Simulator for Risk Assessment of Software Project based on Performance Measurement

P. K. Suri[1], Bharat Bhushan[2], Ashish Jolly[3]

1Department of Computer Science & Applications, Kurukshetra University, Kurukshetra (Haryana), India.
2Department of Computer Science & Applications, Guru Nanak Khalsa College, Yamuna Nagar (Haryana), India.
3Department of Computer Science & Applications, Shri Atmanand Jain Institute of Management & Technology, Ambala City (Haryana), India.

Abstract
Any software project under development is having inherent risk. Most of the software's fails due to over budget, delay in the delivery of the software's, poor performance and so on. [1]
Performance is a nonfunctional software attribute that plays a crucial role in wide application domains spreading from safety-critical systems to e-commerce applications. Software risk can be quantified as a combination of the probability that a software system may fail and the severity of the damages caused by the failure. [2]
An attempt has been made to design a simulator for analyzing the performance measurement of Software Risk Assessment using Markov process. The proposed simulator (during the process of software development) gives the incremental risk for every phase and also the total cumulative risk as the software progress from one release to another release.

Keywords:
Risk Assessment, Markov Process, Transition Probabilities, Simulation

1. Introduction

A risk is a potential problem, which is characterized by the likelihood that an event, hazard, threat, or situation will occur and its undesirable consequence [IEEE 2001]. There are many risks involved in creating high quality software on time and within budget. However, in order for it to be worthwhile to take these risks, they must be compensated for by a perceived reward. The greater the risk, the greater the reward must be to make it worthwhile to take the chance. In software development, the possibility of
reward is high, but so is the potential for disaster. The need for software risk management is illustrated in Gilb's risk principle. "If you don't actively attack the risks, they will actively attack you". [3]
Boehm [4] defines risk management as an emerging discipline whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful operation or major sources of rework.

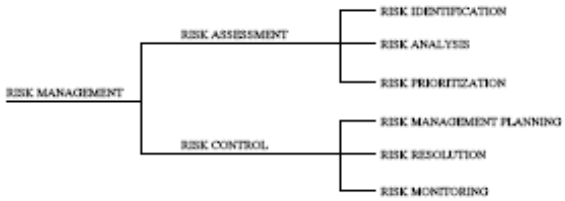


Fig. Software Risk Management Steps

Software development projects are characterized by technical complexity, market and financial uncertainties and competent manpower availability. Therefore, successful project accomplishment depends on addressing those issues throughout the project phases. Effective risk management ensures the success of projects. There are several studies on managing risks in software development projects. Most of the studies identify and prioritize risks through empirical research in order to suggest mitigating measures. Although they are important to clients for future projects, these studies fail to provide any framework for risk management from software developers' perspective. [5]
From the early design phase through the implementation performance assessment of software has been subject to a great variety of approaches in the past. Performance modeling artifacts have to be created and results have to be evaluated. Performance properties often are crucial aspects of a software development project. In many cases only a few parts of the implementation do have to meet critical performance requirements. [6]
In the past decade, several authors have pointed out the potential of simulation as an analysis and decision support tool for software managers. They present a five step simulation-based method to risk assessment, ProSim/RA, which combines software process simulation with stochastic simulation. [7]

Simulation of dynamic complex systems-specifically, those comprised of large numbers of components with

Manuscript received June 5, 2009
Manuscript revised June 20, 2009

stochastic behaviors-for the purpose of probabilistic risk assessment faces challenges in every aspect of the problem. Scenario generation confronts many impediments, one being the problem of handling the large number of scenarios without compromising completeness. [8]

Software risk assessment takes into consideration many aspects of the software product. One of these aspects, with critical influence on risk, is software reliability. For this reason, estimating software reliability measures is very important in the process of software risk assessment. In order to characterize as realistically as possible the evolution of software in time, the software reliability models should take into account the structure of the software. For such models, due to their complexity, mathematical tractability becomes difficult to obtain and simulation is a more flexible alternative. [9]

The software to be released based on the performance measurement may be considered to be in a number of states of deterioration. The performance measurement work of the software is inspected after a regular interval of time say weekly and is classified as being in one of the states. Each state is considered as functionally independent. The evaluation is carried out using Markov analysis which looks at a sequence of states and analyses the tendency of one state to be followed by another, after each release the software restored to a state having 'increased' operating efficiency. Using this analysis one can generate a new sequence of random but related states 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. [10]

A stochastic process whose state at time t is $X(t)$, for $t > 0$, and whose history of states is given by $x(s)$ for times $s < t$ is a Markov process if

$$\Pr\left[X(t+h) = y \mid X(s) = x(s), \forall s \leq t\right] = \Pr\left[X(t+h) = y \mid X(t) = x(t)\right], \quad \forall h > 0.$$

Markov process is a sequence of n experiments in which each experiments has n possible outcomes x_1, x_2, ……, 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 0, 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 eg. daily, weekly, monthly or yearly.

Transition Probability

At each step the system may change its state from the current state to another state (or remain in the same state) according to a probability distribution. The changes of state are called transitions, and the probabilities associated with various state-changes are called transition probabilities.

The controlling factor in a Markov chain is the transition probability, it is a conditional probability for the system to go to a particular new state, given the current state of the system. For many problems, such as simulated annealing, the Markov chain obtains the much desired importance sampling. This means that we get fairly efficient estimates if we can determine the proper transition probabilities.[11]

Mathematically, the probability

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

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}. The symbol n is used to indicate the number of steps or increments.

The transition probability can be arranged in a square matrix form denoted by

$$P = \begin{pmatrix} p_{11} & p_{12} \ldots\ldots\ldots\ldots p_{1n} \\ p_{21} & p_{22} \ldots\ldots\ldots\ldots p_{2n} \\ \vdots & \vdots \\ p_{n1} & p_{n2} \ldots\ldots\ldots\ldots p_{nn} \end{pmatrix}$$

Such that $\sum_{J=0}^{n} p_{ij} = 1$; i=0, 1, 2, 3…..n and $0 \leq p_{ij} \leq 1$

n-step stationary transition probabilities

The n-step stationary transition probabilities are defined to be

$$p_{rs}^{(n)} = P(X_{i+n} = s \mid X_i = r) = P(X_n = s \mid X_0 = r)$$
$$p_{rs}^{(n)} \geq 0 \text{ for all states r and s ; n=1, 2, ……}$$

$$\sum_{s=0}^{n} p_{rs}^{(n)} = 1 \quad \text{for all states r; n=1, 2,….}$$

The above equation assumes there is N+ 1 possible state. Note that if the system is currently in state r, it must be in some state n steps from now. Thus

$$\sum_{s=0}^{n} p_{rs}^{(n)} = 1$$

In general, the n-step stationary transition probabilities can be calculated as follows:

$$p_{rs}^{(n)} = \sum_{j=0}^{n} p_{rj} \cdot p_{js}^{(n-1)}$$

where the possible states are 1, 2,…..N. That is, the probability of going from state r to state s in n steps is the probability of going from state r to state j in one step, times the probability of going from state j to state s in n-1 steps, summed over all j=0, 1, 2,……, N.

First-Passage Time and Recurrence Time

The number of transition (steps) needed to go from state r to state s for the first time is defined to be the first-passage time from state r to state s. It is denoted by T_{rs}.

If the probability of the system going from state r to state s eventually is 1, the corresponding first-passage time T_{rs} is a random variable; otherwise, it is infinity.

The expected value of T_{rs} is give n by

$$\mu_{rs} = E(T_{rs}) = \sum_{t=1}^{\infty} t\, g_{rs}(t)$$

If r = s, the expected first-passage time is called the expected recurrence time for state r. The recurrence time for state r is the reciprocal of the steady state probability of being in state r.

2. Proposed Simulator

The proposed simulator assumed that the new version of the software to be released in the market will be acceptable or not. The efficiency of the newer version will be higher then the previous versions. A lot of efforts will be made by the software developer team to get a steady and stable version of the software to gain maximum operating efficiency and reliability. The software that is currently having version 1 must be in some other version after modifications considering on the basis of Markov process. Under fairly general conditions if the one step stationary transition probabilities are available one can determine n step stationary transition probabilities until the software reaches a steady and stable state.

Assumptions

1. The software to be modified may be considered in one of the three modified states called version 1,2 and 3 representing the next version of the software to be more efficient at the end of a particular release.
2. The operating efficiency is simulated for each state by generating random sample from Normal probability distribution.
3. The one step stationary transition probability may be given or determined from the past data.
4. n step stationary probabilities are calculated successively until the system reaches steady state.
5. Expected first-passage time and recurrence times are computed.

Terms and Notations

NUM : Number of n-step probabilities.
NS : Number of modified states of the software
PI (X0=I) : Probability of being in state I
P1 (I, J) : One step stationary transition probability
PN (I, J) : n steps stationary transition probability
PN (NS, J) : steady-state transition probability
MT (I, J) : probabilities of being in state J after I steps.

Algorithm: Risk_Markov_Sim

To compute n-step probabilities using Markov analysis

Step 1: Start

Step 2.(a) [Input number of states for software modification]
Read NS.
(b) [Read and Compute the probabilities for each modified state using Box-Muller transformation].
(c) [Read one step stationary transition probabilities]
For I=1 to NS do
For J = 1 to NS do
Read P1 (I, J)

Step3. [Calculate n step stationary transition probabilities for n = 1, 2, 3, ….. using the relation]

$$p_{rs}^{(n)} = \sum_{j=0}^{n} p_{rj} \cdot p_{js}^{(n-1)}$$

Step4. [Compute steady state transition probability using the relation]

$$a_s = \sum_{r=0}^{N} a_r p_{rs} \text{ for } s=0, 1, 2,\ldots, N$$

Step 5. [Compute probabilities of being in state j after I steps]

Step 6. [Compute the first-passage time and recurrence time using the relation]

$$\mu_{rs} = E(T_{rs}) = \sum_{t=1}^{\infty} t\, g_{rs}(t)$$

Step 7. [Write Results]
Step 8. [Stop]

3. Case Study:

A company is interested in determining to release the new version of the software or remain with the existing version to get the maximum revenue for a particular period of time. The company tries to modify the software keeping in view the three states of the software viz. performance level 1, 2 and 3 depending upon the features incorporated in the new version using Markov process. It is determined when the software will be stable i.e. in its steady state and will be acceptable in the market, the company will earn maximum profit. Different cases are discussed as below taking into consideration the operating efficiency and the number of states. Their corresponding intermediate transition probabilities and steady state transition probabilities are computed. Also expected first-passage time and recurrence time are computed for transition from one state to another state which is depicted below as Case 1, Case 2 and Case 3.

Case 1

Input: Read the value of Number of states (NS) and compute operating efficiency say 0.17, 0.23 and 0.60.

Table 1 act as input for one step stationary transition probabilities for software modification.

	To State		
From State	0	1	2
0	0.0	0.25	0.75
1	0.0	0.50	0.50
2	0.0	0.20	0.80

Table 1

Output:

1. Intermediate State Transition Probabilities

	To State		
From State	0	1	2
0	.0000	.0000	.0000
1	.1275	.2225	.6500
2	.1105	.2578	.6317
3	.1074	.2521	.6405
4	.1089	.2514	.6397
5	.1087	.2519	.6394
6	.1087	.2518	.6395
7	.1087	.2518	.6395
8	.1087	.2518	.6395

Table 2

2. Steady State Transition Probabilities

State	Steady State Stationary Transition Probabilities
0	0.1087
1	0.2518
2	0.6395

Table 3

3. Expected First-Passage Time and Recurrence Time

	To State		
From State	0	1	2
0	9.1985	2.8571	1.6250
1	8.8235	3.9714	1.6250
2	7.5735	3.7143	1.5638

Table 4

Case 2

Input: Read the value of Number of states (NS) and compute operating efficiency say 0.17, 0.23 and 0.60.

Table 5 act as input for one step stationary transition probabilities for next version of software.

	To State		
From State	0	1	2
0	0.0	0.75	0.25
1	0.0	0.25	0.75
2	0.0	0.15	0.85

Table 5

Output:

1. Intermediate State Transition Probabilities

From State	0	1	2
0	.0000	.0000	.0000
1	.0425	.1700	.7875
2	.1339	.2173	.6489
3	.1103	.2153	.6744
4	.1146	.2150	.6704
5	.1140	.2151	.6709
6	.1141	.2151	.6709
7	.1140	.2151	.6709

Table 6

2. Steady State Transition Probabilities

State	Steady State Stationary Transition Probabilities
0	0.1140
1	0.2151
2	0.6709

Table 7

3. Expected First-Passage Time and Recurrence Time

	To State		
From State	0	1	2
0	8.7681	4.2202	1.2941
1	8.6505	4.6495	1.1765
2	7.4741	4.2936	1.4906

Table 8

Case 3

Input: Read the value of Number of states (NS) and compute operating efficiency say 0.17, 0.23 and 0.60.

Table 9 act as input for one step stationary transition probabilities for next version of software.

	To State		
From State	0	1	2
0	0.0	0.50	0.50
1	0.0	0.75	0.25
2	0.0	0.05	0.95

Table 9

Output:

1. Intermediate State Transition Probabilities

From State	0	1	2
0	.0000	.0000	.0000
1	.0850	.1400	.7750
2	.1318	.2490	.6193
3	.1053	.2537	.6410
4	.1090	.2391	.6519
5	.1108	.2436	.6455
6	.1097	.2438	.6465
7	.1099	.2432	.6469
8	.1100	.2434	.6466
9	.1099	.2434	.6467
10	.1099	.2434	.6467
11	.1099	.2434	.6467
12	.1099	.2434	.6467

Table 10

2. Steady State Transition Probabilities

State	Steady State Stationary Transition Probabilities
0	0.1099
1	0.2434
2	0.6467

Table 11

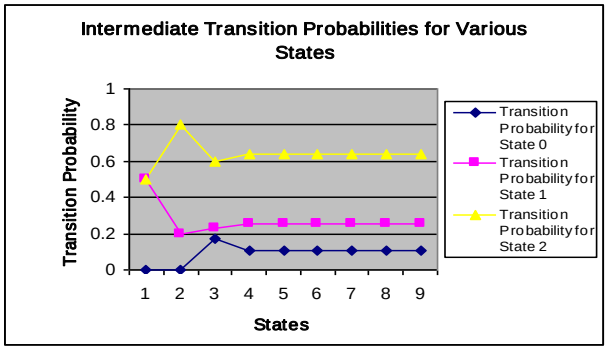
3. Expected First-Passage Time and Recurrence Time

	To State		
From State	0	1	2
0	9.0960	1.8182	1.7895
1	8.3591	4.1091	1.0526
2	7.3065	3.2727	1.5463

Table 12

4. Discussion and Conclusion

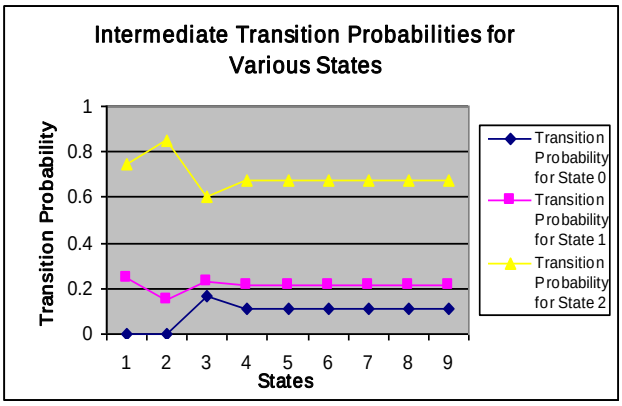
The present simulator computes the n-step probabilities successively until the system reaches its steady-state. Given the initial probability of being in state i for i = 0,1, …., N, our simulator prints out the probability of being in state i for i = 0,1,2, ….., N after n = 1,2, …. steps.
If steady-state probabilities exist, the expected first-passage and recurrence times can be computed. When each state can be reached from every state of the system, the expected first-passage time from any given state to any state can be expressed as a random sample from a specific probability distribution function. In steady-state, the probability of being in any given state is a constant, regardless of the starting state at step 0.
The Graph 1 depicts the relationship between number of states and the intermediate transition probabilities (for Case 1). It is found that the software reaches its steady state probability in eight steps (n=8).



Graph 1

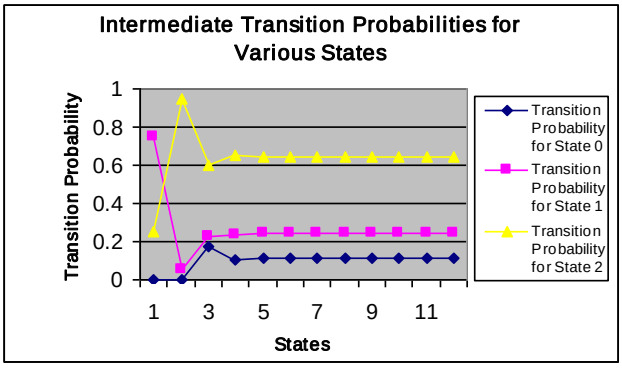
The Graph 2 depicts the relationship between number of states and the intermediate transition probabilities (for

Case 2). It is found that the software reaches its steady state probability in seven steps (n=7).



Graph 2

The Graph 3 depicts the relationship between number of states and the intermediate transition probabilities (for Case 2). It is found that the software reaches its steady state probability in twelve steps (n=12).



Graph 3

This simulator will be an asset in deciding the release policies of various versions of the software during the process of SDLC (System Development Life Cycle).

References:

[1] Gupta, D. Sadiq, M. "Software Risk Assessment and Estimation Model", Aug. 29 2008-Sept. 2 2008, page(s): 963-967 IEEE International Conference on Computer Science and Information Technology, 2008. ICCSIT '08.

[2] [2] Cortellessa, V.; Goseva-Popstojanova, K.; Kalaivani Appukkutty; Guedem, A.R.; Hassan, A.; Elnaggar, R.; Abdelmoez, W.; Ammar, H.H., "Model-Based Performance Risk Analysis", Software Engineering, IEEE Transactions, Volume 31, Issue 1, Jan. 2005 Page(s): 3 – 20.

[3] [3] Linda Westfall, The Westfall Team, PMB 383, 3000 Custer Road, Suite 270 Plano, TX 75075 "Software Risk Management".

[4] Barry W. Boehm., "Software Risk Management", IEEE Computer Society Press, 1989.

[5] Prasanta Kumar Dey, Jason Kinch, Stephen O. Ogunlana, Journal: Industrial Management & Data Systems, Year: 2007, Volume: 107, Issue: 2, Page: 284 – 303, Emerald Group Publishing Limited.

[6] Michael Barth Institut for Informatik, Ludwig-Maximilian-University Munchen, Oettingenstr 67 "Integration of Simulation Based Performance Assessment in a Software Development Process"

[7] ProSim/RA — Software Process Simulation in Support of Risk Assessment, Book: Value-Based Software engineering, Publisher: Springer Berlin Heidelberg, Pages: 263-286

[8] Nejad, H.S.; Dongfeng Zhu; Mosleh, A., "Hierarchical planning and multi-level scheduling for simulation-based probabilistic risk assessment", Simulation Conference, 2007 Winter Volume, Issue, 9-12 Dec. 2007 Page(s): 1189 – 1197.

[9] Risk assessment using software reliability simulation, Florentina Suter (University of Bucharest), Workshop on Mathematical Methodologies for Operational Risk, April 16-17-18, 2007, EURANDOM, Eindhoven, The Netherlands

[10] Gillette Billy E... "Operations Research", Tata Mc Graw Hill Publishing Company Limited, 2004.

[11] P. K. Suri and Bharat Bhushan, "Simulator for Software Maintainability", IJCSNS, Vol 7 No. 11, November 2007.

Dr. P.K. Suri received his Ph.D degree from Faculty of Engineering, Kurukshetra University, Kurukshetra, India and master's degree from Indian Institute of Technology, Roorkee (formerly known as Roorkee University), India. He is working as Professor in the Department of Computer Science & Applications, Kurukshetra University, Kurukshetra - 136119 (Haryana), India since Oct. 1993. He has earlier worked as Reader, Computer Sc. & Applications, at Bhopal University, Bhopal from 1985-90. He has supervised ten Ph.D.'s in Computer Science and eleven students are working under his supervision. He has more than 100 publications in International / National Journals and Conferences. He is recipient of 'THE GEORGE OOMAN MEMORIAL PRIZE' for the year 1991-92 and a RESEARCH AWARD –"The Certificate of Merit – 2000" for the paper entitled ESMD – An Expert System for Medical Diagnosis from INSTITUTION OF ENGINEERS, INDIA. His teaching and research activities include Simulation and Modeling, SQA, Software Reliability, Software testing & Software Engineering processes , Temporal Databases, Ad hoc Networks, Grid Computing , and Biomechanics.

Dr. Bharat Bhushan received his Ph.D degree from Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, M.Sc (Physics) from Punjab University Chandigarh and M.Sc (Comp. Sc.), MCA degrees from Guru Jambeshwar University, Hissar in 2001 respectively. Presently working as Head, Department of Computer Science and Applications, Guru Nanak Khalsa College, Yamuna Nagar (affiliated to Kurukshetra University, Kurukshetra- Haryana, India) and senior most teacher of computer science in Haryana since 1984. He is a member of Board of Studies of Computer Science, Kurukshetra University. His research interest includes Software Engineering, Digital electronics and Simulation Experiments.

Ashish Jolly received his MCA degree from University of Madras, Chennai in the year 1999. Currently he is pursuing Ph.D in Computer Science from Department of Computer Science & Applicatiions, Kurukshetra University, Kurukshetra, India. He is working as a Asstt Professor and Head in the Department of Computer Science & Applications, Shri Atmanand Jain Institute of Management & Technology (affiliated to Kurukshetra University, Kurukshetra), Ambala City, Haryana, India. His research area includes Simulation, Software Engineering and Software Project Management.