International Advance Journal of Engineering, Science and Management (IAJESM)
ISSN -2393-8048, January-June 2022, Submitted in June 2022, iajesm2014@gmail.com

Study on the Reliability Evaluation of the Web-Based Software

Kavita Vijaysingh Tiwari, Research Scholar, Department of Computer Science, Monad University, Hapur, Uttar Pradesh (India)
Dr. Kailash Kumar Assistant Professor, Department of Computer Science, Monad University, Hapur, Uttar Pradesh (India)

Abstract:

The simulator developed in this chapter has been executed for 100000 number of simulation runs in order to compute system reliability for web-based software. The results obtained for web-based software reliability corresponding to different data sets of values of web page reliability. It is found that the system reliability increases as the web page reliability increases. The simulator described in his chapter will be of great importance to evaluate the reliability of web based software. The transition probabilities of web pages which are not connected directly are considered as zero when ideal transition probabilities for the web pages are entered. Each state is prone to failure to some extent, thereby making the web software imperfect. The simulator is executed using various combinations of reliabilities of web pages and it is found that the system reliability is sensitive to the reliability of individual web pages. It is depicted from the graph plotted between web page reliability and web-based software reliability. It is also evident that the web-based software reliability increases as the number of simulation runs increases from the results. This simulator can be used as an access tool by the software quality assurance team for estimating the reliability of web-based system. The reliability evaluation will act as a metric of amount the software quality assurance needs for meeting software project quality objectives.

Keywords: Software, Simulators, Web, Reliabiity

INTRODUCTION

Web-based systems are playing an important role in modern computer savvy society today. Because of the pervasive nature and the massive user population, various existing software engineering approaches need to be adopted for web engineering and web quality assurance (WebQA) is the application area which deals with analysis, testing, quality/reliability improvement for web-based applications.

Davila-Nicanor et al. [DAV2005] focused on the development of a methodology for the evaluation and analysis of the reliability of web-based software applications. They tested the methodology in a web-based software system and used statistical modeling theory for the analysis and evaluation of the reliability. The behavior of the system under ideal conditions was evaluated and compared against the operation of the system executing under real conditions. The evaluation and improvement process is performed in their methodology to evaluate and improve the quality of the software system.

Jeff Tian et al. [JEF2004, ZLI2003] discussed web usage and problems for web applications, evaluate their reliability and examine the potential for reliability improvement. Based on the characteristics of web applications and the overall web environment, they classify web problems and focus on the subset of source content problems. Using information about web accesses, they derive various measurements that can characterize web site workload at different levels of granularity and from different perspectives. These workload measurements, together with failure information extracted from recorded errors, are used to evaluate the operational reliability for source contents at a given web site and the potential for reliability improvement.

Jeff Tian and L. Ma [JEF2003, KAL2001] characterized the problems for web applications, examined existing testing techniques that are potentially applicable to the web environment, and introduced a strategy for web testing aimed at improving web software reliability by reducing web problems closely identified with web source contents and navigations whole by analyzing the dynamic web contents and other information sources.

C. Kallepalli and J. Tian [KAL2001] used statistical testing and reliability analysis effectively to assure quality for web applications. To support this strategy, they extract web usage and failure information from existing web logs. The usage information is used to build models for statistical web testing. The related failure information is used to measure the reliability of web applications and the potential effectiveness of statistical web testing.

L. Ma and J. Tian [JEF2006, LMA2007] discussed defect classification and analysis framework, orthogonal defect classification, to analyze web errors and identify problematic areas for focused reliability improvement. Based on information extracted from existing web server logs, web errors are classified according to their response code, file type, referrer type, agent type, and observation time.

Web-based software is a collection of web pages associated with hyperlinks to communicate with other components of the software. Each web page is considered as a functionally independent component of web based software. The evaluation is carried out using Markov analysis which looks at a sequence of events and analyses the tendency of one event to be followed by another. Using this analysis one can generate a new sequence of random but related events which look similar to the original. This Markov process is stochastic in nature which has the property that the probability of transition from a given state to any future state depends only on the present state and not on the manner in which it was reached [GIL2004].

If $t_0 < t_1 < t_2 <\ldots\ldots\ldots< t_n$ represents the points in time scale then the family of random variables $\{X(t_n)\}$ is said to be a Markov process provided it holds the Markovian property :

$$\mathbf{P\{X(t_n) = x_n \mid X(t_{n-1}) = x_{n-1}, X(t_0) = x_0\} = P\{X(t_n) = x_n \mid X(t_{n-1}) = x_{n-1}\}} \quad \forall \quad \mathbf{X(t_0), X(t_1),\ldots,X(t_n)}$$

Markov process is a sequence of 'n' experiments in which each experiment has 'n' possible outcomes $x_1, x_2, \ldots\ldots, x_n$. Each individual outcome is called a state and probability (that a particular outcome occurs) depends only on the probability of the outcome of the preceding experiment. The simplest of the Markov processes is discrete and constant over time. It is used when the sequence of experiment is completely described in terms of its states (possible outcomes). There is a finite set of states numbered 1, 2, 3,…, n and this process can be only in one state at a prescribed time. The system is said to be discrete in time if it is examined at regular intervals e.g. daily, weekly, monthly or yearly.

The probability of moving from one state to another or remaining in the same state during a single time period is called transition probability. Mathematically, the probability

$$\mathbf{P\ x_{n-1},\ x_n = P\{X(t_n) = x_n \mid X(t_{n-1}) = x_{n-1}\}}$$

is called the transition probability. This represents the conditional probability of the system which is now in state x_n at time t_n provided that it was previously in state x_{n-1} at time t_{n-1}. This probability is known as transition probability because it describes the system during the time interval (t_{n-1}, t_n). Since each time a new result or outcome occurs, the process is said to have stepped or incremented one step. Each step represents a time period or any other condition which would result in another possible outcome. The symbol 'n' is used to indicate the number of steps or increments.

The transition probability can be arranged in a square matrix form such that $\sum \mathbf{p_{ij}} = \mathbf{1;\ i=1, 2, 3\ldots..n,}$ and $\mathbf{0 \leq\ p_{ij} \leq 1,}$ where $\mathbf{p_{ij}}$ are the elements of square matrix.

PROPOSED MODEL

Software reliability can be computed analytically with the help software reliability models. These models are based on some assumption for the simplification of the solution [LYU1992, GOE1982, GOE1985]. Modeling approaches discussed in [GOE1985, HAR1987, KHO1991] are extended and applied to web-based software system. As one approaches to more and more realistic and complex situations, it becomes almost impossible to obtain an analytic solution. Then simulation techniques are used.

This simulator is developed for reliability evaluation of web-based software using high level programming language.

Assumptions

- The control flows from one web page to another web page according to Markovian property.
- The process of transition continues till it reaches a state called the terminal state. This ensures the successful completion of the transitions. TSTATE denotes the terminal state and 1, 2, 3 …… n denotes transition states corresponding to web pages numbered as 1, 2, 3,…. n. Thus web software has state space {1, 2, 3….n, TSTATE}.

The first page of the web software is called a home page and corresponds to initial sate 1.

- In the absence of a successful transition from page 1 to TSTATE successively, the process is probable to fail abruptly. Such a state, if it exists, is denoted by FSTATE.
- Each of transient web pages 1, 2, 3….n is prone to failure. The web page i have an associated reliability rel_i which is the probability that the web will operate correctly when invoked and will transfer the control to other web page successfully as and when intended by the user. Therefore, the probability of failure to enter from state i to state j will be (1 - rel_i). The Markov chain for the imperfect web software becomes (1, 2, 3,…..n, FSTATE/ TSTATE). Therefore

$\mathbf{itpm_{ij} = rel_i * tpm_{ij}}$ $\forall$ ***i=1.2,-----n* and**
$\forall$ ***j=1, 2-----n*, TSTATE**

$\mathbf{itpm_{ifstate}, i = 1- rel_i}$ $\forall$ ***j= 1, 2------n*;**

$\mathbf{itpm_{tstate,, tstate} = 1}$

$\mathbf{itpm_{ifstate, fstate} = 1}$

The two states viz. FSTATE and TSTATE are mutually exclusive and any of the transient states *1, 2; ----n* will eventually lead either to a FSTATE or a TSTATE.

System reliability is defined to be the probability that system eventually completes its task successfully without failing from transition from one page to the other till its termination state is reached. When applied to our model it is simply the probability that the Markov chain for web software is eventually absorbed into TSTATE rather than the FSTATE.

Thus RELWEB = NAB / SIMRUN where NAB is the number of times the control goes to observing state out of SIMRUN simulation runs.

DESCRIPTION OF ALGORITHM: SIM_ REL_WEB

Terms and Notations

SIMRUNS	:	Number of simulation runs
REL	:	Reliability vector
NT	:	Counter for successful termination
pij	:	Probability that control will be passed next on to web page j from i
NPAGES	:	No of web pages in the website
TSTATE	:	Terminal state
FSTATE	:	Failure state
SYSREL	:	System reliability
TPM	:	Ideal state transition probability matrix
TPMI	:	Imperfect state transition probability matrix
CTPM	:	Cumulative state transition probability matrix
RELWEB	:	Reliability of web based software

ALGORITHM : SIM_REL_WEB for Reliability evaluation of Web-based Software

1) [Initialize counter for successful termination]
NT = 0
2) [INPUT]
a) [Number of simulation runs]
Read SIMRUNS
b) [Number of web pages]
Read NPAGES
c) [Ideal state transition matrix (TPM) for web-based software system]
For I=1 to NPAGES
For J=1 to NPAGES+1
TPM(I, J) = SNDY(DUM)
[SNDY is a random number generator]
End for

```
            End for
        d) [Reliability vector (rel)]
            for i=1to n
              read rel(i)
            end
3)      [Compute imperfect state transition probability matrix
        (TPMI =REL* TPM]]
                    For I=1 to NPAGES
                For J=1 to NPAGES+1
                    TPMI (I, J) = REL (I) * TPM (I, J)
             End for
            End for
    4) [Compute cumulative state transition probability Matrix]
        For I=1 to NPAGES
          S=0
              For J=1 to NPAGES+1
                S = S + TPMI (I, J)
                    End for
              CTPM (I, J) = S
        End for
5)      [Repeat step 6 through step 8 SIMRUNS times to select the transition path
    using random and CTPM row corresponding to web page I)
        For KOUNT = 1 to SIMRUNS
            RANDOM = SNDY (DUM)
             I = 1
                        If (RANDOM > 0 and CTPM (I, 1) > RANDOM) THEN

                    I=1
                    Loop

                End if

                If (RANDOM > CTPM (I,1) and CTPM (I,2) > RANDOM)

                THEN

                 I = 2
                      Loop

                      End if
                      If (RANDOM > CTPM (I, 3) and CTPM (I, 3) > RANDOM)
                     THEN
                      I = 3
                     Loop
                     End if
                      If (RANDOM > CTPM (I, 4) and CTPM (I, 4)> RANDOM)
               THEN
                I = 4
                Loop
                      End if
                  If (RANDOM > CTPM (I, 5) and CTPM(I, 5) < RANDOM)
                  THEN
                  NT = NT + 1
               Loop
```

End if

End for

6) [Compute estimated reliability]

RELWEB = NT / SIMRUNS

7) Write RELWEB

8) Stop

RESULTS & DISCUSSION

The simulator developed in this chapter has been executed for 100000 number of simulation runs in order to compute system reliability for web-based software. The results obtained for web-based software reliability corresponding to different data sets of values of web page reliability are presented in the table 1. It is found that the system reliability increases as the web page reliability increases.

TABLE 1: Details of inputs: Reliability vector values and output Web-based software Reliability (Rel(i) and S. Rel)

Page Wise Reliability	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5	Data Set 6	Data Set 7
Rel(1)	0.99	0.99	0.99	0.96	0.99	0.99	0.99
Rel(2)	0.95	0.98	0.96	0.97	0.97	0.98	0.97
Rel(3)	0.91	0.92	0.93	0.98	0.95	0.97	0.98
Rel(4)	0.87	0.91	0.9	0.99	0.93	0.96	0.96
S.Rel.	0.84	0.88	0.89	0.92	0.92	0.94	0.95

A graph is drawn (Figure 1) between web page Reliability and web-based software Reliability to show the trend related to web page reliability and website reliability using the data mentioned in table 1.

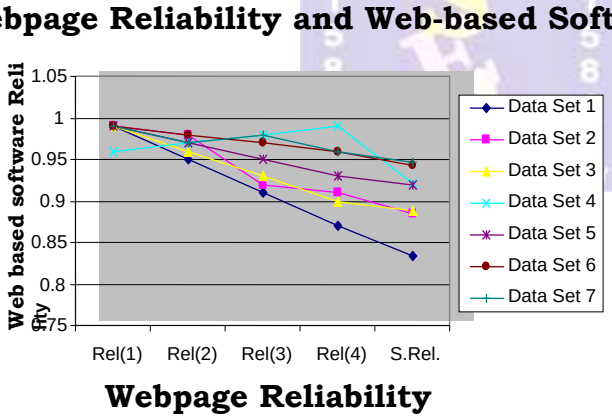

FIGURE 1: Web page Reliability and Web-based Software Reliability

The table 1 gives a view of the system reliability computed for different values of simulation runs keeping the values of reliability vector constant (0.99, 0.98, 0.97, and 0.96). It is found that the system reliability increases as the number of simulation runs increases.

TABLE 2: Details of inputs (Number of simulation runs) and output (Web-based Software Reliability)

Number of Simulation Runs	Web based Software Reliability
1000000	0.947
1010000	0.95
1020000	0.953
1030000	0.9716
1040000	0.981
1050000	0.99

The figure 2 shows the trend relating to number of simulation runs and the website reliability (System Reliability).

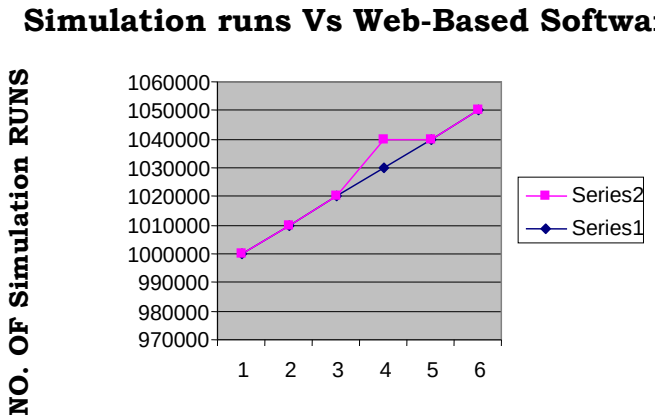

FIGURE 2: Simulation Runs Vs Web-Based Software Reliability

CONCLUSION

The simulator described in his chapter will be of great importance to evaluate the reliability of web based software. The transition probabilities of web pages which are not connected directly are considered as zero when ideal transition probabilities for the web pages are entered. Each state is prone to failure to some extent, thereby making the web software imperfect. The simulator is executed using various combinations of reliabilities of web pages and it is found that the system reliability is sensitive to the reliability of individual web pages. It is depicted from the graph plotted between web page reliability and web-based software reliability (Figure 1). It is also evident from Figure 2 that the web-based software reliability increases as the number of simulation runs increases from the results shown in table 2. This simulator can be used as an access tool by the software quality assurance team for estimating the reliability of web-based system. The reliability evaluation will act as a metric of amount the software quality assurance needs for meeting software project quality objectives.

Reference:

Arcuri, A. and Fraser, G. On parameter tuning in search based software engineering, In the Proceedings of SSBSE, pp. 33-47, 2011.

Arcuri, A. and Fraser, G. Parameter tuning or default values? An empirical investigation in search-based software engineering, Empirical Software Engineering, 18 (3), pp. 594-623, 2013.

Asad, C., Ullah, M. & Rehman, M., 2004. An approach for software reliability model selection. , Int. Computer Software and Applications Conf., (COMPSAC), p. 534–539.

Axelsson S," Intrusion Detection Systems: A Survey and Taxonomy", Technical Report No 99- 15, Dept. of Computer Engineering, Chalmers University of Technology, 2000.

Badis H, Agha K A (2005) QOLSR: QoS routing for ad hoc wireless networks using OLSR. Wiley European Transactions on Telecommunications 15(4):427--442.

Barber, G., Chapter 9 of Human Factors and Interactive Computer Systems, Edited by Yannis Vassiliou, Ablex Publishing Corporation, Norwood, NJ, 1984.

Baresel, A., Harman, M., Binkley, D. and Korel, B. Evolutionary Testing in the Presence of Loop–Assigned Flags: A Testability Transformation Approach, In the Proceedings of the International Symposium on Software Testing and Analysis, 29(4), pp. 108-118, 2004.

Barolli L, Koyama A, Shiratori N (2003) A QoS routing method for ad-hoc networks based on genetic algorithm. Proc. 14th Int. Wksp. Database and Expert Systems Applications 175-179.

Deek, F.P., An Integrated Environment For Problem Solving and Program Development, Unpublished Ph.D. Dissertation, New Jersey Institute of Technology, 1997.

Deek, F.P., Hiltz, S.R., Kimmel, H., Rotter, N., "Cognitive Assessment of Students' Problem Solving and Program Development Skills", Journal of Engineering Education, Volume 88, Number 3, pp. 317-326, July 1999.

Deek, F.P., McHugh, J., " SOLVEIT: An Environment for Problem Solving and Program Development", to appear in Journal of Applied Systems Studies, Special Issue on Distributed Multimedia Systems with Applications, 2000.

Zhao, J., Liu, H.-W., Cui, G. & yang, X.-Z., 2006. Software Reliability Growth Model with Change-Point and Environmental Function. The Journal Of Systems and Software, pp. 1578-1587.

Zhao, M., 1993. Change-point problems in software and hardware reliability. Commun. Statistical-Theory Math., Volume 22, pp. 757-768.

Zou, F. Z., 2003. A Change-Point Perspective on the Software Failure Process. Software Testing, Verification and Reliability, June, Vol. 13(No. 2), pp. 85-93.

Zwass, V., Foundations of Information Systems, Irwin McGraw-Hill, Boston, Massachusetts, 1998.