Linear and nonlinear analysis of a testing effort model

GÁBOR STIKKEL, GÁBOR SZEDERKÉNYI

stiko@compalg.inf.elte.hu, szeder@sztaki.hu

Keywords: software maintenance and testing, state observer, parameter estimation

The maintenance and testing effort is modelled as a predator-prey model in the well-known Lotka-Volterra form in order to facilitate tracking and estimating maintenance and testing resources. Our aim is to find a solution for a crucial software development decision problem, namely for determining the end of testing and releasing the product. The method is based on tools of system theory and is applied on real project data.

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.

One of the most promising approach for modeling 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. The only difference is that faults can not reproduce new faults.

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 equations they suggest 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.

1. Modeling 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.

1.1. The 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 increas 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 *Fig. 1.*

1.2. The nonlinear model

The nonlinear model is described by the following differential system:

$$\dot{x}_1 = -h(x_1)x_2 \tag{1}$$

$$\dot{x}_2 = eh(x_1)x_2 - mx_2. \tag{2}$$

The negative term in the first equation contains $h(x1)$ and $x2$. The first term represents the *functional response of the predators(testers)*. It describes that how many faults will be found by each tester depending on the number of residual faults.

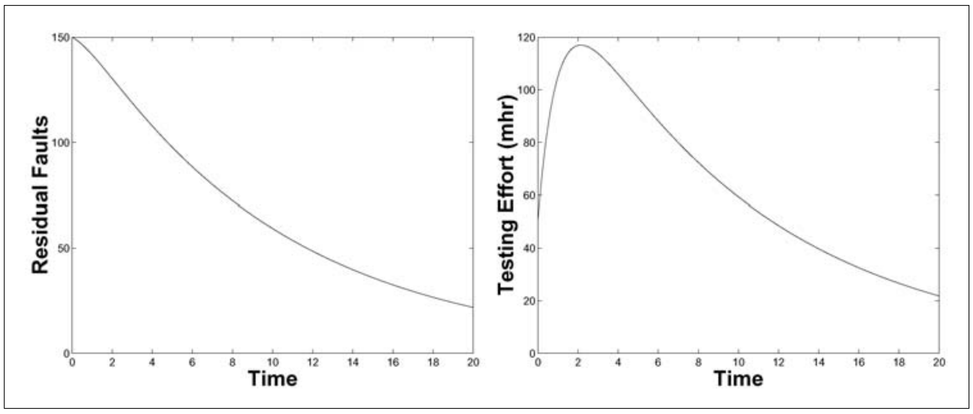


*Fig. 1.
A possible evolution of the linear model with parameters a=0.1, b=1 and c=1.1*

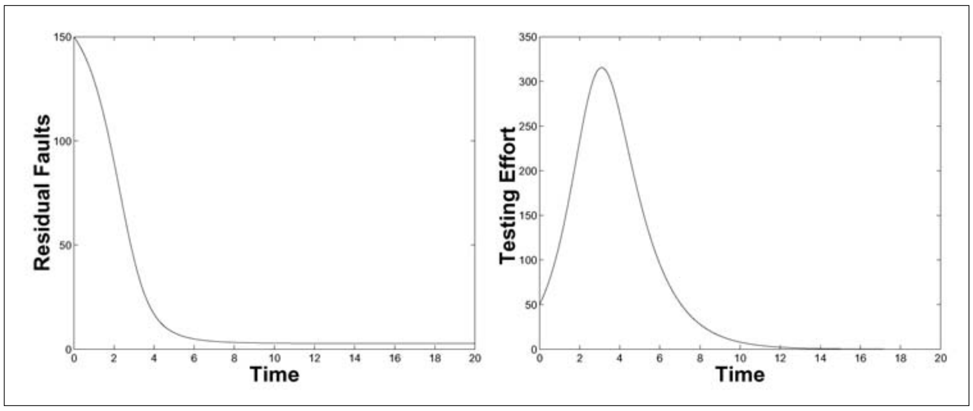


*Fig. 2.
A possible trajectory of the nonlinear model with parameters a=0.5, b=130, e=6 and m=0.7*

Here the Holling function of type 2 is used to model the functional response:

$$h(x_1) = \frac{ax_1}{b+x_1}.$$

Parameter *a* is the asymptotic fault fix rate, while *b* is the fault level for which the fault fix rate is halved. The second equation could be decomposed into two parts. Current fault fix rate is converted into new corrective effort through the *effciency* factor *e*. The second term in the second equation shows the intrinsic decrease of corrective effort over time, with *mortality* factor *m*. Possible trajectories of the two variables can be seen in *Fig. 2.*

2. Observer design for the linear model

2.1. Observer theory for linear time invariant systems

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

$$\dot{x} = Ax + Bu \tag{3}$$

$$y = Cx \tag{4}$$

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

$$\dot{\hat{x}} = A\hat{x} + Gu + Hy. \tag{5}$$

The system (5) is called state observer for system (3) if for all initial states x_0, $\hat{x}_0$ and for all input *u*

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

2.2. 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 (3).

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.

2.3. Estimation methods for the initial value of the observer

2.3.1. Expert judgements

The first proposed method is rather heuristic. It is hard to guess the exact value of the initial residual faults. Instead, one might use data from testing i.e. number of faults found until a certain time instant of the project. This number will be a lower bound on $\hat{x}(0)$ and as the project progress the manager can give more and more accurate estimates of the number of residual faults.

Suppose that initially there were 150 faults in a system. With this initial condition the model tells us that there will be approximately 21 faults remaining after the 20. day of testing. The project manager guess an initial fault containment of 100 which gives an estimation of 14 faults on the 20. day. However, after the 10th day of testing it has turned out that already 68 faults has been found. It makes the manager improve his guess to 170 which results an estimation of 24 faults after the 20. day.

2.3.2. Estimation based on software reliability growth models

The second method is based on software reliability models presented in [9] and [10]. These models are dealing with the cumulative number of faults found during testing as a function of time. Logistic [9] and Gompertz [10] differential equations and their discrete variants are fit on empirical data. The input of this procedure is a few data points i.e. number of faults found during the first days of testing. It is necessary to have at least as many data points as the fifth of the whole in order to have fairly good parameter estimation of the logistic curve [9].

By knowing the parameters we can predict the total number of faults found. These predictions can be a basis of the estimation of the initial value of the observer. The method is depicted in *Fig. 3.* Other methods for predicting residual faults and failures can be found in [3].

3. Stability analysis of the nonlinear model

A very brief stability analysis can be found in [2]. The stability of the linear model is no matter of discuss: if the parameters are positive then both eigenvalues of the system matrix have negative real parts implying the asymptotic stability of the system. The latter means that the states of the system tends to zero whatever the initial condition is. Another qualitative property of an equilibrium point is its attractiveness. Such a point is attractive if there exist a neighborhood of it so that if the system is initialized from that neighborhood than the solutions tend to the equilibrium point. Necessary attractive equilibriums should be isolated. But, as we will see the equilibriums of the nonlinear model constitute a half line, i.e. they are not attractive as was stated in [2]. This fact motivates the investigations presented in this section.

Both state in the proposed nonlinear model are supposed to have non-negative values. It can be seen that in the case when $ea - m \leq 0$ the second variable will be decreasing which is somewhat contradictory to the process observed in real life situations (testing effort is used to increase for a certain amount of time). Hence we can continue the analysis with the reasonable assumption that $ea - m > 0$.

The equilibrium points of (1) can be determined by solving the equations $\dot{x}_1^* = 0$ and $\dot{x}_2^* = 0$. The solutions are $(x_1^*, 0)$, where x_1^* is a non-negative number. These points constitute a half line which immediately implies that only local stability results make sense in this context. In order to restrict our attention to the stability analysis of the zero state, the equilibrium point is shifted to get the following equations:

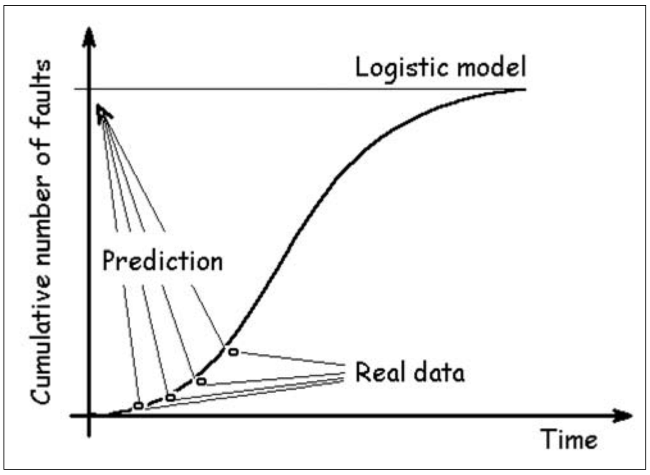
$$\dot{x}_1 = -\frac{a(x_1 + x_1^*)x_2}{b + x_1 + x_1^*} \tag{6}$$

$$\dot{x}_2 = \frac{ea(x_1 + x_1^*)x_2}{b + x_1 + x_1^*} - mx_2. \tag{7}$$

To investigate local stability of the system the Jacobian matrix of its map will be used:

$$F = \begin{bmatrix} 0 & -h(x_1^*) \\ 0 & -m + h(x_1^*). \end{bmatrix}$$

Fig. 3. Estimating initial value of the observer based on logistic reliability groowth model



Eigenvalues of the Jacobian are 0 and $-m+h(x_1^*)$. It can be seen that the condition for the second eigenvalue to be negative is $x_1^* < mb/(ea-m)$. Hence equilibrium state $(x_1^*, 0)$ is unstable if $x_1^* \geq mb/(ea-m)$.

The case of system whose Jacobian has some eigenvalue with zero real part is commonly referred to as a *critical case of the asymptotic analysis*. Center manifold theory is of great help in analyzing critical cases.

The argument presented in [4] starts with considering instead of 6, the following form:

$$\dot{y} = Ay + Pz + g(y,z) \tag{8}$$

$$\dot{z} = Bz + f(y,z), \tag{9}$$

where A is a matrix having all eigenvalues with negative real part, B is a matrix having all eigenvalues with zero real part. Then we have to solve the following partial differential equation:

$$\frac{\partial \pi}{\partial z}(Bz + f(\pi(z), z)) = A\pi(z) + Pz + g(\pi(z), z).$$

Theorem 1
The (asymptotic) stability of the zero state of

$$\zeta = B\zeta + f(\pi(\zeta), \zeta) \tag{10}$$

implies the (asymptotic) stability of (y, z) = (0, 0) *of (8).*

In our case
$A = -m + eh(x_1^*)$, $P = 0$, $g(y,z) = \frac{ea(z+x_1^*)y}{b+x_1^*+z} \quad eh(x_1^*)y$,
$B = 0$, $g(f,z) = -\frac{a(z+x_1^*)y}{b+x_1^*+z}$

and the underlying differential equation is ordinary having the trivial solution namely $\pi(z) \equiv 0$. It results that the reduced equation (10) takes the simple form of

$$\dot{\zeta} = 0$$

implying that all the equilibrium points which satisfies $0 \leq x_1^* < mb/(ea-m)$ are stable.

4. Parameter estimation

4.1. Parameter estimation of the linear model

Consider again the linear model of dynamic variation of the testing effort and residual faults. This model has three positive parameters. The problem of parameter estimation is that we have to determine the values of *a, b* and *c* from the data on the second variable.

Suppose we have data on x_2 in discrete time instances denoted by $x_2(0), \hat{x}_2(1), \ldots, \hat{x}_2(N)$. We would like to choose parameter values such that the squared error

$$\sum_{i=0}^{N}(x_2(i) - \hat{x}_2(i))^2$$

is minimal. It was carried out by creating a simulation model in MatLab and running a built-in optimization tool which uses simplex method. In order to have a fairly good estimation we need so many data points such that the peak of the testing effort curve is reached.

4.2. Nonlinear parameter estimation by algebraic elimination

As it was visible in section 2, the physical nature of the testing effort model is basically nonlinear. Moreover, linear modeling often cannot provide models of satisfactory quality for certain purposes (e.g. control). Therefore the parameter estimation of the nonlinear model structure given in (1)-(2) is carried out in this section.

4.2.1 Calculation of a nonlinear input-output model

Considering the assumption that the only measurable state variable of the nonlinear model (1)-(2) is x_2, the purpose of this section is to formulate a nonlinear inputoutput model in the following form:

$$F(y, \dot{y}, \ddot{y}, u, \dot{u}) = 0 \tag{11}$$

where *y* denotes the measurable output (i.e. $y = x_2$) and *u* is a fictitious input which is assumed to be known. An obvious and computationally simple selection for *u* is *m* in (2).

The aim is now to eliminate the non-measurable state variable x_1 from the state equations and thus to obtain a model of the form (11). For this, let us rewrite the state equations and the fictitious output equation as

$$\dot{x}_1 + \frac{ax_1x_2}{b+x_1} = 0 \tag{12}$$

$$\dot{x}_2 - \frac{eax_1x_2}{b+x_1} + ux_2 = 0 \tag{13}$$

$$y - x_2 = 0 \tag{14}$$

From (14) and (7) we get

$$\dot{y} - \frac{eax_1x_2}{b+x_1} + ux_2 = 0,$$

from which x_1 can be expressed as

$$x_1 = \frac{b(\dot{y} + ux_2)}{-\dot{y} - ux_2 + eax_2} =: \frac{n(t)}{d(t)} \tag{15}$$

using the notations *n* and *d* for the numerator and denominator of (15) respectively. Taking the time derivative of (15) gives

$$\dot{x}_1 = \frac{\dot{n}(t)}{d(t)} - \frac{n(t)\dot{d}(t)}{d^2(t)}. \tag{16}$$

Using eqs (16), (12) and the notation in (15) we can eliminate $\dot{x}_1$ from the state equations i.e.

$$-\frac{a\frac{n(t)}{d(t)}x_2}{b + \frac{n(t)}{d(t)}} = \frac{\dot{n}(t)}{d(t)} - \frac{n(t)\dot{d}(t)}{d^2(t)}, \tag{17}$$

which can be rewritten as

$$-\frac{an(t)x_2}{d(t)b + n(t)} = \dot{n}(t) - \frac{n(t)\dot{d}(t)}{d(t)}. \tag{18}$$

Computing the time derivative of *n* and *d* gives the required input-output model (that can be easily rearranged to the form (11))

$$-\frac{\dot{y} + uy}{ey} = b(\ddot{y} + u\dot{y}) - \frac{b(\dot{y} + uy)(-\ddot{y} - u\dot{y} - \dot{u}y + ea\dot{y})}{-\dot{y} - uy + eay}. \tag{19}$$

Using the fact that in our case the parameter $u = m$ is constant and thus $\dot{u} = 0$, a simpler form of the model is obtained

$$-\frac{\dot{y}+uy}{ey} = \frac{bea(y\ddot{y}-\dot{y}^2)}{-\dot{y}-uy+eay}. \tag{20}$$

Introducing the transformed parameters

$$c_1 = be^2a, \quad c_2 = ea \tag{21}$$

gives the following model

$$(\dot{y}+uy)^2 = c_1(y\ddot{y}-\dot{y}^2) + c_2y(\dot{y}+uy), \tag{22}$$

which is *linear in the new parameters*. Now, c_1 and c_2 can be estimated by any of the standard techniques, measuring only x_2.

4.2.2. Parameter estimation

For the practical implementation of the parameter estimation it's useful to calculate a discrete-time model from (22). First, let us introduce the δ operator for the notation of the numerical derivatives (calculated by using a simple Euler-approximation) as follows

$$\delta z(k) = \frac{z(k+1)-z(k)}{t_s}, \tag{23}$$

where z denotes a discrete time sequence and t_s is the sampling time.

Using (23) the input-output model (22) in discrete time is written as (24)

$$(\delta y(k)+uy(k))^2 = c_1(y(k)\cdot\delta^2y(k)-(\delta y(k))^2) + c_2y(k)(\delta y(k)+uy(k))$$

which is in a standard regression form

$$w(k) = \phi^T(k)\theta \tag{25}$$

with (26-28)

$$\begin{aligned} w(k) &= (\delta y(k)+uy(k))^2 \\ \phi^T(k) &= [y(k)\cdot\delta^2y(k)-(\delta y(k))^2 \quad y(k)(\delta y(k)+uy(k))] \\ \theta &= [c_1 \quad c_2]^T. \end{aligned}$$

Therefore c_1 and c_2 can be estimated using the well-known least squares method which minimizes the quadratic criterion

$$V(n,\theta) = \frac{1}{N}\sum_{i=1}^{N}(w(i)-\phi^T(i)\theta)^2 \tag{29}$$

with respect to θ.

Note that in this case we have to assume that one parameter from *a, b* and *e* is known, and then we can solve (21) for the remaining two unknown parameters. This is a strict limitation of the usefulness of the parameter estimation method, however it can be used in order to refine or to check the goodness of previously estimated parameters.

5. 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 *Fig. 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.

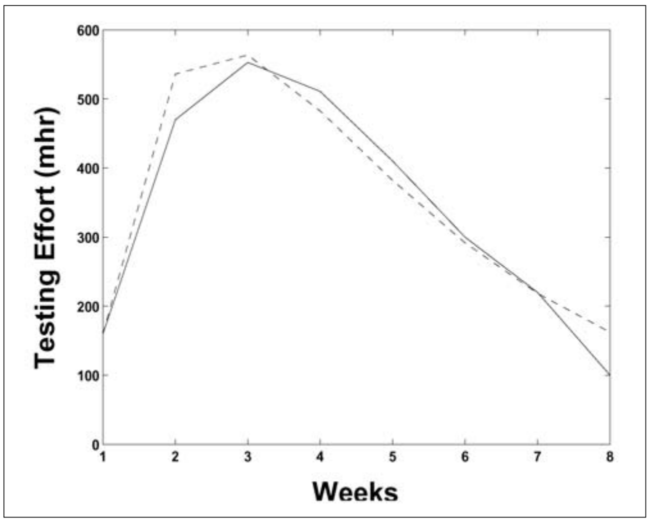
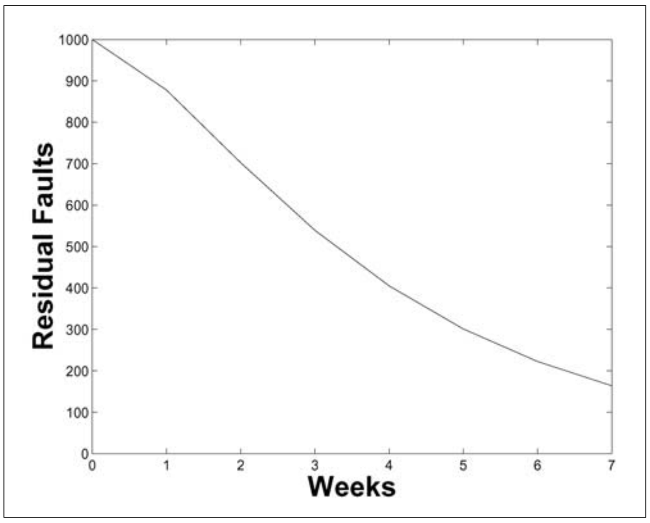


Fig. 4. Effort data on the project (solid) and the testing effort given by the estimated parameters (dashed; a = 0.31, b = 0.9 and c = 1.22)

The number of faults found during testing was 848. It makes us to estimate the initial value of the residual faults as 1,000. Hence the model can be accepted if the simulation of the first variable results that there are ≈152 faults in the system after test.

Fig. 5. 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.

Fig. 5. Simulation results (residual faults)



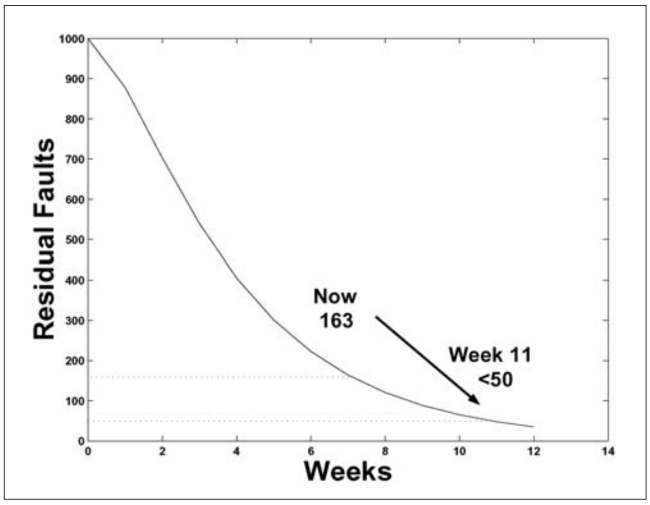


Fig. 6. Simulation result of the first variable

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 *Fig. 6.*

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.

6. Conclusion

Proposed maintenance and testing effort based on linear and nonlinear Lotka-Volterra systems 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 (Ericsson Telecommunications Hungary) and Gusztáv Bártfai (previously Ericsson Telecommunications Hungary) for project data and for valuable comments.

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, pp.109–119.

[2] F. Calzolari, P. Tonella, G. Antoniol:
Maintenance and testing effort modeled by
linear and nonlinear dynamic systems.
Inform. and Software Techn., 43(2001), pp.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 5th International Symposium on Software metrics,
Bethesda, MD, 1998.

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

[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, pp.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, pp.39–53.