

3. GUI Structural Profile

3.1 A Simple Event Profile of GUI

Let's consider a simple event profile in which the input domain of the GUI is divided into three equivalent classes C_1, C_2, C_3. Various inputs are selected from the input domain and applied to the software system. Each time an input is selected from C_i with probability P_i. Consequently, GUI runs by invoking various modules. Suppose that the software operational profile, which describes the behaviors of software input domain, is given as follow:

$$TP(\lambda)=\{<C_i, P_i>, i=1,2,3\} \quad (3)$$

$$Where \sum_{i=1}^{3} P_i = 1 \quad (4)$$

3.2 Transition Matrix

Considering the state of GUI, all event of GUI have new states once an event is triggered. The changes of states of the GUI are called transitions, and the probabilities associated with various state changes are called transition probabilities. The process is characterized by a state space, a transition matrix describing the probabilities of transitions, and an initial state (or initial distribution) across the state space, just as Markov chain [13].

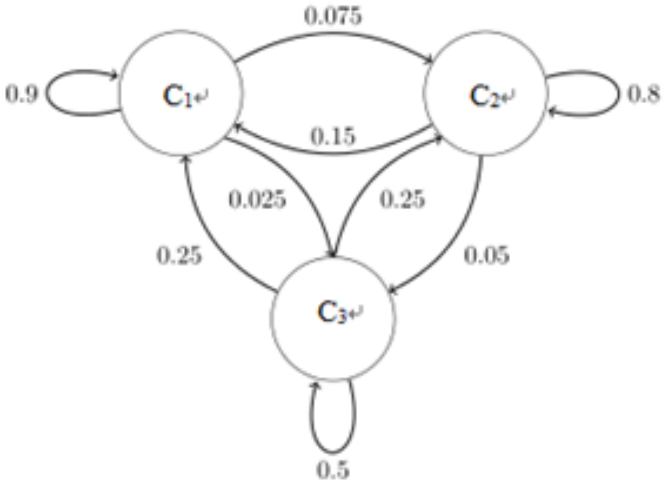


Figure 1. GUI States Changes Diagram for a Simple Example

GUI is in a certain state at a specific time and its state changes randomly within the profile. The GUI states that the conditional probability distribution for the system at the next step depends on the current state of the system. Since the GUI state changes randomly, a state diagram for a simple example is shown in the Fig. 1, using a directed graph to demonstrate the state transitions. The states represent the transition probabilities from the i^{th} state to the $i+1^{th}$ state. Labelling the state space $\{C_1, C_2, C_3\}$, the transition matrix for this example is shown as follow:

$$P_{ij} = \begin{pmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{pmatrix} \quad (5)$$

Where, the sum of each row is 1.

3.3 GUI Structural Profile

Consider the state of GUI and the state transition discussed above. GUI profile, which describes the behavior of GUI input domain, is given as follow:

$$STP(\lambda)=\{<C_i, P_{ij}, M_j>, i=1,2,3\} \quad (6)$$

$$Where \sum_{i=1}^{3} P_i = 1 \quad (7)$$

where P_{ij} means that an input is selected from C_i with probability P_i in the j^{th} step; M_j means that the transition matrix from the j^{th} to the $j+1^{th}$ step.

3.4 Operational Profile

As described in section 2 and section 3, the input domain of the GUI is divided into three equivalent classes as three windows. It is given as follow:

$$OP(\lambda)=\{<W_i, P_{ij}, M_j>, i=1,2,3\} \quad (8)$$

where W_1 means that event handler belonged to main window, W_2 means that event handler belonged to modeless window, W_3 means that event handler belonged to modal window.

P_{ij} means that an input is selected from W_i with probability P_i in the j^{th} state.

M_j means that the transition matrix from the j^{th} state to the $j+1^{th}$ state.

3.5 Functional Profile

Also, the input functions of the GUI are divided into several classes as

$$FP(\lambda)=\{<F_i, P_{ij}, M_j>, i=1,2,3,4,5\} \quad (9)$$

Where F_i means that the i^{th} function of GUI, P_{ij} means that an input is selected from F_i with probability P_i in the j^{th} state, M_j means that the transition matrix from the j^{th} state to the $j+1^{th}$ state.

4. Bayeisan Network Reliability Model of GUI

In the process of GUI reliability assessment, tester or manager of GUI testing are more concerned with the test itself rather than the reliability of GUI. As stated before, the GUI testing can benefit from information based on GUI structure, component and window. However, as a user of GUI, the customers focus on the reliability of specific functions since reliability is a user-oriented measure. It is noted that the actual value of the GUI reliability for one customer cannot be known until the GUI is put into operation for a sufficiently long time. An estimate of the reliability of the GUI can be obtained within a confidence interval. However, different strategies for test case selection usually lead to different test results and thus affect the reliability estimation. This section discusses how to build the GUI reliability assessment model.

4.1 Assumption of Model

(1) The source code of the GUI is frozen during the entire test. In other words, no debugging is performed when a failure is observed.

(2) A test action consists of (a) selecting a test case from the *m*-th equivalent classes, (b) executing the test case, (c) classifying the outcome of execution as success or failure, and (d) updating the GUI reliability estimate, when necessary.

(3) Each test execution leads to one of the two outcomes: success, or failure. The probabilities of each outcome are denoted by θ_i^s and θ_i^f respectively:

θ_i^s = *Pr* {no failure is observed/ a test case from the *i*-th equivalent classes is executed},and

θ_i^f = *Pr* { failure is observed/ a test case from the *i*-th equivalent classes is executed}.

Where $\theta_i^s + \theta_i^f = 1$. (10)

(4) A total of *n* test actions are allowed to test the software for reliability assessment; testing is stopped after *n* test cases are executed.

(5) GUI Structural profile is described in section2d and section2*e*.

Note that assumption 1 indicates that the GUI testing process concerned in this paper is GUI testing for reliability assessment, rather than for reliability growth. Therefore the GUI under assessment is code-frozen.

Then, according to the Nelson's model [14], reliability is defined as follow:

$$R=\sum_{i=1}^{m} P_i\ (1-\theta_i) \quad (11)$$

and the corresponding unreliability is

$$\rho=\sum_{i=1}^{m} P_i\ \theta_i=1-R \quad (12)$$

where p_i is the probability that a test case is selected from the *i*-th equivalent classes, and θ_i is the failure detection rate for the *i*-th equivalent classes.

4.2 A Simple Bayesian Network

In the work of Z.F.Yang [10], a simple Bayesian model of GUI is proposed as shown in Fig.2.

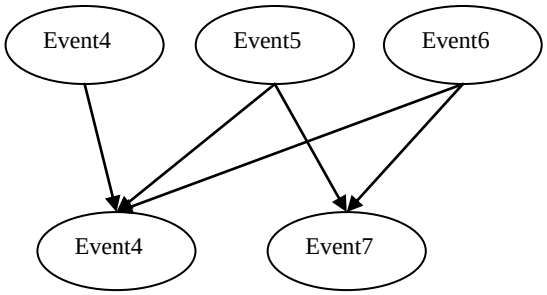


Figure 2. A Simple Bayesian Network Model of GUI

In Fig.2, there are three independent nodes, Event 4, Event 5, and Event6 in the first layer. Also, there are two independent nodes, Event 4 and Event 7 in the second layer. The nodes in the first layer refer to parent nodes and that in the second layer are named child notes. Assumptions are listed as follow:

(1) Each parent nodes has two states: 0 (this event was not be carried out) and 1 (this event was carried out), and the last event, also have two states: 0 (this event was be executed and failed) and 1 (this event was executed correctly).

(2) If one or more parent events (Event 4, Event 5, or Event 6) occur, then the child event will also occur.

(3) The conditional probability of each child node follows the simple cases although it has a few difficulties.

Thus, in this model shown in Fig. 2, one or more of Event 4, Event 5, and Event 6 occur due to the assumption of node dependency. However, if one event occurs, then the state of GUI will change. Thus Event 4, Event 5, and Event 6 cannot occur at the same time, it is contradictory to node dependency.

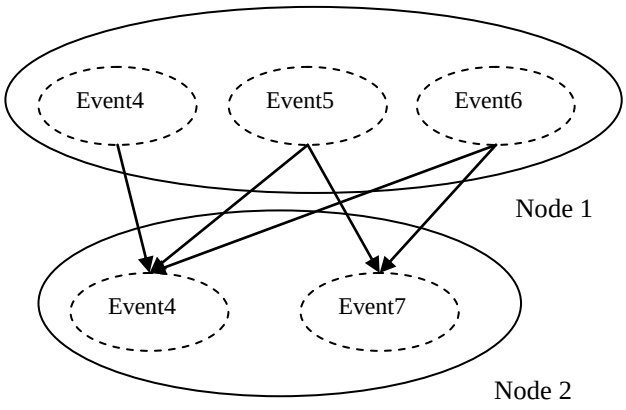


Figure 3. A New Bayesian Network Model of GUI

A new Bayesian model of GUI testing is shown in Fig. 3 with the following assumptions:

Each parent node has *n* states (the *n*-th state means that the *n*-th type event was occur). Thus, as shown in Fig. 3, the node 1 has three states and the node 2 has two states. Considering states of GUI and the trouble of state explosion, in this paper, test cases are divided into three groups according the profile depicted in section2*d*.

4.3 The Topology of Bayesian network

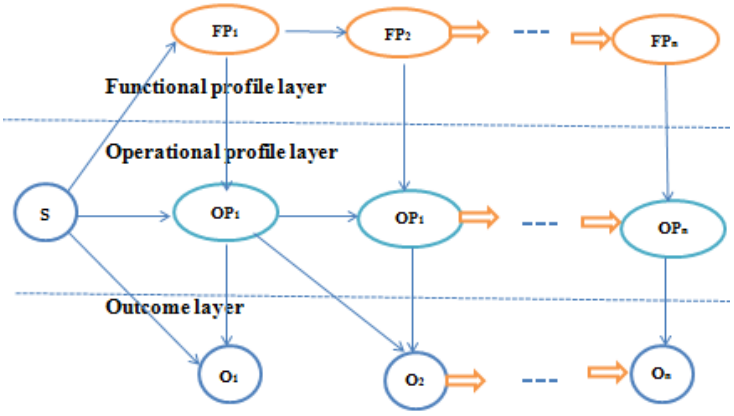


Figure 4. A Complex Bayesian Network Model of GUI

A new Bayesian model of GUI testing reliability is shown in Fig.4.In this graph model, the Bayesian network is divided into three layers: functional profile layer, operational profile layer, and outcome layer.

Functional profile layer is described in section2e. In this layer, each node has *m* states, each of which represents one function of GUI. Each node was influenced by the front node.

The second layer is operational profile layer, which was described in section2d. In this layer, each node has 3 states. Each state represents one type window of GUI. Each node was influenced by the front node and the Layer above.

The third layer is outcome layer, which was described in section4. In this layer, each node has 2 states. Each state represents success or failure. Each node was influenced by the front node and the above layer.

This model is suitable for length of GUI test case is n. Node Start means that GUI is start.

5. Mathematical Analysis of the Bayesian Model

5.1 Parameter Learning and Data Updating Process

A basic Bayesian model contains both topology and parameter analysis. The topology has briefly discussed above. Mathematical analysis of the Bayesian will be mentioned in Fig. 5. Node *S* means that GUI starts and node OP_1 has 3 states. Each state of OP_1 represents that one type event handler is executed, which is described in section 2*e*.

First, an empirical conditional probability distribution is set according to the testers, the test manager or user feedback experience. In order to obtain an objective prior distribution, N_i is defined as the number of event handler belonging to the *i*-th type in initial test pool, where i=1, 2, 3.

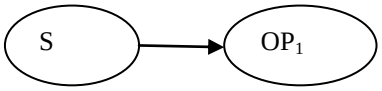


Figure 5. A Basic Bayesian Network Model of GUI

Second, *S* are two-dimensional discrete random variables. $S=0$ means that GUI initial with failure while $S=1$ represents that GUI initial successfully. OP_1 are three-dimensional discrete random variables. $OP_1=i$ means that the *i*-th type event handler is executed in state 1, where i=1, 2, 3.

Third, considering the probability of $\{OP_1 = i\}$ under the condition of $\{S=1\}$, that is to find the probability as Eq. (13):

$$\{OP_1 = i \mid S=1\},\ i=1, 2, 3 \tag{13}$$

According to the conditional probability formula, it can be deduced as Eq. (14):

$$\{OP_1 = i \mid S=1\} = \frac{P\{OP_1 = i \mid S=1\}}{P\{S=1\}} \tag{14}$$

Fourth, the experience of the conditions given in the probability distribution will be according to Eq. (15).

$$P\{OP_1=i \mid S=1\} = \frac{P\{OP_1 = i \mid S=1\}}{P\{S=1\}} = \frac{N_i}{\sum_{j=1}^{3} N_j},\ i, j=1,2,3 \tag{15}$$

In actual testing process, we define $N_i^{\divideontimes}$ is the real number of N_i in actual test procedure, so the updating of the condition probability will be given as Eq. (16):

$$P\{OP_1 = i \mid S=1\} = \frac{P\{OP_1 = i \mid S=1\}}{P\{S=1\}} = \frac{N_i}{\sum_{j=1}^{3} N_j} = \frac{N_i + N_i^{\divideontimes}}{\sum_{j=1}^{n} (N_j + N_{m-j}^{\divideontimes})},\ i, j=1,2,3 \tag{16}$$

5.2 Complete Data Parameters Estimated

In the above simple example, the experimental data is complete. Considering the event $\{OP_1=i\}$ under the conditions of occurrence of event $\{S=1\}$, it has three possible values, i=1, 2, 3.

There is only one parameter in this case for one event node, $\theta=P\{OP_1=i \mid S=1\}$. For the case in 2.3.1, supposing that $\theta=P\{OP_1=1 \mid S=1\}$, the sample data *D* have the following sample composition $D=(D_1,D_2,\ldots,D_m)$. To simplify the computational, two assumptions are made:

First, individual samples D_m in the given θ are mutually independent and its likelihood function is:

$$L(\theta \mid D)=P(D \mid \theta)=\prod_{i=1}^{m} P(D_i \mid \theta) \tag{17}$$

The binomial likelihood can be deduced:

$$L(\theta \mid D)=P(D \mid \theta)=\prod_{i=1}^{3} P(D_i \mid \theta) = \theta^{N_1}(1-\theta)^{N_2+N_3} \tag{18}$$

Since $L(\theta)$ and $\ln L(\theta)$ obtain the extreme value in the same θ, so:

$$\frac{d}{d\theta}\ln L(\theta)=0 \quad (19)$$

It can be calculated:

$$\hat{\theta}=\frac{N_1}{N_1+N_2+N_3} \quad (20)$$

Of course, we can also get that:

$$\hat{\theta}_i=\frac{N_i}{N} \quad (21)$$

where i=1, 2, 3.

Note that:

$$N=\sum_{i=1}^{3} N_i \quad (22)$$

Bayesian estimation of the complete data

In Bayesian theory, θ_i is considered as a random variable and estimation of m is also calculated m posterior probability distribution. Therefore, it need probability distribution $P(\theta)$ to summarize prior knowledge of θ and then likelihood function is utilized to sum up the impact of the data D=(D_1,D_2,…,D_m);

$$L(\theta \mid D)=P(D \mid \theta) \quad (23)$$

After that, it can be derived from Bayesian formula that:

$$P(\theta \mid D) \propto P(\theta) L(\theta \mid D) \quad (24)$$

The Eq. (24) is posterior distribution of θ and Bayesian estimation of θ.

Maximum likelihood estimation is fixed values of $\hat{\theta}_i$. it can predict the probability of later events that will be executed. Bayesian estimation is a probability distribution P (θ |D) and can be used for predicting the probability of occurrence of the next events.

Based on the above two assumptions, Eq. (24) is rewritten as:

$$P(\theta \mid D) \propto\propto \theta^{N_1}(1-\theta)^{N_2+N_3} P(\theta) \quad (25)$$

Let the prior distribution be β distribution $B(\alpha_h, \alpha_t)$,

$$P(\theta)=\frac{\Gamma(\alpha_h+\alpha_t)}{\Gamma(\alpha_h)\Gamma(\alpha_t)}\theta^{\alpha_h-1}(1-\theta)^{\alpha_t-1} \quad (26)$$

where Γ (.) is a Γ function; both α_h *and* α_t are parameters of Γ function or hyper parameter. Assuming that prior distribution $P(\theta)$ is β distribution $B(\alpha_h, \alpha_t)$, is actually that priori knowledge is equivalent to input (α_h +α_t) test cases, where α_h is the new test cases when {OP_1=i}, α_h is the other test cases was executed.

Combining two equations (25) and (26), it can be calculated that:

$$P(\theta \mid D) \propto \theta^{N_1+\alpha_h-1}(1-\theta)^{N_2+N_3+\alpha_t-1} \quad (27)$$

Note that the posterior distribution of θ is P(θ |D), which can also be β distribution $B(N_1+\alpha_h-1, N_2+N_3+\alpha_t-1)$:

$$P\left(\frac{\theta}{D}\right)=\frac{\Gamma(N_1+\alpha_h+N_2+N_3+\alpha_t)}{\Gamma(N_1+\alpha_h)\Gamma(N_2+N_3+\alpha_t)}\theta^{N_1+\alpha_h-1}(1-\theta)^{N_2+N_3+\alpha_t-1} \quad (28)$$

Then, it can be calculated::

$$P\{OP_1=1 \mid S=1\}=\int \theta P(\theta \mid D)d\theta=\frac{\Gamma(N_1+\alpha_h+N_2+N_3+\alpha_t)}{\Gamma(N_1+\alpha_h)\Gamma(N_2+N_3+\alpha_t)}\int \theta P(\theta \mid D)d\theta$$

$$=\frac{\Gamma(N_1+\alpha_h+N_2+N_3+\alpha_t)}{\Gamma(N_1+\alpha_h)\Gamma(N_2+N_3+\alpha_t)}\int \theta\theta^{N_1+\alpha_h-1}(1-\theta)^{N_2+N_3+\alpha_t-1}d\theta$$

$$=\frac{N_1+\alpha_h}{N_1+\alpha_h+N_2+N_3+\alpha_t}=\frac{N_1+\alpha_h}{N+\alpha} \quad (29)$$

Note that

$$N=N_1+N_2+N_3,\ \alpha=\alpha_h+\alpha_t \quad (30)$$

Obviously, Eq. (28) ~ (30) mean that (1) this estimate is mainly dependent on the a priori knowledge when the sample size is small; (2) this estimate is increasingly dependent on data when the sample size becomes larger and larger; and(3) this estimation is close to maximum likelihood estimation when the sample size is extremely large:

$$\hat{\theta}=\frac{N_1}{N_1+N_2+N_3} \quad (31)$$

6. Case Study

In this paper, open source software TerpPaint, developed by Professor Memon and his students, is utilized for testing. It includes three types of windows. In accordance with the functions, the GUI includes five functions, including File**,** Edit, View & Image, Draw graph and Filter & Layer. Each function shows different profile including operational profile as shown in Table 1.

Table 1. Operational Profile for Five Functions in State1

OP_1	Function				
	File	***Edit***	***view&image***	***draw graph***	***Filter&layer***
state 1	0.34	0.65	0.30	0.03	0.16
state 2	0.20	0.08	0.06	0.94	0.23
state 3	0.46	0.27	0.64	0.03	0.61

6.1 Assessment Process

Bayesian model testing techniques algorithm is as follows:

Operational profile: The original test suite is divided into three equivalent classes. It is given as section 2*d*:$OP(\lambda)=\{<W_i, P_{ij}, M_j>, i=1,2,3\}$;

Functional profile： The original test suite is divided into five classes. It is given as section2e: $FP(\lambda)=\{<F_i, P_{ij}, M_j>, i=1,2,3\}$;

Initialization parameter *N*, *N* means that test step,$n=0$;

Initialization failure detection rate parameter $\theta_O^f=\{\theta_1, \theta_2, \theta_3\}$and $\theta_F^f=\{\theta_1, \theta_2, \ldots, \theta_5\}$;

Building network and initializing parameters

According to the operational profile to select a number of test cases as *section4a(2);*

Record the results of testing, parameter learning and data updating;.

$n=n+1$, if,$n<N$,return to *step f*,

else, go to *step i.*

Assess the reliability of GUI according to Nelson's model, parameter $\theta_o^f=\{\theta_1, \theta_2, \theta_3\}and\theta_F^f=\{\theta_1, \theta_2, \ldots, \theta_5\}$;

Check test termination conditions and terminate the test.

6.2 A Real Bayesian Model

A real model in our case study is shown in Fig. 6. This model is suitable for the case that length of test case is 2. Every node is introduced in section4*b*.

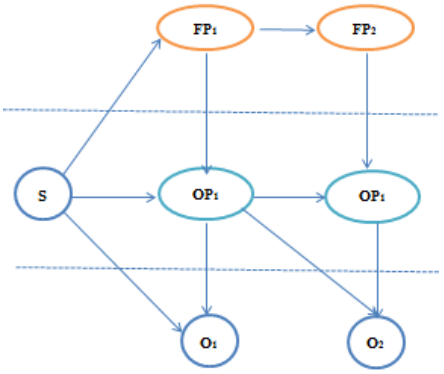


Figure 6. A Real Bayesian Network Model of GUI

6.3 Basic Results

6.3.1 Failure Detection Rate of Operational Profile:

In Fig. 6, most of failures are recorded by node O_2. The operational profile has 2 nodes: OP_1 and OP_2. The failure detection rates of OP_1 and OP_2 are shown in Fig. 7 and Fig. 8 respectively.

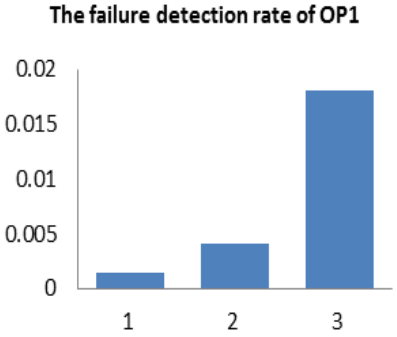


Figure 7. The Failure Detection Rate of OP_1

The node OP_1 has three states: State 1, 2, and 3 respectively represents the 1-st type, the 2-nd type, and the 3-rd type event handler. In Fig.7, it is found that the 3-rd type event handler has the highest failure detection rate, while the 1-st type event handler has the lowest failure detection rate. Apparently, the 3-rd type event handler should be paid more attention in advance to improving the reliability of GUI.

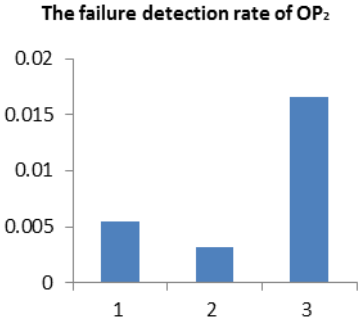


Figure 8. The Failure Detection Rate of OP_2

Similar to OP_1, the node OP_2 also has the three states. In Fig.8, it is found that (1) the 3-rd type event handler has the highest failure detection rate; and (2) the 2-nd type event handler has the lowest failure detection rate.

6.3.2 Failure detection rate of function profile:

In Fig. 6, the functional profile has 2 nodes: FP_1 and FP_2. The failure detection rates of FP_1 and FP_2 are shown in Fig.9 and Fig.10, respectively.

In Fig. 9, the node FP_1 has five states: state file, state edit, state view, state graph, and state filter mean the function of file, edit, view & image, draw graph, and filter & layer respectively. It is found that (1) the functions of file, view and filter & layer have the similar failure detection rate, and (2) function of draw graph has the lowest failure detection rate.

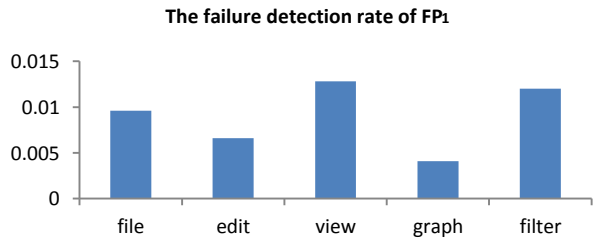


Figure 9. The Failure Detection Rate of FP_1

Similar to FP_1, the node FP_2 has the five states. In Fig. 10, it is shown that (1) the functions of file, edit, view & image and filter & layer have the similar failure detection rate, and (2) function of draw graph has the lowest failure detection rate.

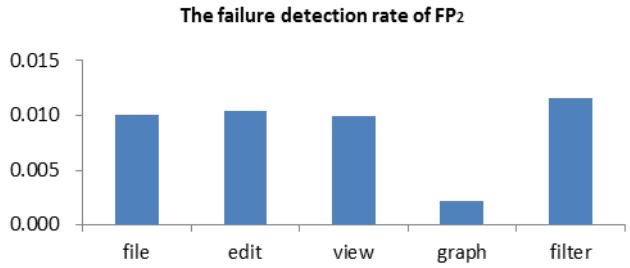


Figure 10. The Failure Detection Rate of FP_2

In both Fig.9 and Fig.10, the function of draw graph has the lowest failure detection rate. As a result, the other functions should be paid more attention –for improving the reliability of GUI.

6.4 The GUI Failure Detection Rate Influenced by Function Interaction:

In Fig.6, different functional requirements lead to the distinct customer experience.The failure detection rates of five function interaction are shown in Table 2.

Table 2. Failure Detection Rate Influenced by Function

FP_2	FP_1				
	File	***Edit***	***view***	***graph***	***Filter***
File	0.0149	0.0136	0.0116	0.0009	0.0158
Edit	0.0101	0.0091	0.0080	0.0001	0.0106
view	0.0196	0.0175	0.0147	0.0003	0.0205
graph	0.0028	0.0044	0.0045	0.0044	0.0043

FP_2	FP_1				
	File	***Edit***	***view***	***graph***	***Filter***
Filter	0.0189	0.0172	0.0145	0.0010	0.0200

In Table 2, it is shown that the failure of function interaction is subject to the influence of its sequential function interaction. For example, the failure of function interaction view-edit is 0.0080, while the failure of function interaction edit-view is 0.0175. It also explains the necessity of defining structural profile of GUI.

6.4.1 Posterior Probabilities:

The posterior probabilities of operational layer are then computed within the given *failure* level shown in Fig.11.

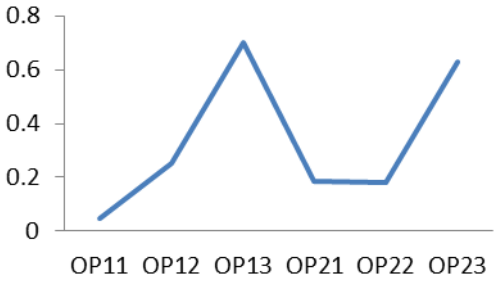


Figure 11. The Posterior Probabilities of Operational Layer

In Fig.11, OP11, OP12, OP13, OP21, OP22, OP23 are state 1 of node OP_1, state 2 of node OP_1, state 3 of node OP_1, state 1 of node OP_2, state 2 of node OP_2, and state 3 of node OP_2. In this figure, it is found that the probability of OP13 and OP23 when the node O_2 is failure. More attention should be paid to OP13 for improve the reliability of GUI.

6.4.2 The Reliability of GUI:

In Fig. 6, the reliability of GUI is derived by combning the structual of fuctional profile and failure detection of operational profile . The Transition matrix is:

$$M_1 = \begin{pmatrix} 0.15 & 0.18 & 0.15 & 0.18 & 0.16 \\ 0.18 & 0.19 & 0.17 & 0.17 & 0.18 \\ 0.12 & 0.13 & 0.23 & 0.11 & 0.12 \\ 0.35 & 0.31 & 0.28 & 0.36 & 0.36 \\ 0.20 & 0.19 & 0.17 & 0.18 & 0.18 \end{pmatrix}$$

According to function profile, operational profile and transition matrix, the reliability of GUI will be calculated as R=0.988

7. Conclusion and Future

This paper describes an effective reliability assessment model for GUI, discusses the reliability model structure and its issues encountered in the modeling process. With the challenge of the GUI's complex architecture, information about GUI function, windows and their interactions each other can be utilized to guide the reliability assessment process and establish confidence assessment of GUI. A case is presented to verify the validity of the model during the GUI reliability assessment process. The model also reveals that GUI reliability will changes significantly with different profiles. It explains that GUI reliability has more complicacy than traditional software.

Of course, the work of this paper is not perfect, the model is too simple to apply directly to GUI. In the near future, more comprehensive Bayesian model will be discussed by taking into account of more factors.

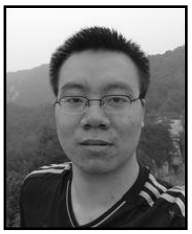
References

[1] F.Belli, M.Beyazit and N.Güler, "Event-Based GUI Testing and Reliability Assessment Techniques -- An Experimental Insight and Preliminary Results", IEEE 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW), **(2011)**.
[2] L.White and H.Almezen, "Generating Test Cases for GUI Responsibilities Using Complete Interaction Sequences", Proc.the 11th International Symposium on Software Reliability Engineering, **(2000)**.
[3] L.White, H.Almezen and N.Alzeidi, "User-Based Testing of GUI Sequences and Their Interactions", Proc. the 12th International Symposium on Software Reliability Engineering , **(2001)**.
[4] P.A. Brooks and A.M.Memon, "Automated GUI Testing Guided by Usage Profiles", Proc. 22nd IEEE/ACM international conference on Automated software engineering , **(2007)**.
[5] X.Yuan, M.B.Cohen and A.M.Memon, "Towards Dynamic Adaptive Automated Test Generation for Graphical User Interfaces", IEEE International Conference on Software Testing, Verification, and Validation Workshops ,**(2009)**.
[6] X.Yuan, M.B.Cohen and A.M.Memon, "GUI Interaction Testing: Incorporating Event Context", IEEE Transactions on Software Engineering, **(2011)**, pp. 559-574
[7] L.Zhao, "GUI Software Testing based on Event Handlers", Beijing:Beihang University, **(2010)**.
[8] Z.F.Yang, Z.X.Yu and C.G.Bai, "The approach of graphical user interface testing guided by bayesian model" , Proc. the 2013 International Conference on Computer Engineering and Network , **(2013)**.
[9] JUDEA PEARL, "Causality: Models, Reasoning, and Inference", Cambridge University Press, **(2000)**.
[10] C.G.Bai, C.H.Jiang, *et al*, "A Reliability Improvement Predictive Approach to Software Testing with Bayesian Method", Proc. 29th Chinese Control Conference, **(2010)**.
[11] D. A. Wooff, *et al*, "Bayesian Graphical Models for Software Testing", IEEE Transactions on Software Engineering, **(2002)**, pp.510-525.
[12] http://www.webopedia.com/TERM/W/window.html
[13] http://en.wikipedia.org/wiki/Markov_chain
[14] E.Nelson,"Estimating software reliability from test data," Microelectronics and Reliability, **(1978)**, pp.67–74
[15] M. R. Lyu , "Handbook of Software Reliability Engineering". McGraw-Hill publishing, **(1995)**.
[16] A. Wood, "Software Reliability Growth Models" .Tech Report: TR-96.1, Tandem Computer Inc, **(1996)**.
[17] M. Xie, "Software Reliability Modelling", World Scientific, **(1991).**
[18] http://users.ece.cmu.edu/~koopman/des_s99/sw_reliability/#tools
[19] F.Belli, M.Beyazit, N.Güler, "Event-Oriented, Model-Based GUI Testing and Reliability Assessment-Approach and Case Study", Advances in Computers, **(2012)**, pp.277-326.
[20] B.B.Yin, Y.Shi, C.G.Bai and K.Y.Cai," A Case Study for Invalidating the Markovian Property of GUI Software Structural Profile" Proc.the 30th International Computer Software and Applications Conference, **(2006)**.

Authors

Zhi-Fang Yang. He received his M.Sc. in Materials Science and Engineering (2008) from Zhengzhou University. Now he is a PhD student of Control science and Engineering at Automation Science and Electrical Engineering Department, Beihang University. His

current research interests include software testing, Bayesian network and reliability of software.



Zhong-Xing Yu. Now he is a PhD student of Control science and Engineering at Automation Science and Electrical Engineering Department, Beihang University. His current research interests include different aspects of software testing.

Bei-Bei Yin. She received his PhD in Control science and Engineering (2010) from Beihang University. Now he is researcher of informatics at Control science and Engineering at Automation Science and Electrical Engineering Department, Beihang University. She is a Academic Visitor of Flinders University in Australia from sep. 2006 to Dec. 2006. His current research interests include different aspects of software testing and software reliability.

Cheng-Gang Bai. He received his PhD in Control science and Engineering (1999) from Zhejiang University. Now he is full professor of Control science and Engineering at Automation Science and Electrical Engineering Department, Beihang University. His current research interests include different aspects of Bayesian network and reliability of software.

