

Modeling and Controlling the Software Test Process

João W. Cangussu*

Department of Computer Sciences - Purdue University

West Lafayette - IN 47907-1398 USA

(765)494-7812 cangussu@cs.purdue.edu

ABSTRACT

A novel approach for modeling and control of the software test process is presented. The approach is based on the concept of state variables and uses techniques from the well-established field of Automatic Control Theory. An initial model of the software test phase is described and the results of a case study analysis are presented.

Keywords

Software process, software testing, modeling, feedback control, state variable.

1 INTRODUCTION

The software development process (SDP) has been modeled and studied quite extensively [2]. As an alternative, we are investigating a strategy based on Feedback Control Theory [4]. Our strategy allows comparison of the output variables of the software process with one or more “set-point(s)” to determine how the inputs to a process and its internal parameters should be regulated to produce the results related to schedule, cost, effort, or other software process factors.

In this work we are considering the modeling and control of the Software Test Process (STP). A quantitative (state) model based on first principles can be used to predict an estimate of the (remaining) errors in a software product, the necessary time frame for a given reduction in errors, and strategies for improving or maintaining an error reduction schedule in the face of unanticipated disturbances. Hence, it can be used to evaluate project schedules and the allocation of resources within the test phase.

Though software process can be characterized as a natural feedback system, the use of control theory to regulate the process is not an easy task. The difficulty is due to the fact that software development is mostly a creative, not a physical process. Thus, we are unable to determine precisely or predict the behavior of a software development environment.

* Financial support by CAPES and UFMS

Despite those problems we believe that it is possible to define a model that captures the dominant behavior of the STP.

We decided to use differential equations to model each component acting on the STP. Although it appears to be difficult to capture the behavior of the software process using an analytical and continuous approach, when the behavior of elements acting on the process is modeled independently this task becomes more reasonable. The global behavior of the STP is then captured when the components acting on it are combined. The spread use of differential equations to model so many different types of systems combined with the fact that most of the models were developed using analogies to physical systems and based on assumptions similar to ours [4] further justify the approach used here. The fact of using differential equations allows the use of classical feedback control and brings with it all the established theory developed in the area, such as optimization, self-compensation, and system identification [3].

2 TEST PROCESS MODEL

Our focus here is the modeling of the test phase using differential equations as a first step in constructing a state model. Using the parameters characterizing the STP [1], we make the assumptions that lead us to the state variable model. The assumptions and the related equations are presented below.

Assumption 1: The magnitude of the rate at which the remaining errors are decreasing is proportional to the net applied effort for the test phase and inversely proportional to the software complexity ($\ddot{r} = \frac{e_n}{s_c} \Rightarrow e_n = \ddot{r} s_c$), where \ddot{r} is the second derivative of r , and e_n is the net applied effort.

Assumption 2: The magnitude of the effective test effort is proportional to the product of applied work force and the number of remaining errors, for an appropriated ζ ($e_{et} = \zeta w_f r$).

Assumption 3: The error reduction resistance is proportional to the error reduction velocity and inversely proportional to the overall quality of the test phase, for an appropriate constant ξ ($e_r = \xi \frac{1}{\gamma} \dot{r}$).

Assumptions 2 and 3 established effort applied in our model, while Assumption 1 established the resultant force balance. Hence, combining these assumptions produces the second-

order differential equation $-\zeta w_f r - \xi \frac{1}{\gamma} \dot{r} = s_c \ddot{r}$. Based on the equations presented above, we can define the following state model:

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{\zeta w_f}{s_c} & -\frac{\xi}{\gamma s_c} \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{s_c} \end{bmatrix} F_d \quad (1)$$

The force F_d represents a disturbance during the test phase, and feedback will be used to minimize this disturbance and produce the expected output in the appropriated time, if this task is feasible.

3 CASE STUDY

The case study presented in this section uses data from a large commercial project [1] underway at Razorfish, a company located in New York, NY. Razorfish currently has an application that contains about four million lines of code in COBOL. This application is to be transformed into a functionally equivalent application in SAP/R3. Razorfish is developing a tool, hereafter referred to as *transformer*, to automate this transformation. The exercise started by computing the parameters to plug into the model. The parameters s_c , γ and w_f were computed using our guidelines and in accordance with the project manager. Since an estimate of the initial number of errors was not available, data from the first weeks were used to compute it [1].

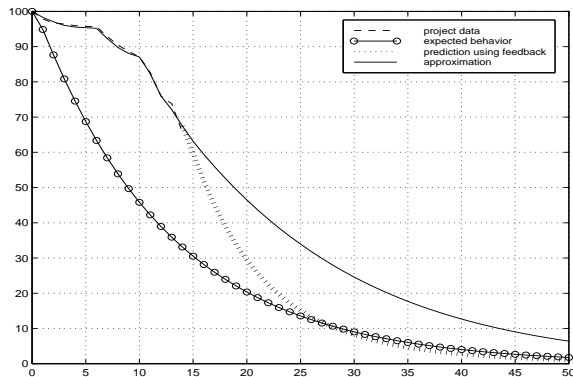


Figure 1: Behavior of the COBOL Transformer Project

In Figure 1 we can see the expected behavior for the *transformer* project plotted using the computed parameters. The observed behavior diverges from the expected one for the first 15 weeks of the project. This divergence is due to disturbances present during the STP and alternatives to correct its effect, using feedback, are discussed next.

Our model predicts that it will not be possible to finish the project by the expected deadline as one can see when comparing our approximation with the expected curve. What changes can be done in the STP in order to accomplish the deadline? Feedback will help us to answer this question. By week 15 the fraction of errors dropped to around 67.5%, and if no change is made, it will take around 35 weeks to

reach the expected level of error reduction (approximately 14.7 %). Suppose the project manager desires to achieve the same results in only 10 weeks. What “feedback” modifications are necessary? Our approach predicts an increase of 2.5 in the work force or a combination of changes in w_f and γ to achieve the project goal [1].

In general, we can conclude that our model behavior is reasonably accurate when applied to the COBOL *Transformer* project, and that feedback can be used to answer questions related to performance and cost of the STP. The validation exercise using Knuth’s T_EX78 error log report also provided accurate results.

4 PROGRESS AND FUTURE WORK

So far, two main initial goals were achieved. The first one is the development of a second order model for the software test phase. The approach used appears to be adequate to capture the essential behavior of the STP. The second goal was the analysis of the model using data from a large project. The obtained results are reasonably accurate and the piecewise approximation is shown to be a reasonable alternative to handle environmental changes in the software test process. Two other important achievements in the work are the successful application of feedback control to direct changes in the process and the use of control theory techniques to estimate some parameters of the model.

Regarding future work, the model for the test phase will be expanded to account for aspects that are not addressed yet. Learning and communications issues are two examples. We then plan to use System Identification techniques to provide a methodology to calibrate the model on a per-project/per-company basis, followed by the analysis of optimization issues.

ACKNOWLEDGEMENTS

I would like to thank Prof. Aditya P. Mathur and Prof. Ray A. DeCarlo for their help. I also would like to thank people from Razorfish, specially the project manager Omar Velasco, for their contributions.

REFERENCES

- [1] João Cangussu, Raymond DeCarlo, and Aditya Mathur. A state variable model for the software test process. In *Proceedings of 13th International Conference on Software & Systems Engineering and their Applications (ICSSEA)*, Paris-France, December 2000.
- [2] G. Cugola and C. Ghezzi. Software processes: A retrospective and a path to the future. *Software Process Improvement and Practice*, 1999.
- [3] Lennart Ljung. *System identification: theory for the user*. Prentice-Hall, Englewood Cliffs, NJ, 1987.
- [4] David G. Luenberger. *Introduction to Dynamic Systems: theory, models and applications*. John Wiley & Sons, 1979.