6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

How long shall we test - a dynamic model

Gábor Stikkel, Gábor Szederkényi

Abstract

Detailed model analysis of a software testing process is presented in this paper. The maintenance and testing effort of a software development project is modeled as a predator-prey system in the well-known Lotka-Volterra form. This modelling technique enables us to estimate the reliability of the software product. By considering the realistic assumption that only one state variable is measurable, a linear observer is designed for the on-line estimation of the residual faults in a system. The problem of model parameter estimation from measured data is also investigated. Finally, the theory is applied on real data in order to answer the question in the title.

1. Introduction

Software maintenance and testing activities consume most of software project resources. This fact motivates research in the field of planning, estimating and tracking maintenance and testing resources.

An approach for modelling maintenance and testing effort was suggested by Calzolari et.al. [2]. This model considers as prey the software faults which cause environmental needs and corrective actions. Predators are the testers or developers observing and removing the prey. The dynamical change of the number of faults in the testing process or after release shows similarities to predator-prey competition.

Similar models was introduced in the literature previously. Lehman et.al. [5],[6] used dynamic models to describe the evolution of relevant software engineering metrics. Those models were successful in describing the changing of the size of software systems among releases.

Another approach which is close to the one presented here is in [1] and [8]. The authors gave a comprehensive system dynamics model of the software development process. The outcome of their simulation can help in predictions and making decisions. On the other hand the construction of these models and the estimation of the parameters is a hard, human intensive task. As far as predator-prey like model is concerned, the parameter estimation can be automated.

2. Modelling testing effort by differential equations

The classical predator-prey model was proposed by V. Volterra and A.J. Lotka. In this model a system of two differential equations model the variation of two populations. This model was adapted to maintenance and testing activities by Calzolari et.al. [2] in the following way: corrective interventions are considered to be predating software faults and the associated effort is fed by the discovery of faults.

The result of the adaptation is two new models a linear and a nonlinear one. The dynamics generated by these models represents the effort evolution within a given release, hence when a new release is delivered the dynamics starts again with another initial values.

2.1. Proposed linear model

The linear model is defined by the following differential equations [2]:

$$\dot{x}_1 = \quad -ax_2$$

$$\dot{x}_2 = bx_1 - cx_2.$$

The first variable denotes the residual faults in the underlying software system while the second one is the testing or maintenance effort. Parameters a, b and c are positive. The first equation describes the decrease in the number of residual faults as a function of actual value of testing effort. The latter quantity can increase with a rate proportional to the available fault number. The decrease term in the second equation represents the intrinsic *mortality* of this *population*.

The modification of the classical Volterra-Lotka model was needed because the faults can not reproduce themselves. A possible system evolution is depicted in Figure 1.

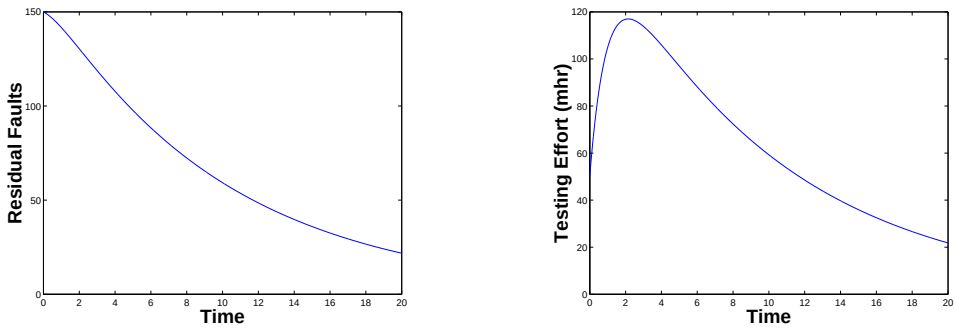

Figure 1: A possible evolution of the linear model with parameters $a = 0.1$, $b = 1$ and $c = 1.1$.

3. Observer design for the linear model

3.1. Observer theory for linear time invariant systems

The theory of linear time invariant systems is dealing with the following system of linear differential equations:

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx \tag{2}$$

where x represents the state of the system, u is the input while y is the output.

In the proposed linear model [2] the matrices are

$$A = \begin{bmatrix} 0 & -a \\ b & -c \end{bmatrix}, \qquad B = 0, \qquad C = [0 \;\; 1].$$

A method for estimating the parameters was presented in [2] for both linear and nonlinear models. In the nonlinear case we will suggest another method but for the linear case parameters are supposed to be known in the remaining.

From system theoretic point of view it is an interesting question whether the state x can be reproduced from the input u and the output y. The answer is affirmative if the system is observable [7], i.e. $y(t) \equiv 0$ implies that $x(0) = 0$. An equivalent characterization of observability of linear systems is that the kernel of the matrix

$$\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

contains only the zero vector.

The observability can be carried out by a state observer [7] which is another dynamical system of the form

$$\dot{\hat{x}} = A\hat{x} + Gu + Hy. \tag{3}$$

The system (3) is called state observer for system (1) if for all initial states $x_0, \hat{x}_0$ and for all input u

$$\lim_{t \to \infty} \hat{x}(t) - x(t) = 0.$$

3.2. Observer design

The computation of the matrices G, H can be found in [7]. It can be shown that in our special case $G = H = 0$, hence

$$\dot{\hat{x}} = A\hat{x}$$

is the state observer for system (1).

The effectiveness of the observer in applications depends on the initial condition $\hat{x}(0)$. If a project manager can guess the exact value of $x_1(0)$ then by knowing $x_2(0)$ the observer initial state can be set to $\hat{x}(0) = [x_1(0) \;\; x_2(0)]^T$ and the estimation error, $(\hat{x} - x)$ will be identically zero. It means that the manager is able to track the number of the residual faults. It is a difficult task to give an accurate estimation of $x_1(0)$. The next section two methods are suggested how to handle this problem.

3.3. Estimation methods for the initial value of the observer

The first and straightforward method is to use some optimization method. We have tested a MATLAB built-in function (`fminsearch`). A function that simulates the linear system is defined whose inputs were the five parameters of the system ($a, b, c, x_1(0)$ and $x_2(0)$). Then the error of the approximation was defined as

$$\sum_{k=1}^{N} (\tilde{x}_2(kT) - x_2(kT))^2,$$

where $\tilde{x}_2(k)$ is the observed value of effort at time instant kT, while $x_2(kT)$ is the simulated value of this variable. T denotes the sampling time. However, the results of parameter estimation was not as good as expected hence we suggest another method described in the rest of this subsection.

Denote the variation of the state variables between two observation points by

$$D_k = \begin{bmatrix} x_1(k-1) - x_1(k) \\ x_2(k-1) - x_2(k) \end{bmatrix}, \quad k = 1, \ldots, N.$$

The solution of the system of differential equations (1)-(2) implies that

$$D_k = e^{AT} D_{k-1}, \quad k = 2, \ldots, N.$$

Then $M := e^{AT}$ satisfies the following system of equations:

$$[D_2 \;\; D_3 \;\; \ldots \;\; D_N] = M \; [D_1 \;\; D_2 \;\; \ldots \;\; D_{N-1}].$$

It is well-known ([11]) that the eigenvalues of a matrix and of its exponential are in strong relation. Namely, if λ is an eigenvalue of matrix A then e^{λ} is an eigenvalue of e^A. Hence by calculating the eigenvalues of M, say λ_1^M, λ_2^M, the eigenvalues of A are $\lambda_i^A = \frac{log\lambda_i^M}{T}$, $i = 1, 2$. Considering (1)-(2), the relation between parameters of the system and eigenvalues of A we have that

$$\hat{c} = -(\lambda_1^A + \lambda_2^A)$$
$$\hat{a}\hat{b} = \lambda_1^A \lambda_2^A$$

are the estimations for the parameters of the system. The only problem is that we need to estimate a and b independently. From (1) we can estimate a by taking into account that

$$\frac{x_1(kT) - x_1((k-1)T)}{T} = -a x_2((k-1)T)$$

and all quantities in the above equation is known except a.

As far as the estimation of the initial value $x_1(0)$ is concerned the solution of the differential equation implies that $(M^{k-1} - M^k)x(0) = D_k x(0)$, $k = 1, \ldots, N$. It follows that $x(0)$ is the solution of the following equation:

$$\begin{bmatrix} I - M \\ M - M^2 \\ \vdots \\ M^{N-1} - M^N \end{bmatrix} x(0) = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_N \end{bmatrix}.$$

We had data on a case study project (see the next section for details) to test the estimation methods. The intrinsic property of the problem implies that the estimation of the number of initial faults in the software can not be known. We examined the goodness of fit of the estimation of the two state variables. The squared difference $(\sum_k (\tilde{x}_i(kT) - x_i(kT))^2)$ was used to measure the fit of the parameters. The comparison of the two approaches (Table 1) shows that the latter provides better approximation. Next section contains a figure (Figure 4) about the result of estimations. Part of our future work is to test this estimation method on more projects.

	Squared error of x_1	Squared error of x_2
Method 1	121.1	10.09
Method 2	52.1	4.87

Table 1. Comparison of parameter estimation methods.

4. Application of the linear observer to real data

Our study project was an open, standards based, modular and distributed application. The source has a length of approximately 250,000 lines of code (LOC - without comments) and was written in C++ with an effort of approximately 75,000 man-hours. We have effort data on this project (x_2 variable) from which parameters can be estimated, see Figure 4.

We also have data on faults found during testing from which we could estimate the initial value of the observer. The goodness of the observer can be tested because the number of residual faults (i.e. the observed x_1 variable) should be approximately equal to the difference of our estimated initial value ($x_1(0)$) and the number of faults found during testing.

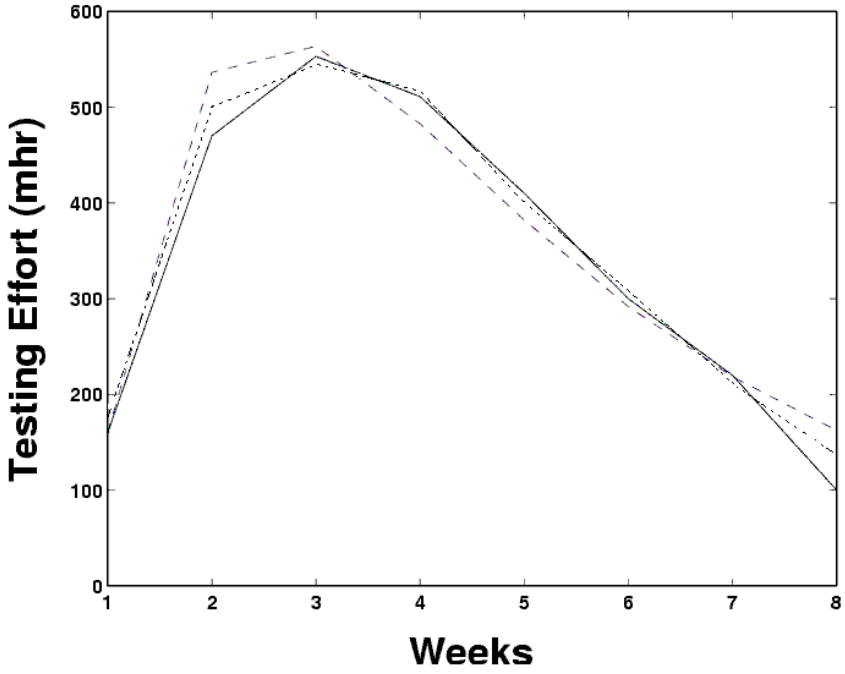

Figure 2: Effort data on the project (solid) and the testing effort given by the estimation methods (dashed - method 1; $a = 0.31$, $b = 0.9$ and $c = 1.22$; dotted - method 2; $a = 0.37$, $b = 1.2$ and $c = 1.02$).

Method 2 is used for parameter estimation in the rest of the paper. The number of faults found during testing was 848. The estimate the initial value of the residual faults is 1,010. Hence the model can be accepted if the simulation of the first variable results that there are ≈ 152 faults in the system after test. Figure 4 shows the simulation results which tells us that there are 163 residual faults in the system, hence the error of the estimation is below 10 percent.

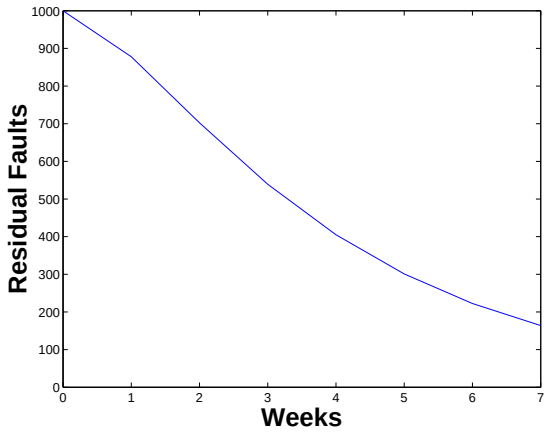

Figure 3: Simulation result of the first variable (residual faults).

Accepting this model we arrived to answer the question posed in the title of the paper. Suppose that the manager would like to continue testing until the estimated fault content of the software system is below 50. How long shall we test? Running again the simulations with the identified parameters we can depict Figure . It suggests that the testing process should continue for four more weeks to reach the specified quality. The model can also be used to answer what-if questions regarding the trade off between testing effort and software quality.

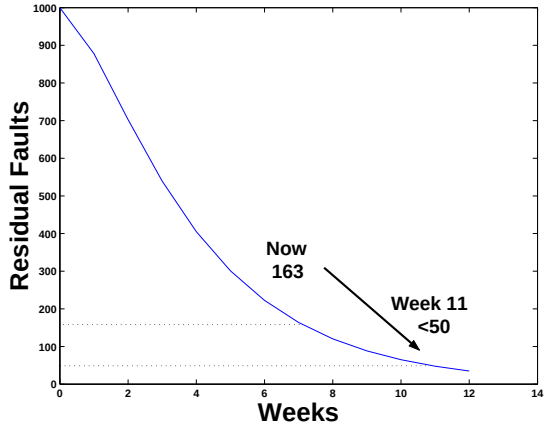

Figure 4: Simulation result of the first variable (residual faults).

5. Conclusion

Proposed maintenance and testing effort based on linear Lotka-Volterra system was revisited and was investigated from system theoretic point of view. We have found that state observer for the linear model can be used to predict residual number of faults in a software system. It can also give estimation for the manager how long the testing phase should be continued in order to reach a specified software quality. Future work will be focused on model extension and observer design for the nonlinear model.

Acknowledgements
The authors would like to thank Domonkos Asztalos and Attila Kovács for their valuable comments and for making it possible to conduct a case study in a software producing environment.

References

[1] T.K. Abdel-Hamid: The dynamics of software project staffing: a system dynamics based simulation approach. IEEE Transactions on Software Engineering, 15(2). 1989. 109-119.

[2] F. Calzolari, P. Tonella, G. Antoniol: Maintenance and testing effort modeled by linear and nonlinear dynamic systems. *Information and Software Technology*, 43(2001). 477-486.

[3] M. Grottke, K. Dussa-Zieger: Prediction of Software Failures Based on Systematic Testing. Electronic Proc. 9th European Conference on Software Testing Analysis and Review (EuroSTAR), Stockholm, 2001.

[4] A. Isidori, *Nonlinear Control Systems.* Springer–Verlag, 1995.

[5] M. M. Lehman, D.E. Perry, J.F. Ramil: Implication of evolution metrics on software maintenance. Proceedings of the International Conference on Software Maintenance, Bethesda, MD, 1998, pp. 208-217.

[6] M. M. Lehman, D.E. Perry, J.F. Ramil: On evidence supporting the feat hypothesis and the laws of of software evoution. Proceedings of the Fifth International Symposium on Software metrics, Bethesda, MD, 1998.

[7] J. M. Maciejowski: Multivariable Feedback Design. Addison-Wesley, 1989. Wokingham, U.K.

[8] R. Madachy: System dynamics modelling of an inspection-based process, Proceedings of the International Conference on Software Engineering, Berlin, 1996. pp. 376-386.

[9] D. Satoh: A Discrete Gompertz Equation and a Software Reliability Growth Model. *IEICE Transactions on Information and Systems* E83(2000) No. 7. 1508-1513.

[10] D. Satoh, S. Yamada: Parameter Estimation of Discrete Logistic Curve Models for Software Reliability Assessment *Japan Journal of Industrial and Applied Mathematics* 19(2002) No. 1. 39-53.

[11] G. Stoyan, G Takó: Numerikus módszerek I. (in Hungarian) Typotex, 1996. Budapest, Hungary.