QUEUEING MODELS FOR COMPUTING SYSTEMS MANAGEMENT AND MAINTENANCE

MADSEN, Henrik
DTU Informatics, Denmark
hm@imm.dtu.dk

POPENȚIU-VLĂDICESCU, Florin
University of Oradea, UNESCO IT Chair
popentiu@imm.dtu.dk

ALBEANU, Grigore
Spiru Haret University, Bucharest, Romania
g.albeanu.mi@spiruharet.ro

Abstract

Software development is a special activity typically carried out by individuals that work best on their own. Recently, it is full accepted that software products require documentation, testing, quality assurance and a complete understanding of customer requirements. Such activities are better carried under an efficient software management and by maintenance actions which do not increase the overall system cost. This paper presents some queueing models, successfully used for hardware service systems, in order to provide appropriate tools in the decision process related to resource allocation for developing, testing, and upgrading or maintaining the software under current exploitation.

Keywords: *queueing models, software management*

AMS classification: 62P30

1. **Introduction**

Software predicting quality and reliability can be, especially at this moment, a quite difficult job. According to [1], "the increasing software complexity and the cost and time constraints against software reliability maximization for the most software projects have made mandatory the ussage of a standardized approach in producing such items". In this context, for achieving a high capability maturity level, more actions must be taken for the improving the management process. What kind of actions? What is the mathematical base? It is clear that modeling science is called to help.

From a practical point of view, hardware quality evaluation is relatively well incorporated in the design process whereas software quality evaluation is limited even though early and recent published work shows the continuous growing of the software cost in the total system cost. The optimal resource allocation during software developing and maintenance period, in order to minimize the software cost. The activities of the software maintenance, from the quality assurance point of view, include the analysing fault reports and enhancement features, verifying and fixing defects after delivery of a software product to customers. Mainly, the software quality is measured by counting the faults or designing defects found in one of the software components (modules). Such an approach helps only the software developers, not the managers. From the manager point of view some questions are important enough: How many professionals in software developing are needed? How large must be the testing team? How large will be the maintenance team after delivering to the customers? A simulation model could help the manager to find the answer. A useful strategy based on queueing theory, in the hardware maintenance context, was reported in [2]. Different successfully application areas of queueing theory could be found in [3], [4], [5] and [7], to mention only some references on the subject. Recently a queueing approach is proposed in the software case [6]. Unfortunately things are more complex, and the modeling science must provide more.

This paper considers some queueing models with immediate application to help the management and maintenance teams in their work. The material introduces the fundamental terminology and notations, in the next section, while the third section describes a general fixing time distribution model. Concluding remarks review the advantages and drawbacks of the queueing approach for the software field.

2. **General remarks**

Queues are common entities in our economical, technological and social life. The elements of a queueing system are: "customers" from a "population" or source which enter to receive some type of services, and the "service facility" having one or more servers. The word "customer" is used in generic form. In the framework of software management and maintenance, a customer could be a fault report. The mission of the service facility will be the removing (fixing) the reported faults. The basic protocol asks the developer to take the faults from the queue according to the queue principle in order to investigate and fix them. In this interpretation, the queue consists in a database whose records are individual fault reports, while the service facility is represented by the maintenance team or the validation team.

In any type of system modeled as a queueing system some trade-offs can be considered. One aspect concerns the situation when the service facility has a large capacity that queues rarely form. In this case some unused capacities there exists. When the servers are always busy, the length of the queue will increase and a large time interval will be necessary to remove all the faults.

In the following, standard notations which are common in the context of the queueing systems are presented. Let c be the number of identical servers, or the number of individuals in the maintenance team. If λ is the average arrival rate of fault reports and μ is the average service rate per server for fixing bugs (removing faults), then the fraction $\lambda/(c\mu)$ describes the server utilization, that means the fraction of time the maintenance team is busy. Let also N(t) be the random variable describing the number of fault reports in the system at time t, $N_q(t)$ be the random variable giving the number of reports waiting in the queue at time t, and $N_s(t)$ be the random variable describing the number of bugs in fixing state at time t. When the system meets the steady-state conditions then N, N_q and N_s there are random variables describing the corresponding number of items. Obviously $N(t) = N_q(t) + N_s(t)$ and $N = N_q + N_s$. If L is the expected steady state number of fault reports in the system (E[N]), L_q is the expected steady state number of fault reports in the queue, not including those in fixing process ($E[N_q]$) and $L_s = E[N_s]$, then $L = L_q + L_s$. Similarly, for the random variable s describing the fixing time, for the random variable τ describing interarrival time and the random variable w describing the total time a fault report spends in the system (w = q + s, where q is the time a fault report spends in the waiting line before start the fixing), if W = E[w], $W_q = E[q]$, $W_s = E[s]$ then $W = W_q + W_s$. A common distribution function appropriate for a random service is the exponential distribution: $W_s(t) = P[s \le t] = 1 - \exp(-\mu t)$, with μ being the average fixing rate. Other good fixing distributions are Erlang-k, constant and hyperexponential, the last one appropriate when the variance is large enough relative to the mean. The squared coefficient of variation ($\xi_x = Var[X]/E[X]^2$ is a useful parameter to measure the character of probability distributions used above. For constant variables $\xi_x = 0$; in the case of the exponential distribution $\xi_x = 1$, and if $\xi_x = 1/k$ (k integer) then the distribution is Erlang-k. For the hyperexponential distribution $\xi_x > 1$.

Some queueing models assumes only a single server, that is only one team or one developer, others are based on c identical servers. However, every queueing model has a standard structure and will be described using the Kendall notation: A/B/c/K/m/D, where A describes the interarrival time distribution, B is the service time distribution, c is the number of servers, K is the system capacity (maximum number of fault reports simultaneously allowed), m is the source size, in general difficult to estimate, and D is the queue discipline, mainly in use the FCFS (First Come First Served) strategy. The most general assumptions state that the system has a general independent interarrival time (GI) and is based on a general service time distribution (G). However, a simplified version can be used. The notation M/M/1 describes a queueing model based on one server, exponential interarrival time and exponential service time distribution.

Each developer's service rate is estimated based on his capabilities, skills, the professional experience and based on the characteristics of the product under service. If a team is composed by n members and μ_i is the full rate of fixing faults by the *ith* developer and p_i is the proportional ratio of time spends in the fixing process, the fixing fault of a whole team is $\mu = \sum_{i=0}^{n} \mu_i p_i$. However, we must differentiate between a team with n members (viewed like one server) and c servers (c teams).

One fundamental measure of queueing system performance is the traffic intensity $u = E[s]/E[\tau]$. Since $\lambda = 1/E[\tau]$ and $\mu = 1/E[s]$ the traffic intensity can also be written as $\lambda E[s]$ or λ/μ. The server utilization is given by $\rho = u/c$. According to Little [3], the following formulas hold: $L = \lambda W$, and $L_q = \lambda W_q$. Another useful measure of a queueing system performance is the probability that there are n fault reports in the queueing system, denoted by p_n.

For the basic queueing model M/M/1, with parameters λ and μ, $\rho = \lambda/\mu$, the following performance indicators are available:

$p_n = P[N = n] = (1 - \rho)\rho^n$, n = 0, 1, 2, …;

$L = E[N] = \rho/(1-\rho)$,
$W = E[w] = L/\lambda = E[s]/(1-\rho)$,
$W_q = W - E[s] = \rho E[s]/(1-\rho)$,
$L_q = \rho^2/(1-\rho)$ and $p_n = (1-\lambda/\mu)(\lambda/\mu)^n$.

If $\rho > 1$ (the number of fault reports is higher than the number of developers) a queue overloading appears. There is only one developer or one team in the M/M/1 model. A large number of developers is necessary to reduce the fixing time for a set of faults. In the case of the M/M/c model, with parameters λ and μ ($u = \lambda/\mu$), and c identical servers then $\rho = u/c$ and

$$p_0 = \left[\sum_{n=0}^{c-1} \frac{u^n}{n!} + \frac{u^c}{c!(1-\rho)}\right]^{-1}.$$

For n from 0 to c it is valid that

$$p_n = \frac{u^n}{n!} p_0,$$

but for $n \geq c$ the value is

$$p_n = \frac{u^n}{c!c^{n-c}} p_0.$$

The primary measures of system performance L_q, Wq, W, and L are given by:

$$L_q = \frac{p_0 u^c \rho}{c!(1-\rho)^2}, \quad W_q = L_q/\lambda,\ W = W_q + 1/\mu, \text{ and } L = \lambda W.$$

Example 1. A software house receives fault reports from many testers, the arrival pattern being random (Poisson), with an average of 12 fault reports per day and has a team which provides an exponential fixing service with an average time of 30 minutes; the team is available 8 hours per day. The team utilization ratio is $\rho = 3/4$. The average time a fault report spends in the fixing department is $W = E[s]/(1-\rho) = 2h$, the average number of fault reports waiting for fixing are $L_q = 2.25$, the average number of fault reports in nonempty queues $E[N_q \mid N_q > 0] = 1/(1-\rho) = 4$. Also, the average waiting time to start fixing is $W_q = \rho E[s]/(1-\rho) = 1.5h$.

If additional fixing teams are considered, say c = 2, then the server utilization ρ is 3/8. In this case it is obtained: $p_0 = 5/11$, $L_q = 27/20$ and the average waiting time in the queue, W_q is 0.18h.

It is clear now that such models can help the software managers. If they are able to estimate the fault report rate and the fault fixing rate, after computing the performance coefficients, it is possible to decide about speeding up the fault-fixing rate using especially trained developers or increase the number of people in the developing team. The analyse presented in [6] shows that increasing the number of the developers do not improve as much as the manager wants the expected response time in order to serve optimally the clients. This agree with the requirement for continuously improving the testing/developing process in order to have an adequate capability maturity for the software testing and fixing. Different aspects on the software quality from the management point of view can be found in [1].

3. **General time distribution based models**

In the following we consider the M/G/1 queueing system, that means a queueing system which has a Poisson input process with average value λ and a general service time distribution. Different fault reports have independent fixing times.

Following [3], related to the steady state formulas for M/G/1 queueing system, and using the above notations the performance characteristics have the following expressions.

Let $p_n = P[N = n] = P$ [n fault reports in system], n = 0, 1, 2, …. Then

$p_0 = 1-\rho$,

where

$\rho = \lambda E[s]$,

P[developing team is busy] = $P[N>1] = \rho$,

$W_q = \dfrac{\lambda E[s^2]}{2(1-\rho)} = \dfrac{\rho E[s]}{1-\rho}\left(\dfrac{1+C_s^2}{2}\right)$, according to the Pollaczek's formula,

$L_q = E[N_q] = \lambda W_q = \dfrac{\rho^2}{1-\rho}\left(\dfrac{1+C_s^2}{2}\right)$, $W = W_q + E[s]$

and

$L = E[N] = \lambda W$.

When the fixing time distribution is Erlang-k, then $E[s^2] = E[s]^2(1+1/k)$. For a constant fixing rate (code D), $E[s^2] = E[s]^2$.

Example 2. Let us consider a fixing time distribution as Erlang-2 while the parameter λ is 1.4h and E[s] = 30 minutes. The server utilization is 0.7, $E[s^2] = 3/8$, Wq = 19/12, $L_q = 133/60$.

In the case of GI/M/1 model, with general independent interarrival time distribution and Poisson fixing time distribution, the steady state number of fault reports in the system, N, has the distribution $\{p_n\}$ given by

$p_0 = P[N = 0] = 1-\rho$,

$p_n = P[N = n] = \rho\, \pi_0(1- \pi_0)^n$, n = 0, 1, 2, 3, …

where π_0, the probability that an arrival fault report finds the developing team idle. According to [3], $1-\pi_0 = A^*(\mu\pi_0)$ with A*(.) being the Laplace Stieljes transform of the interarrival time.

When Erlang-2 distribution is used, π_0 as a function of ρ has the following values:

ρ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95	0.98	0.999
π_0	0.971	0.906	0.822	0.725	0.618	0.504	0.384	0.260	0.132	0.066	0.027	0.001

If X is the random variable describing the steady state number of fault reports such as a new fault report finds in the system, then $E[X] = (1-\pi_0)/ \pi_0$ and Var [X] = $(1-\pi_0)/\pi_0^2$. Other important characteristics are:

$L = E[N] = \rho/\pi_0$ with $\mathrm{Var}[N] = \dfrac{\rho(2-\pi_0-\rho)}{\pi_0^2}$;

$L_q = E[N_q] = \lambda\, W_q = \dfrac{(1-\pi_0)\rho}{\pi_0}$, $W = E[s]/\pi_0$.

If the management team considers the server utilization as input data, with π_0 already known, then analysing the system characteristics will decide about the fixing team capabilities and capacity. Only minor changes in the pattern of fixing rate or the failure rate will change significantly the system performances.

4. **Conclusions**

The queueing approach can provide valuable information for managers related to the debuging and maintaining software in a similar manner as for the hardware case. However, the queueing model parameters are difficult to estimate from input data. We can try to estimate such parameters by multilinear regression (based on different explanatory variables) or by some nonlinear regression models. However, the numerical drawbacks generated by ill-conditioned matrices appeared in the computational process make such estimation unusable.

If descriptive statistics is used and the above models are applied, the manager can obtain information related to the software testing/maintaining process and it is able to change either number of beta testers (which generate fault reports) or the size of fixing team.

References

1. Albeanu, G., Popenţiu-Vlădicescu Fl., *Total Quality for Software Engineering Management*, in Springer Reliability Engineering Handbook (H. Pham editor), Springer Verlag, 2002.
2. Cătuneanu, V. M., Popenţiu Fl., Gheorghiu M., Albeanu G., *Maintenance Strategies Using Queueing Theory*, CNETAC'88 – The 4th Conference in Electronics, Telecommunications, Automation and Computers, 1988, pag. 137-143 (in Romanian.)
3. Kleinrock, L., *Queueing Systems*, Volume I: *Theory*, Wiley, New York. 1975.
4. Kleinrock, L., *Queueing Systems*, Volume II: *Computer Applications*, Wiley, New York, 1976.
5. Lee, A. M., *Applied Queueing Theory*, The Macmillan Press Limited, London, 1966.
6. Luong B., Liu D-B, *Resource Allocation Model in Software Development*, Proc. Ann. Reliability & Maintainability Symp., 2001, pp. 213-218.
7. Mihoc Gh., Ciucu G., Muja A., *Modele matematice ale aşteptării*. Editura Academiei RSR, Bucureşti, 1973 (in Romanian.)