5th International Conference on Experiments/Process/System Modeling/Simulation/Optimization
5th IC-EpsMsO
Athens, 3-6 July, 2013

A SIMULATION APPROACH TO EVALUATE THE QUALITY AND THE PERFORMANCE OF A SOFTWARE REQUIREMENTS ENGINEERING PROCESS

Kyriaki D. Tsilika[1], and Vassilis C. Gerogiannis[2]

[1]University of Thessaly, Department of Economics
Korai 43, 38333, Volos, Greece
e-mail: ktsilika@uth.gr

[2]Technological Education Institute of Larissa, Department of Project Management
Larissa, 41110, Greece
e-mail: gerogian@teilar.gr

Keywords: Stochastic Simulation, Software Process Development, Software Requirements Engineering, Requirements Prioritization

Abstract. *Requirements engineering in software development includes, among other activities, the process of evaluating, prioritizing and selecting the requirements which will be implemented in the next release of a software product. In particular, in case of market driven software development, the various requirements issued from the market (potential customers of the software) are examined through evaluation activities to determine which requirements are valuable and cost effective to be implemented in the next software release. One practical approach often applied in this process is the so called "requirements triage" that results in a classification of requirements into different priority classes. In this paper, we present a re-entrant queuing model to analyze, through simulation, the decision quality and the performance of such a process. The presented model considers that requirements arrive following a Poisson distribution and they are channeled into dedicated servers (i.e., requirements analysts) which evaluate them by applying an initial screening and then a more detailed evaluation activity. In comparison with other relevant approaches in the literature, the proposed model is more enhanced since it includes structural elements, such as different service rates for servers (i.e., analysts may have different productivity rates), re-entrant lines and garbage collectors for requirements as well as the option that the initial classification of a requirement might be re-evaluated and altered during the process.*

1 INTRODUCTION

This paper presents a re-entrant queuing model [4] to analyze, through simulation, the decision quality and the performance of the so called "requirements triage" process [2]. In software requirements engineering, requirements triage is a process that has its origins from the medical treatment field and it is usually applied to evaluate which requirements of a software product will be finally implemented in the next software release, given the time and resources available. The process follows some sequential phases (i.e., requirements screening, evaluation and construction) and finally results in classifying the software requirements into different priority classes. In particular, we assume that a requirements triage process classifies all arriving requirements into requirements of Class A which must be realized in the next software release, requirements of Class B which need not be realized in the next software release and requirements of Class C which should be realized but further examination is required to determine if their implementation is cost-effective [7].

The paper suggests a simulation model of analyzing a requirements triage process that makes use of a symbolic simulation technique (Monte Carlo method). The model attempts to replicate the characteristics of a requirements triage process through the use of estimated probability distributions for the process parameters. The process is modeled by a re-entrant M/M/c/100/∞ queuing system [9], in which the process phases (screening, evaluation and construction) are repeated several times. In the relevant literature, there are also other approaches which apply queuing systems simulation to analyze requirements engineering processes [1], [3] [5], [6]. In comparison with these approaches, the proposed model is more enhanced since it considers different service/productivity rates for the servers (i.e., requirements analysts) which undertake the process phases. In the suggested model there are also re-entrant lines which feed certain requirements back in the process for further evaluation. This characteristic allows to re-evaluate the initial classification and prioritization of requirements. The model finally includes garbage collectors for collecting those (unimportant and cost-effective) requirements which need not be realized in the next software release.

The suggested stochastic simulation model can be used for deriving information about the requirements flow in a software requirements engineering workflow and based on estimations of the rates of requirements arrivals, the simulation can be applied to calculate: i) the number of requirements being balked in the process, ii) the number of requirements being processed by every server (i.e., every requirements analyst) in the process and iii) the number of finished (analyzed) requirements. The model also calculates the average waiting and execution times spent by any requirement in the process. An important goal of the suggested analysis is to test the effectiveness, the quality and the performance of the requirements triage process by computing useful metrics. These metrics are: (i) the proportion of "erroneous" evaluation decisions performed by the requirements analysts, (ii) the proportion of requirements driven to garbage collectors and (iii) estimates for the process delays due to rework and analysts' work overload. The simulation is implemented in the Queuing System Simulator of WINQSB software package. The WINQSB simulator provides a way to perform an early analysis for the process efficiency. Finally, WINQSB offers facilities for the visualization of all computed effectiveness / performance process measures.

2 THE SIMULATION MODEL

In the current section of the paper we simulate the requirements flow in a market-driven software development process, starting from Screening (S) through Evaluation (E) phase, using a methodological approach based on queuing theory. The proposed model assumes that, if perfect information about a requirement was available, it would be possible to examine, in advance, if it is a good decision to include or not a requirement in the next software product release. In particular, the requirements arriving at the process are assumed to be classified into three priority classes, namely A, B and C class. Class A includes important requirements which (under perfect information) must be developed in the next software release; Class B contains unimportant requirements which (under perfect information) need not be realized in the next software release; Class C includes requirements which (under perfect information) are considered of medium importance. Simulation results can be used to examine the quality of the decisions (i.e., the proportion of correct / erroneous decisions) taken during the requirements triage process by computing the number of evaluation decisions which are not consistent with the initial classification of the incoming requirements into the three priority classes presented before.

The three requirements classes are associated with corresponding requirements arrival populations. Requirements are assumed to enter the process with Poisson entrance and they are channeled into dedicated servers (requirements analysts) of the Screening phase of the process. Screening servers (requirements analysts) either direct the requirements into dedicated servers (other analysts) of the Evaluation stage (after re-evaluating their classification) or put the requirements on hold for a second decision/glance. The screening phase may also decide for a requirement to exit the queuing system without finishing the designed service sequence. The screening servers' evaluation is characterized by using probabilities values which denote the connections to the following nodes (i.e., an analyst could change the initial requirement's class). Requirements driven to the servers (analysts) working at the Evaluation phase form queues of finite capacity which follow also a FIFO discipline. Requirements which need rework form pseudo queues which feedback requirements to the servers of the Screening phase. The rework flow is modeled as a loop connecting each Screening server and a rework buffer. Service times are assumed to be exponentially distributed with different service (productivity) rates per server (analyst).

2.1 Model Structure

Figure 1 depicts the re-entrant line queuing model under study. The model consists of four types of elements: (i) requirements of classes A, B and C, (ii) servers working at the screening phase (i.e., servers AS, BS, CS) and servers working at the evaluation phase (i.e., servers AE, BE, CE), (iii) outlets/garbage collectors (i.e., GA, GB, GC) which collect all requirements rejected from corresponding servers (i.e., servers AS, BS, CS, respectively) and (iv) finite-capacity queues of requirements. The three requirements populations (A, B and C) arrive to the servers of the Screening phase with different arrival rates and they are queued in a FIFO discipline. Servers AS, BS, CS use an output rule based on probabilities attached to their output connections. Requirements enter the next phase according to these estimated probabilities. Requirements which require rework form pseudo queues which return back requirements to the servers of the previous phase. The queues formed by the requirements before and after the Screening phase follow a FIFO discipline with a finite capacity of 100 units. Thus, the presented queuing model consists of 21 components: 3 requirement sources, 9 queues, 3 garbage collectors (that is an outlet for requirements to exit the queuing system without to be considered for further development) and 6 servers (analysts).

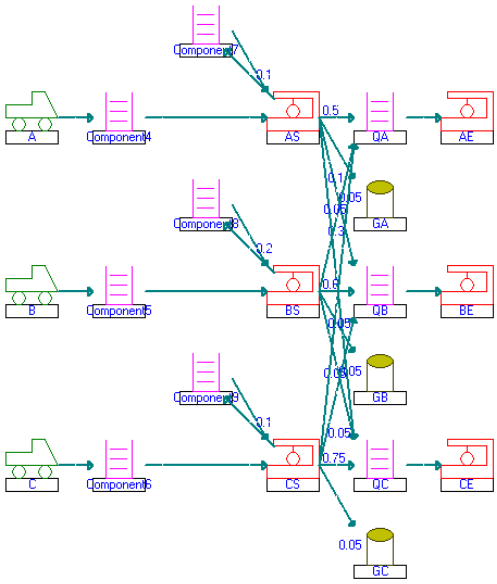

Figure 1: The process model from Screening through Evaluation phase depicted in QSS simulator

2.2 Probability Distributions and Parameters Estimation

The choice of adopting the exponential distribution for modeling requirements interarrival and service times is also made in other studies in the relevant literature [1], [5], [6]. For example, Antoniol et al.[1] suggested an exponential distribution when the square of the coefficient of variation, defined as the ratio of the standard deviation to the mean, takes values between 0.7 and 1.3. They also performed foodness-fit-tests to assess the similarity of real data distributions with exponential distributions.

In the current approach, we follow a hypothesis that was also adopted in [8]. In particular, the basic parameters of a queuing system that describes a software development process can be estimated either by analogy based on past projects or by using judgment expressed by experts (experienced software professionals within a software development organization). Of course these estimates can be, at a certain degree, imprecise, however, as more experience is gained by repeating the same steps of a software development process, more refined estimates can be obtained. The values of the parameters considered in the experimental model are, to some extent, time dependent and case sensitive. The values of the parameters λ_A, λ_B, λ_C stand for the rates that the development organization receives from the market requirements. These rates correspond respectively to the arrival rates of customer (requirement) classes A, B and C in the simulation model. The value of the service rates (i.e., the values of the parameters μ_{AS}, μ_{BS}, μ_{CS}, μ_{AE}, μ_{BE} and μ_{CE}) are subject to variations of the service (analysts) team size and team composition (expert and novice analysts). We used parameter estimates by considering also that a screening phase is performed more rapidly than the subsequent evaluation phase. The assumptions for the model parameter are consistent with the example cases presented in [8].

	λ_A	λ_B	λ_C	μ_{AS}	μ_{BS}	μ_{CS}	μ_{AE}	μ_{BE}	μ_{CE}
A	5	3.33	4	6.67	4	5	4	2	3.33
B							3.33	1	2
C							2	0.5	1

Table 1. Parameter assumptions (requirements rates per day)

Probability attached to the connection (from / to)	Rework buffer	Rework buffer	Rework buffer	QA	QB	QC	GA	GB	GC
AS	0.1			0.5	0.3	0.05	0.05		
BS		0.2		0.1	0.6	0.05		0.05	
CS			0.1	0.05	0.05	0.75			0.05

Table 2. Probabilities concerning servers' classification criteria

3 SIMULATION ANALYSIS OF THE REQUIREMENTS TRIAGE PROCESS

In the experimental model, we considered a requirements triage process of 100 man-days. Applying the data entry parameters of Table 1 and Table 2, 30 simulations were carried out for 100 man-days each. Based on the specified random seed (Figure 2), 30 similar instances of the requirements triage process were simulated.

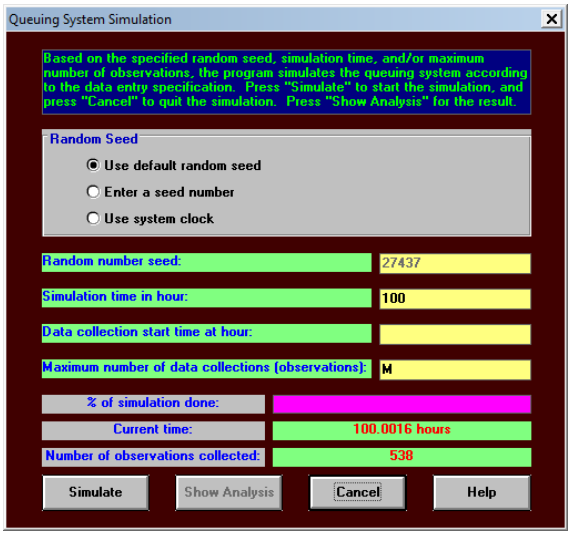

Figure 2. Specification of random number sequence

3.1 Simulation Results

The simulation provided the following prompt results per simulation (Figures 3, 4 and 5).

05-25-2013	Server Name	Server Utilization	Average Process Time	Std. Dev. Process Time	Maximum Process Time	Blocked Percentage	# Customers Processed
1	AS	21.98%	0.0406	0.0440	0.3226	0.00%	541
2	BS	20.19%	0.0582	0.1046	0.8527	0.00%	347
3	CS	19.81%	0.0416	0.0428	0.2791	0.00%	476
4	AE	11.07%	0.0383	0.0392	0.3229	0.00%	289
5	BE	4.47%	0.0399	0.0430	0.2233	0.00%	112
6	CE	8.34%	0.0446	0.0462	0.3126	0.00%	187
	Overall	14.31%	0.0440	0.0593	0.8527	0.00%	1952
Data	Collection:	0 to	100		CPU	Seconds =	28.6280

Figure 3. Results from analyzing servers' performances

05-25-2013	Result	A	B	C	Overall
1	Total Number of Arrival	461	337	375	1173
2	Total Number of Balking	0	32	14	46
3	Average Number in the System (L)	160.9994	-20.4788	135.8674	276.3880
4	Maximum Number in the System	306	15	258	579
5	Current Number in the System	306	-86	258	478
6	Number Finished	133	364	89	586
7	Average Process Time	0.5323	0.6296	0.5244	0.5915
8	Std. Dev. of Process Time	0.7116	1.0311	0.6574	0.9184
9	Average Waiting Time (Wq)	0.0197	0.0232	0.0136	0.0209
10	Std. Dev. of Waiting Time	0.0377	0.0507	0.0318	0.0457
11	Average Transfer Time	0	0	0	0
12	Std. Dev. of Transfer Time	0	0	0	0
13	Average Flow Time (W)	0.1289	0.1392	0.1298	0.1354
14	Std. Dev. of Flow Time	0.2506	0.1986	0.1480	0.2052
15	Maximum Flow Time	2.8221	2.8587	0.8872	2.8587
	Data Collection: 0 to	100			
	CPU Seconds =	28.6280			

Figure 4. Results from analyzing customers (requirements) data

05-25-2013	Garbage Collector	# Customers Collected	Average Flow Time	Std. Dev. Flow Time	Maximum Flow Time	Average Process Time	Std. Dev. Process Time
1	GA	27	0.1112	0.1157	0.4998	0.4994	0.9883
2	GB	10	0.9302	1.8049	5.7350	1.0029	1.7837
3	GC	27	0.1433	0.1228	0.5459	0.6080	0.8155
	Overall	64	16.1732	42.8877	5.7350	39.9288	102.9207
Data	Collection:	0 to	100		CPU	Seconds =	28.6280

Figure 5. Results from analyzing the garbage collector of requirements (customers)

3.2 Process Measures

The statistical analysis of 30 simulated average service times and average waiting times, results in the following graphical comparison (Figure 6A and Figure 6B).

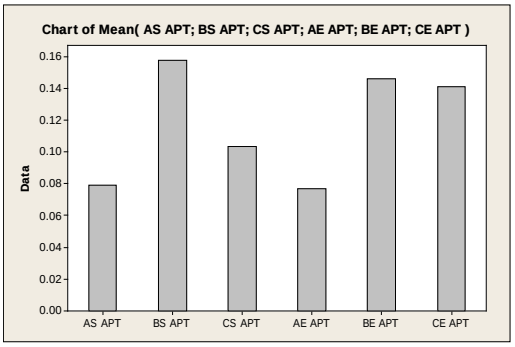

Figure 6A. Mean values of Average Process Time per analyst

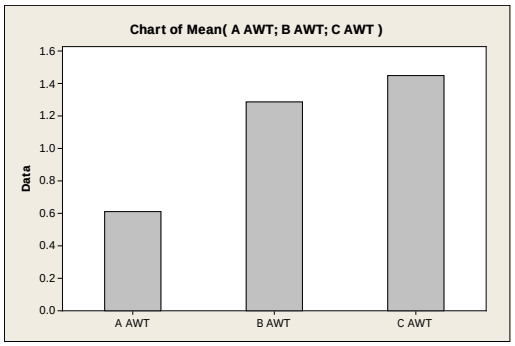

Figure 6B. Mean values of Average Waiting Time per requirement

The model effectiveness is measured using four performance measures: (i) the percentages of "erroneous" decisions by noting down requirements of class A (or B or C) which are not assigned at the same class as they follow in evaluation phase, (ii) the percentages of requirements driven to garbage collectors, (iii) the number of reworked requirements and (iv) the servers' utilization coefficients. These measures are defined as follows:

- Number of requirements which enter and exit the queuing system= Number of A (or B or C) requirements finished (line 6 in Requirement/Customer Analysis) + Number of requirements driven to garbage collectors GA (or GB or GC)
- Percentages of "erroneous" decisions = | Number of A (or B or C) requirements finished (line 6 in Customer Analysis) + Number of requirements driven to garbage collectors GA (or GB or GC) (Garbage Collector Analysis) - Number of requirements processed by AE (or BE or CE) | / Number of requirements which enter and exit the queuing system
- Percentages of requirements driven to garbage collectors= Number of requirements driven to garbage collectors GA (or GB or GC) / Number of requirements which enter and exit the queuing system
- Number of requirements being reworked = Number of requirements processed by Screening servers - Number of requirements processed by Evaluation servers – Number of requirements driven to garbage collectors

For each one of the performance measures, 95% confidence interval estimations are given based on the simulation results:

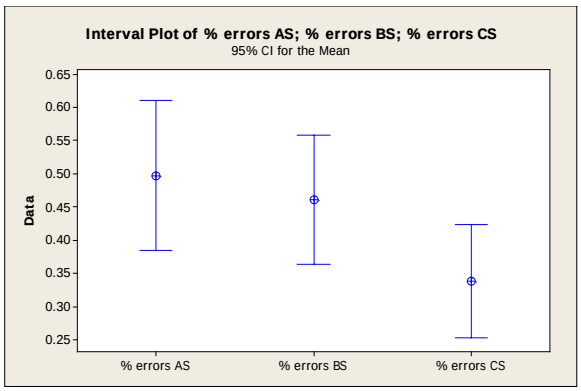

Figure 7A. 95% CI for the mean of percentages of

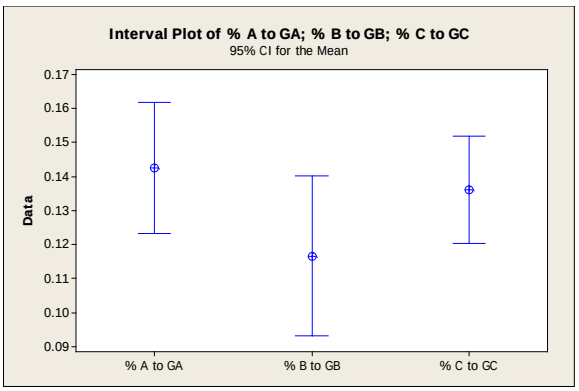

Figure 7B. 95% CI for the mean of percentages driven

"erroneous" decisions

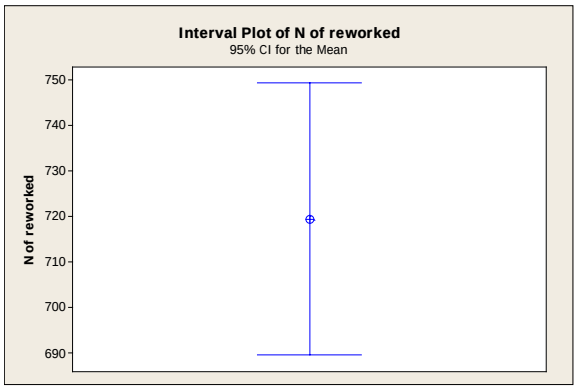

Figure 7C. 95% CI for the mean of reworked requirements

to garbage collectors

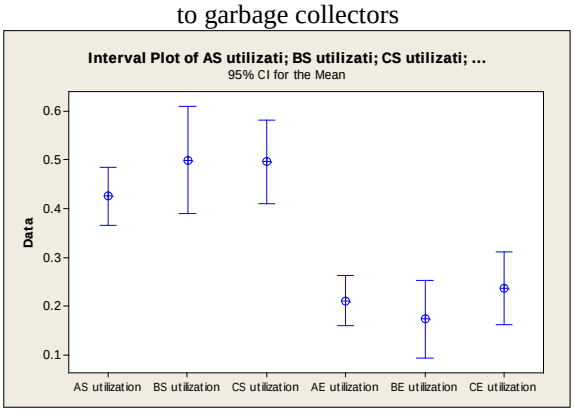

Figure 7D. 95% CI for the mean of servers' utilization

4 POTENTIAL USEFULNESS OF THE SIMULATION RESULTS

The usage of an objective and quantitative evaluation method can be helpful to evaluate a software requirements engineering process such as a requirements triage process. Instead of using a trial and error method, a simulation approach can estimate the impact of variations of λ, μ and decision making probabilities to the process completion and waiting times, the length of requirements queues and the number of erroneous evaluation decisions. For different sets of parameter values, the preferred number of simulations is carried out to result in a statistical estimation of the process measures. Simulation can consider different productivity rates caused by changes in team size or composition, adoption of different development methodologies, and changes of relative costs. A project manager could then check whether or to which extend increasing the number or the qualification of analysts can influence the process quality/performance measures.

5 CONCLUSIONS

The objective of this study is to quantitatively evaluate the phases of a requirements triage process in a software product software development. To achieve this objective, a simulation approach based on queuing network performance model was proposed. The attributes of the model were consistent to ones reported by other authors in cases of actual software product releases. The analysis was based on a symbolic stochastic (Monte Carlo) simulation. The model activities were simulated for 30 periods of 100 days each and simulation results were stored in a simulation database. Statistical analysis of the simulated results resulted in descriptive statistics of the process performance measures, confidence intervals for the mean of "erroneous" evaluation decisions, the mean number of requirements driven to garbage collectors, the mean number of reworked requirements and the mean value of servers' utilization (requirements analysts' effort). Our stochastic approach could be exploited in a predictive fashion as well, by changing the input parameters values at every performance of the simulation, to foresee the impact of possible process changes prior to the deployment of a process in an actual setting of a software project.

ACKNOWLEDGMENTS

This research has been partially supported by the European Regional Development Fund and Greek National Funds in the context of the research project "SPRINT SMEs", Act: "ARCHIMEDES III" in the Operational Program "Education and Lifelong Learning 2007-13".

REFERENCES

[1] Antoniol, G., Cimitile, A., Di Lucca, G. A. and Di Penta, M., (2004), "Assessing Staffing Needs for a Software Maintenance Project through Queuing Simulation", IEEE Transactions on Software Engineering, Vol. 30(1), pp. 43-58.

[2] Davis, A. M., (2003), "The Art of Requirements Triage", IEEE Computer, Vol. 36(3), pp. 42-49.

[3] Donzelli, P. A. (2006), "Decision Support System for Software Project Management", IEEE Software, Vol. 23(4), pp. 67-75.

[4] Gross, D. and Harris, C. (1998), *Foundamentals of Queuing Theory*, Wiley & Sons.

[5] Höst, M., Regnell, B., Nattoch, Dag J., Nedstam, J. and Nyberg, C. (2001), "Exploring Bottlenecks in Market-driven Requirements Management Processes with Discrete Event Simulation", The Journal of Systems and Software, Vol. 59(3), pp. 323-332.

[6] Höst, M., Regnell, B. and Tingström, C. (2008), "A Framework for Simulation of Requirements Engineering Processes1", *Proceedings of 34th Euromicro Conference Software Engineering and Advanced Applications*, IEEE, pp. 183-190.

[7] Khurum, M., Gorschek, T., Angelis, L. and Feldt, R. (2009), "A Controlled Experiment of a Method for Early Requirements Triage Utilizing Product Strategies", *Proceedings of 15th Working Conference on Requirements Engineering - Foundations for Software Quality*, Springer, pp. 22-36.

[8] Regnell, B., Karlsson, L. and Höst, M. (2003), "An Analytical Model for Requirements Selection Quality Evaluation in Product Software Development", *Proceedings of 11th IEEE International Requirements Engineering Conference*, IEEE, pp. 254-263.

[9] Sarma, V. V. S. and Vijay Rao, D. (1997), "A Re-entrant Line Model for Software Product Testing", Sadana, Vol 22(1), Springer India, pp. 121-132.