

Reliability Test Cases Generation of Web Application Based On Log Analysis

Shan Zhang ^a, Junfei Huang ^b

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

^a954347891@qq.com, ^bhuangjunfei@bupt.edu.cn

Abstract. With the popularity of web application, its reliability is getting more importance. In this paper, we explore a method to generate web application reliability test cases based on log analysis. This method consists of four steps or modules: preprocessing logs, constructing Markov chain, generating test cases, determining adequacy. In the process, we propose some theories including design pattern and save model of Markov chain for web application and an adequacy determination algorithm. We make an experiment using part of BUPT official website logs. By analyzing the experiment result, we can confirm the feasibility and validity of the method.

Keywords: Web application reliability test; Log mining; Markov chain; Adequacy determination.

1. Introduction

With the rapid development of Web Technology, the kinds of Web applications are constantly increasing. Compared to traditional software, Web applications have a lot of special features, such as distribution, concurrency, large amount of data, heterogeneity, high update speed and many more. Due to the popularity and the intense competition, Web applications occupy an increasingly important position, and the development cycle is getting shorter. The Reliability of software has been attached more importance.

Software reliability is defined as the probability of the software failure-free operation in a certain period of time. Reliability testing for Web applications is similar to traditional software reliability test, which is aimed to measure its probability of no failure operation within a certain time interval by simulating real user behavior. There are four ways to generate Web application test case: record/playback method; html analysis; source code analysis; log analysis. The four method have their own advantages and disadvantages and application scenarios. This

In this paper, we studied the reliability test case generation method for web application which include analyzing the web access log, extracting user behavior info, establishing use model, generate test case and determining adequacy. With reference to the test method based on model, we choose the Markov chain as use model. The advantage of Markov chain[1,2,3] is its Markov property which corresponds to web applications. And the Markov theory is currently the most widely used and relatively mature expression. Related to the adequacy criteria, currently there are two typical algorithm- Euclidean distance and Discriminant value. But the two algorithm can't fulfil our requirements.

To solve the problems above, we proposed a design pattern and storage model based on Markov chain for web applications and a new adequacy determination algorithm applied for our system.

2. Related Work

2.1 Web Application Reliability Testing

Software reliability is defined as the probability of the software failure-free operation in a certain time interval. Software reliability testing in order to meet user requirements for software reliability by quantitatively describing the product's use, making testing more realistically reflect the actual situation, testing the software and detecting and correcting software defects. Web application reliability testing is the same as software reliability, additionally, it is also within the scope of web application performance testing it. Related studies show: the quality of Web application performance testing depends on the degree of load simulates actual situation.

There are four methods to simulate user behavior in web application. Their advantages and disadvantages are as follows:

(1) record/playback method

Advantages: Simple and practicable; Regardless of implementation technology of web application.

Disadvantages: A lot of manual labor and time is need when the test suite is large.

(2) html analysis

Advantages: It can be applied to various aspects of testing for web application such as functional testing, security testing and so on; Regardless of implementation technology of web application.

Disadvantages: There appears no effective method to automatically fill forms in the pages. It can result in combinatorial explosion and loop problem when generating test cases from use model.

(3) source code analysis

Advantages: It can find the hidden interface which make the test more fully and obtain higher code coverage.

Disadvantages: The commonality is poor.

(4) log analysis

Advantages: The raw data which is used to generate test cases is easy access and it can reflect the actual users' behavior. Regardless of implementation technology of web application.

Disadvantages: It is difficult to detect a POST parameter values passed over. The access data accumulation is necessary, so the application can't use this method for testing before putted into use. And it is difficult to test never used functions.

The web application records a large number of log files in the service process, which makes it possible to analysis the application's use by log files. Comparing the above-mentioned methods, we find the method of analyzing log files is more effective to simulate real user behavior. So this paper used this method to generate web application reliability test cases. It can be divided into four steps: 1) Preprocess log. 2) Construct use model. 3) Generate test cases. 4) Determinate adequacy.

2.2 Markov Chain Model

Markov Chain's main feature is Markov property. The Markov property refers: under the condition of present known state, the future states are independent on all states in the past.

Markov Chain consists of states, edges, incentives and transition probability [4-6] of every edge. State represents the state and internal environment of the software in use. Edge represents the software transits to another state when it encounters an incentive. Incentive may be from outside the system, such as user input, click, and it can also be from inside. In actual use, the emergency of different incentives follow a statistical distribution, namely the statistical distribution of use profile. Therefore, edges in Markov chain follow the statistical distribution. Every edge has its own transition probability. Each edge of the model is different, so is each state.

Features of the model are as follows:

(1) There is a unique starting state.

(2) There is a unique end state.

(3) There are a number of intermediate states and each state is different.

(4) There are a number of edges that are connected to the respective state.

Each path from the starting state to the end state of Markov chain represents a running process of software. Markov chain can be described in the forms of maps, tables, or matrices. Below is a simple Markov chain model described by a directed graph.

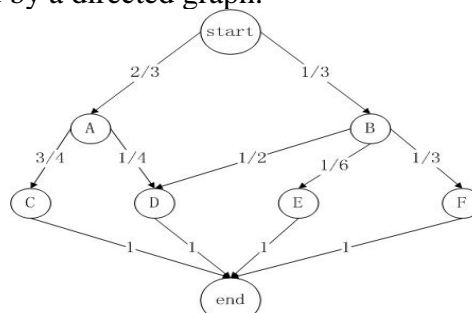


Fig. 1 Markov chain example

2.3 Adequacy Determination[3, 6, 7] of Test

The adequacy criteria based on Markov chain is a model comparison criteria. The usage chain is Markov chain constructed by analyzing log files. The test chain is constructed in process of generating test cases. We can believe the test is full when the difference between use chain and test chain is small enough. Then the reliability calculated in the test environment can represents the actual reliability.

The typical adequacy criteria based on Markov chain are Euclidean distance [8] and discriminant value [1]. But the accuracy of Euclidean distance is low meanwhile it is a requirement to calculate discriminant value which is that the test chain must cover all edges in use chain. So above-mentioned algorithms all can't meet our requirements. A new applicable algorithm to indicate the difference between Markov chains is proposed in this paper.

3. Design of Markov Chain for Web Application

In essence, the operation on Web application is the transition of state. In this process, there are three elements-initial state, state transition condition and state after transition. In Web application, the URL is just corresponding to the state and the user operation is corresponding to the state transition condition. This indicates that creating use model by using a state transition diagram is feasible. Markov chain of Web applications is designed as follows:

- (1) The URL corresponds to the state of Markov chain
- (2) User operations such as click, input correspond to the state transition condition of Markov chain.
- (3) The partial order relation of URLs corresponds to the edge of Markov chain.

Below is a Markov chain of Web application example:

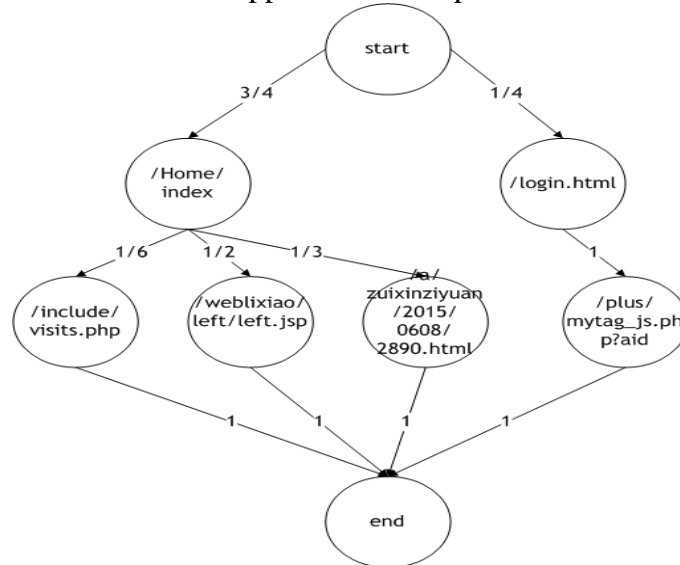


Fig. 2 Markov chain of Web application example

4. New Adequacy Determination Algorithm

According to the description, one of the two current typical adequacy determination algorithms has low accuracy, and another require the test chain cover all edges of use chain. None of them can meet our requirement. So we propose a new algorithm new algorithm to determinate adequacy based on Markov chain. Using this algorithm, we calculate the difference value between use chain and test chain. The value ranges from 0 to 1. The value equals 0 if and only if the test chain is exactly the same as use chain what means that there is no difference. The values equals 1 that means the difference biggest. The formula is as follows:

$$d = \sum_{i=1}^n \omega_i \frac{|p_i - p_{it}|}{p_i + p_{it}} \quad (1)$$

It is equals to:

$$d = \sum_{i=1}^n \omega_i (1 - \frac{2\min(p_i, p_{it})}{p_i + p_{it}}) \quad (2)$$

In this formula, n is the edge count of Markov use chain. P_i is transition probability of the i th edge. W_i is weight of the i th edge, and its formula is as follows:

$$\omega_i = \frac{p_i}{\sum_{i=1}^n p_i} \quad (3)$$

The features of the algorithm is as follows:

(1) Nonnegative: $d \geq 0$.

Proof: Because $|p_i - p_{it}| \geq 0$, so $0 \leq p_i \leq 1$, $0 \leq p_{it} \leq 1$, so $p_i + p_{it} \geq 0$, and $0 \leq \omega_i \leq 1$, so

$$d = \sum_{i=1}^n \omega_i \frac{|p_i - p_{it}|}{p_i + p_{it}} \geq 0 \quad (4)$$

(2) Extremum property: the max value of d is 1.

Proof: Because $0 \leq p_i \leq 1$, $0 \leq p_{it} \leq 1$, so $0 \leq \frac{|p_i - p_{it}|}{p_i + p_{it}} \leq 1$, while $p_{it} = 0$, it has $\frac{|p_i - p_{it}|}{p_i + p_{it}} = 1$, which is the max value.

(3) Affirmative

Proof: While $\text{MarkovChain} = \text{MarkovChain}_{\text{test}}$, $\forall p_i = p_{it} (i = 1, 2, \dots, n)$, obviously, while $p_i = p_{it}$, $\frac{|p_i - p_{it}|}{p_i + p_{it}} = 0 (i = 1, 2, \dots, n)$, so $d = \sum_{i=1}^n \omega_i \frac{|p_i - p_{it}|}{p_i + p_{it}}$.

5. Implementation of Web Application Reliability Test Cases Generation

5.1 The Module of Log Files Preprocessing [9, 10]

Web log preprocessing is intended to eliminate useless property and data in the web log mining process and save the data in the form which mining algorithms can recognize. According to the composition structure of Web application Markov chain, we need to extract useful information from the original web access logs. Then the information is used in the process of construct Markov chain. The steps of preprocess log is as follows:

(1) Data clean

Because web log mining is aimed to study the web user behavior, so only use data which can accurately describe user browsing behavior to mine that we can find the right rules and patterns. HTTP requests can be used to accurately describe users browsing behavior. The result of a HTTP request is a web page rendered in the browsers, whereas the request may be trigger others requests so that there are more than one record written into the access log. Generally the request for html are used to describe the user behavior.

In this paper, data cleaning is to delete irrelevant data by checking the response code, domain and URL suffix. Specifically, the requests whose response code prefix is not 2, domain is not in the range of we study and URL suffix is not html, htm, jsp, php and so on will be filtered.

(2) Dimensions reduction from URL to interface

The composition of URL is: protocol://domain or IP address/path [? query]. In that, the part of query may be multiple key-value pair, and the key is parameter's name meanwhile value is parameter's value. According to the web application design, calling the same interface makes the same impact on sever. The nature of a handle of HTTP request is an interface call. Due to the query parameters can be assigned different values, so an interface can correspond to many URLs. It is a many-to-one relationship between URL and interface. In order to facilitate subsequent statistical analysis, we reduce dimensions from URL to interface. The implement has two steps: remove the query parameters' values but remain names; Query parameters name will be sorted alphabetically. An example: The initial value of URL is '/ Ashx / Census Cus2. Ashx? Type=1 & title=% E5% AE% A3% E4% BC% A0% E7% AB% 99 & remarks='. After treatment, it change into '/ Ashx / Census Cus2. ashx? Remarks & title & type'.

5.2 The Module of Markov Chain Construct and Save

After web log preprocessing, we need to statistic and save useful info when read the record in log. Using the *Mysql* database, we save the info into two table. One is *urlinfo*, and another is *edgeinfo*.

The columns of urlinfo include url(transformed URL namely interface), url_default (a default URL of an interface), times(statistical transformed URL times). The columns of edgeinfo include fromUrl (edge's tail vertex), toUrl (edge's head vertex), times (statistical edge times), frequency (edge's transition probability [11]), frequency Sum, weight.

Some related formulas are as follows:

- (1) $frequency_t = \frac{time_{St \text{ in } edgeinfo}}{time_{url \text{ in } urlinfo=fromUrl}}$;
- (2) $frequencySum_t = \sum_{i=1}^t frequency_{ith \text{ edge with same fromUrl}}$;
- (3) $weight_t = \frac{frequency_t \text{ in } edgeinfo}{count(url) \text{ in } urlinfo \text{ except for end node}}$.

To construct Markov chain, we need to statistic and save information above into corresponding table after preprocessing logs. Therefore it is just the constructing process that we statistic times and calculate other columns value according to formulas above. The process is as follows:

```
While (tempString = reader.readline() != null) {
    Preprocess ( tempString);
    referrerUrl = getFromUrl (tempString);
    url = getToUrl (tempString);

    // set times = times + 1 where url = url;
    updateDB (urlinfo, url);

    //set times = times + 1 where fromUrl = referrerUrl, toUrl = url;
    updateDB (edgeinfo, referrerUrl, toUrl);
}
completeModel () {
    for (edge : edgeinfo) {
        //calculate frequency, frequencySum, weight of every edge
        calculate();
    }
}
```

Taking the Markov chain of figure 1 to example, the result is as follows:

Table 1 edgeinfo of figure 1 Markov chain

fromUrl	toUrl	frequency	frequencySum	weight
Start	A	2/3	2/3	2/3/7
Start	B	1/3	1	1/3/7
A	C	3/4	3/4	3/4/7
A	D	1/4	1	1/4/7
B	D	1/2	1/2	1/2/7
B	E	1/6	2/3	1/6/7
B	F	1/3	1	1/3/7
C	End	1	1	1/7
D	End	1	1	1/7
E	End	1	1	1/7
F	End	1	1	1/7

5.3 The Module of Test Cases Generation

In this paper, we take the random generation method[12] to generate test cases. Each test case generation process is a path from the initial state to the final state in the Markov chain. In this process, each time arriving at a node, a random number ranging from 0 to 1 is produced to make a choice of selecting the next node. Taking the Markov chain of figure 1 to example, if current node is B, there are three next nodes we can choose including D, E, F. The three nodes frequencySum are 1/2, 2/3, 1 respectively shown in table 1 . Then we generate a random value in [0, 1]. If the value is less than or equal to 1/2, the next node is D. If the value is more than 1/2 and less than or equal to 2/3, the next

node is E. If the value is more than $\frac{2}{3}$ and less than or equal to 1, the next node is F. Start at the start node and repeat the process above until reaching the end node, then we finish a test case generation.

5.4 The module of Adequacy determination

Obviously only one test case can't meet our requirement. We use a loop to generate test cases. With the generation of test cases, the test chain is also constructed and saved into tables *urlinfotest* and *edgeinfotest*. The method is the same as use chain. To judge whether we can stop the loop, we need to calculate the difference value between use chain and test chain using the algorithm proposed.

```

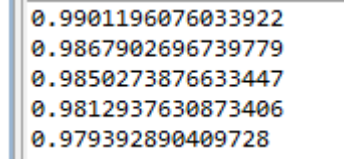
calculateDifference() {
    res = 0;
    for ( edge : edgeinfo) {
        frequency = getFrequency(edge, edgeInfo);
        weight = getWeight(edge, edgeinfo);

        frequencyTest = getFrequency(edge, edgeinfoTest);
        res += abs(frequency - frequencyTest) / (frequency + frequencyTest);
    }
}

```

6. Experiment and Verification

We got part of web logs of BUPT official website with 556214 records totally 14M. After web log preprocessing, the count of url in urlinfo is 5151, and the count of edges in edgeinfo is 11690. It means that the states of use chain are a total of 5151, and edges are 11690. In the process of generating test cases, each generating one hundred test cases, we print the calculated difference values in console. The result is as follows:



```

0.9901196076033922
0.9867902696739779
0.9850273876633447
0.9812937630873406
0.979392890409728

```

Fig. 3 Printed result of difference values

The result indicates that the difference value is decreasing with the test cases quantity increasing. It means that the difference between use chain and test chain is getting smaller which conforms to reality. So if the value reaches to threshold we set, the generating process can be stopped. Then we think the test chain can completely represent actual user behavior. Therefore this method is validate to generate web application reliability test cases, and we can set different threshold under different conditions.

7. Conclusion

In this paper we contribute a web application reliability test cases generation method based on access logs. We propose a design pattern and storage model of Markov chain for web application and a new decision algorithm to determinate adequacy. To evaluate the method, we make an experiment using part of BUPT official website logs. The experiment result verified the feasibility and validity of the method and determination algorithm.

Acknowledgments

This project is supported by Key Technology Research and System Development of Code Automation Coverage Testing of National Natural Science Foundation of China (No. 91318301).

References

- [1] J. A. Whittaker, M. G. Thomason. A Markov Chain Model for statistical software testing [J].IEEE Transactions On Software Engineering 1994, 30 (10), p.812-824.
- [2] J. H. Poore, H. Mills, D. Mutchler. Planning and certifying software system reliability [J].IEEE transaction on Software Engineering, 10 (1), 1993, p.88-99.
- [3] J. A. Whittaker, J. H. Poore. Markov analysis of software specification [J]. ACM Transactions on Software Engineering and Methodology, 1993, 2 (1): p.93-106.
- [4] G. H. Walton, J. H. Poore, C. J. Trammell. Statistical testing of software based on a usage model [J]. Software Practice and Experience, 1995, 25 (1): p.97-108.
- [5] S. J. Prowell. Computations for Markov Chain Usage Models[R].Technical Report, University of Tennessee, UT-CS-03-505.www.utk.ed.
- [6] J. A. Whittaker, J. H. Poore. Markov Chain model for statistical software testing [J].IEEE Transactions on Software Engineering 30 (10): p.812-824, 1994.
- [7] Goodenough J B, Gerhart S L. Toward a theory of test data selection. IEEE Transactions on Software Engineering, 1975, SE-3: p. 156-173.
- [8] Kirk sayre. Improved techniques for software testing based on Markov chain usage models [D]. Knoxville: University of Tennessee, 1999.
- [9] JIANG Chang-bin. Application of Cloud Model in Personalized Service Recommendation of Web Log Mining. 2010 International Conference on Biomedical Engineering and Computer Science,2010
- [10]P. Nithya; P. Sumathi. Novel Pre-Processing Technique for Web Log Mining by Removing Global Noise and Web Robots. Computing and Communication Systems (NCCCS), 2012 National Conference on, 2012
- [11]J. A. Whittaker, J. H. Poore, Statistical testing for Cleanroom Software Engineering [A]. Proceedings of the Hawaii International Conference on Systems Science, IEEE Computer Society Press, 1992, 2: p.428-436.
- [12]J. W. Duran and S. C. Ntafos. An evaluation of random testing [J]. IEEE Transactions on Software Engineering, 1984, 10 (4): p.438-444