

Integrating Statistical and Feedback Process Control for the Monitoring of the Software Test Process

João W. Cangussu
Department of Computer Science
University of Texas at Dallas
Richardson-TX, USA
email: cangussu@utdallas.edu

ABSTRACT

Statistical Process Control (SPC) and its variations have shown to be a powerful technique to predict unstable and incapable processes. However, SPC lacks the ability to quantify the necessary changes to correct deviations in the process when incapability is detected. On the other hand, a technique based on control theory has shown to be adequate to quantify changes in the test process in order to make the process converge to the desired behavior. These two approaches appear to complement each other to better monitor and control the software test process. A combination of these approaches is presented here along with simulation results demonstrating its applicability.

KEY WORDS

Software test process, state variable model, statistical process control, feedback control.

1 Introduction

When monitoring and control a software test process two questions are of special interest. First, “is the process behaving as expected?” This question has not a straightforward answer since the rate at errors are being detected and removed depends on a complex combination of many parameters. For example, the saturation effect [1] impacts the outcomes of the software test process according to the technique being used. Once the process has been detected to be in non-conformance with the desired results, the second question of interest is: “How to make corrections in the process to achieve the desired results within the specified time?” In most organizations, the experience of the test manager determines the answers for these questions.

To decrease the dependency on project manager predictability skills over the duration of a software test process, the use of control techniques such as Statistical Process Control (SPC) and the Feedback Process Control (FPC) based on the state variable model [2, 3] are of great importance. The FPC technique can identify deviations from the expected behavior and suggest changes to correct such deviations. However, changes in the Software Test Process (STP) cannot be done at any frequency and at any time. A test manager will, most likely, not make changes in

the process if a non-significant schedule slippage is detected or if the error detection rate is found to be off, for example, for just a few days. SPC can be used to determine when the deviations observed on the outcomes of the process justify changing the process. At a certain level, it can be stated that SPC presents a more appropriated process monitoring procedure to the STP than FPC. However, SPC does not have a feedback mechanism to quantify or indicate which changes are necessary to make the process converge back to the expected behavior. FPC has such ability and combined with optimization techniques such as the Linear Quadratic Regulator [4] presents a powerful alternative to the control of the STP. As presented in this paper, the combination of the monitoring aspects of SPC and the closed feedback loop solution of FPC complement each other to improve the overall controllability of the STP.

Other techniques such as simulation [5] and Software Dynamics [6] are successful in predicting the process behavior but lack controllable concepts that are inherent to SPC and FPC. Hence we restrict our attention to only SPC and FPC. Further, the combination of these two approaches is done in the context of the Software Test Process (STP), mainly because FPC techniques have not been developed for other phases of the SDP. This is not a drawback because the mechanisms of SPC and FPC used to control the STP appear to be essentially the same for other phases of the SDP.

The remainder of this paper is organized as follows. Section 2 presents a brief description of the state variable model of the software test process. A description of standard Statistical Process Control and a variation that allows its application to the monitoring of the software test process are presented in Section 3. A combination of the two approaches is shown in Section 4 and the results appear in Section 5. Finally, the concluding remarks are presented in Section 6.

2 State Model for the STP

The linear model of the STP is based upon three assumptions. These assumptions and the corresponding equations are presented below [2]. They are based on an analogy of the STP with the physical process typified by a spring-

mass-dashpot system and also in Volterra's predator-prey model [7]. A description and justification of this analogy and the choice of a linear model is outside the scope of this paper. [2]

Assumption 1: *The rate at which the velocity of the remaining errors changes is directly proportional to the net applied effort (e_n) and inversely proportional to the complexity (s_c) of the program under test:*

$$\ddot{r}(t) = \frac{e_n(t)}{s_c} \Rightarrow e_n(t) = \ddot{r}(t) s_c \quad (1)$$

Assumption 2: *The effective test effort is proportional to the product of the applied work force and the number of remaining errors:*

$$e_f(t) = \zeta(s_c) w_f r(t) \quad (2)$$

where $\zeta(s_c) = \frac{\zeta}{s_c^b}$ is a function of software complexity. Parameter b depends on the characteristics of the product under test.

Assumption 3: *The error reduction resistance (e_r) opposes, is proportional to the error reduction velocity (\dot{r}), and is inversely proportional to the overall quality (γ) of the test phase. For an appropriate constant ξ , this leads to*

$$e_r(t) = -\xi \frac{1}{\gamma} \dot{r} \quad (3)$$

The negative sign indicates that the error reduction always opposes \dot{r} .

Combining Eqs. 1, 2, and 3 in a force balance equation and organizing it in a State Variable format ($\dot{x} = Ax + Bu$) [7] produces the following system of equations.

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-\zeta w_f}{s_c^{(1+b)}} & \frac{-\xi}{\gamma s_c} \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{s_c} \end{bmatrix} F_d \quad (4)$$

F_d above is included in the model to account for unforeseen disturbances such as hardware failures, personnel illness or any event that slows down or even interrupts the continuation of the test process. The level of disturbance, i.e. how much the observed behavior deviates from the expected one, is detected over a period of time and computed as a portion of the effective test effort (e_f). It is then partially or fully propagated to the remaining period.

Along with the model in Eq. 4 an algorithm has been developed to calibrate the parameters of the model [2]. The fast convergence presented by the algorithm increases the model applicability and accuracy. Finally, a parametric control procedure is used to compute required changes in the model in order to converge to desired results according to time constraints [2]. The input $u(t)$ is, in general, used to drive the system. However, the procedure used for the control of the system in Eq. 4 uses a parametric approach and the input F_d is used to account for unforeseen perturbations, as stated earlier.

3 Statistical Process Control

Statistical Process Control, as described by Florac and Carleton [8], is an analysis tool to improve the controllability of the Software Development Process (SDP). The determination if a process is stable and/or capable represents the major concern of the technique.

A Control Chart is used commonly to determine whether or not a process is under control, i.e., stable. Several different types of Control Charts exist; X-Bar, Range Charts, U Charts, and Z Charts are a few examples. Each chart is designed to be used based on specific conditions related to the available data. We exemplify the use of a Control Chart by describing an X-Bar chart [8].

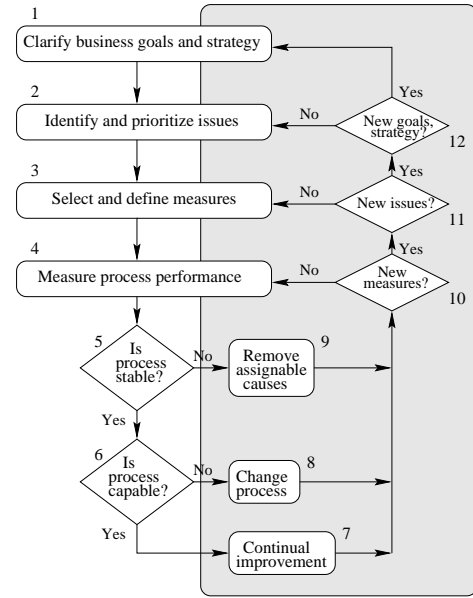


Figure 1. Sequence of steps for Statistical Process Control [8].

An X-bar chart is based on averages of grouped data. It requires that data be grouped into subgroups of size at least two. The data for each subgroup i is averaged (\bar{X}_i) and the standard deviation for the subgroup σ_i is computed. Then, the average of the subgroup averages ($\bar{\bar{X}}$) is computed along with the average standard deviation ($\bar{\sigma}$). Based on these values, three lines are presented in an X-Bar chart: the (CL) center line which is $\bar{\bar{X}}$, the (UCL) upper control limit $\bar{\bar{X}} + 3\bar{\sigma}$, and (LCL) the lower control limit $\bar{\bar{X}} - 3\bar{\sigma}$. An example X-Bar chart is shown in Figure 4(c). In this work we assume the standard control limit ($\bar{\bar{X}} \pm k\bar{\sigma}$) for $k = 3$. However, as pointed out by Jalote and Saxena [9], an optimized control limit can be established to better match the specific characteristics of a software process. This change in the control limit will not affect the SPC variant presented here.

The analysis of an X-Bar chart determines process stability. A process is said to be out of control (unstable) if

Table 1. Steps in applying the SPC_{log} approach

Step	Stability	Capability
1	$ld_i = \log(v_i), i = 1, \dots, n$	compute the expected decay ($e_v = v_0 e^{-\lambda \times t}$).
2	compute least square fit (lsf_i) of ld_i	compute the log of the expected value ($\log(e_v)$) and the upper ($e_v^{up} = (1 + \Delta) \times \log(e_v)$) and lower ($e_v^{low} = (1 - \Delta) \times \log(e_v)$) specification limits for a given Δ variation.
3	compute $x_i = \sqrt{(lsf_i - ld_i)^2}$	compute TCL, the average Euclidean distance between the expected and observed averages ($TCL = \sqrt{(e_{ld}(i) - \log(v_i))^2}$).
4	using $x_i, i = 1, \dots, n$, compute the control limits as in SPC	compute $USL = TCL + \sqrt{(e_v^{up}(i) - e_{ld}(i))^2}$ and $LSL = TCL - \sqrt{(e_{ld}(i) - e_v^{low}(i))^2}$.
5	check stability using SPC tests	check capability using SPC tests

any one of the following tests fail [8].

- Test 1 : A single point is outside the limits of LCL and UPL.
- Test 2 : At least two out of three successive values fall on the same side of, and more than 2σ units away from, the center line.
- Test 3 : At least four out of five successive values fall on the same side of, and more than 1σ unit away from, the center line.
- Test 4 : At least eight successive points fall on the same side of the center line.

We observe in Figure 4(c) that according to the tests defined above the process under observation is under control, that is, the process is stable. Thus, now we need to analyze whether or not the process is capable of accomplishing the planned task when restricted to a predetermined cost and schedule. A Capability Histogram such as the one in Figure 4(d) can be used to accomplish this task by determining if a process is outside the expected limits. Being outside expected limits represents a higher probability of generating non-conforming results and indicates that the process should be adjusted in order to achieve the expected results. The equations used to determine process capability [8] are presented below.

$$ST = \frac{USL - LSL}{\sigma} \quad (5)$$

$$DNS = \min\{Z_u, Z_l\} \quad (6)$$

$$Z_u = \frac{USL - \bar{X}}{\sigma} \quad (7)$$

$$Z_l = \frac{\bar{X} - LSL}{\sigma} \quad (8)$$

where \bar{X} is the average of the observed values during the period under consideration, ST is the Specification Tolerance, USL is the Upper Specification Limit, LSL is the

Lower Specification Limit, DNS is Distance to the Nearest Specification, Z_u is the distance in σ units between the process average and USL; and Z_l is the distance in σ units between the process average and LSL. Wheeler states "... sigma units express the number of measurement units which correspond to one standard unit of dispersion" [10]. Sigma units are used to characterize how much of the data is within a given α distance from the observed average [10].

In Figure 1 we observe the steps followed to monitor a process when SPC [8] is used. Step 5 represents one of the stability tests listed earlier. The capability test is represented by Step 6. The details of how SPC determines if a process may or may not be capable are provided in Figure 2.

SPC as presented above does not support the monitoring of a process presenting an exponential behavior [11] as the decrease in failure intensity and number of remaining errors of the system test phase. However, a variation of the standard SPC technique, named SPC_{log} , does provide the ability for the monitoring of such processes [11]. A summary of the steps/transformations for the application of SPC_{log} appears in Table 1. A detailed description of SPC_{log} is beyond the scope of this paper and can be found elsewhere [11].

4 Combining SPC_{log} and the State Variable Model

Before engaging in combining the two approaches it is necessary to clarify the terminology distinctly used in both approaches. The concepts of stability, capability, and controllability are of major importance and are analyzed next under the perspective of both approaches.

Stability: with regard to SPC, a process is stable if it varies within predictable limits. If the variation is beyond these limits, the process is considered unstable (and *not under control*) [10]. From a control theory perspective, a process is stable if, when starting from any initial point x_0 , the system will always converge to a finite *equilibrium point*

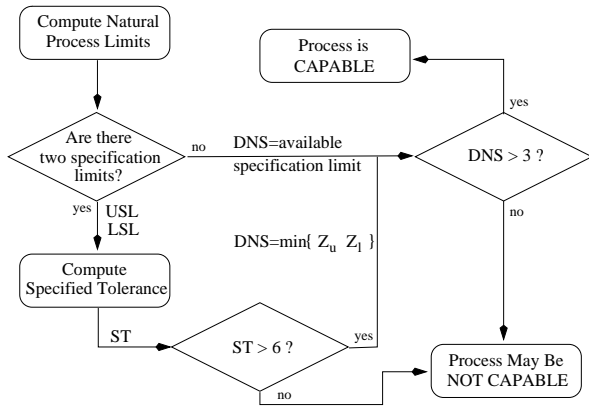


Figure 2. Capability verification for Statistical Process Control [8].

x [7] or more generally to a finite cycle. Otherwise, the process is unstable. A finite *equilibrium point* x is a state vector bounded in norm where the system will remain once reached [7]. If the system state always converge to an equilibrium value of $x=0$, then the system is asymptotically stable.

Controllability: in control theory, a system is completely controllable if it can be driven by some stimulus from any initial state at time t_0 to any other state (e.g. the origin) at time t_1 for $t_0 < t_1 < \infty$. Controllability implies that if $r_0 > 0$ is an initial number of errors at time t_0 , the number of errors can be reduced to any desired (small) value at time t_1 . Although this requirement is implied by controllability, the converse is not in general true. It should be noticed that the concept of controllability presented above is not present in SPC since the technique does not allow for the tuning of the process. In summary, controllability in SPC means “a process under control” whereas with regard to control theory it means “be able to control”.

Capability: capability is defined in SPC as: *A process is capable if it is able to achieve the expected results within a specified time. Capability is based on the probability that the process can achieve the results within the specified time. Clearly this presupposes that the process is stable.* [10] Although the concept of capability is not explicitly defined in control theory, the availability of a model to predict the future behavior of the process allows the assessment of the capability as done in SPC.

4.1 Combining the approaches

A software test process differs from a physical process not only by the type of the product but also regarding managerial decisions. A machine in a physical plant can be adjusted as frequently as necessary and the results can be easily identified. In this case a self-regulation mechanism can be design to automatically correct the process and produce the desired results within the specified time. Such process would be suitable for the sole application of con-

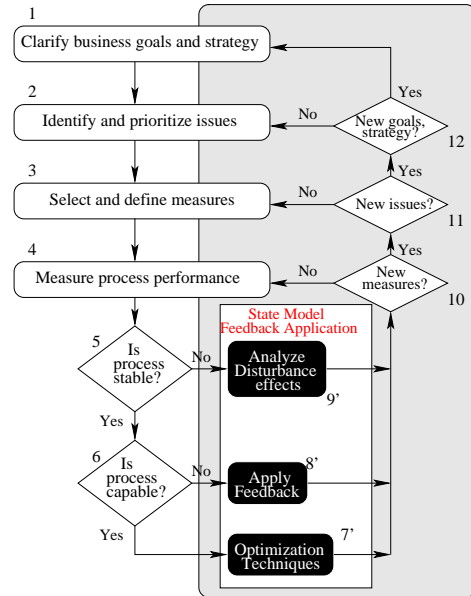


Figure 3. Combined approach of Statistical Process Control (SPC) and the Feedback Process Control (FPC) approach based on a state variable model. Steps 7, 8, and 9 in Figure 1 are replaces by quantitative feedback procedures from FPC.

control theory. However, a test manager cannot and should not make frequent changes, for example, in the test team or in the technique used to test the product. The side effects of such frequent changes will, most likely, slow down the process performance and the quality of the product. Under this condition one would argue that the STP is not suitable for the application of FPC. As stated before, the question to be asked is “When is the appropriated time to make changes in the software test process”.

Statistical Process Control provides the answer to the above question by monitoring the process under observation and determining when it has crossed a specified limit based on averages and σ units (standard deviation) computed with the collected data. The assumption is that when subgroup averages are too distant from the overall average the process is not stable (Step 5 in Figure 1) due to assignable causes and changes are required. At this point, it is up to the manager to identify the assignable causes and make the appropriated change in the process. These assignable causes can be seen as unforeseen perturbations in FPC and once identify, the control mechanism in the state variable model can be used to adjust the process in order to make it converge to the expected behavior [12]. Therefore Step 9 in Figure 1 can be replaced by Step 9’ in Figure 3. The use of a control mechanism decreases the dependency of the process on the manager skills and thus improves its controllability.

The control mechanism in FPC is also used when the process is determined to be not capable (Step 6 in Figure 1). Again SPC does not provide means to quantify the changes

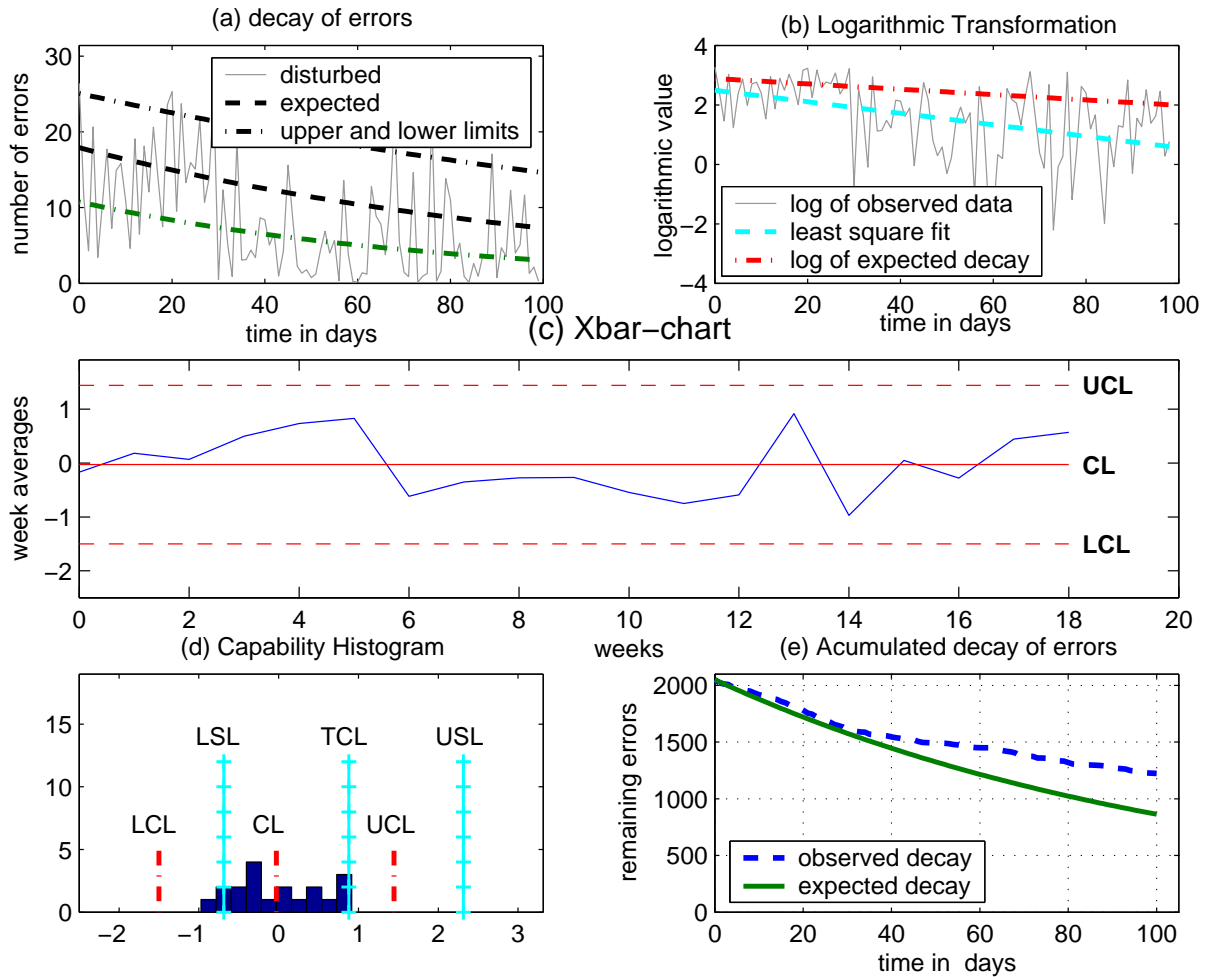


Figure 4. Results from the application of SPC_{log} to one of the simulation runs for the first 100 days of the process.

necessary to make the process capable and the decisions are based mostly on the experience of the manager. The quantification of the required changes is provided by FPC and in this case, Step 8 in Figure 1 is replaced by Step 8' in Figure 3.

When a process is determined to be stable and capable, the experience/expertise of the test manager has again the major impact for the continuing improvement in the process (Step 7 in Figure 1). However, the availability of the state model for the software test process allows the application of optimization techniques present in control theory [4, 7]. Though these optimization techniques are still under investigation for the state model of the STP, the replacement of Step 7 in Figure 1 by Step 7' in Figure 3 can be foreseen.

The use of the control mechanism of the state model of the STP diminishes the dependency on the test manager experience to regulate the process and therefore increases its controllability. However, it should be clear that managers are still part of the feedback process but now they can gain not only from the monitoring aspects of SPC but

also from the quantitative feedback solution provided by FPC.

5 Application of the combined approach

Simulation is used in this section to show the applicability of the combined approach. Data is generated by randomly slowing down the exponential decay of errors for the process. Assume a software test process is being monitored using SPC_{log} , the logarithmic variation of SPC. Also assume that the process is characterized by the following parameters: $s_c = 15$, $w_f = 3$, $\gamma = 0.6$, and $b = 1.12$. The expected error reduction for the first 100 days for this process is depicted in Figure 4(e) and the entire period is depicted in Figure 5. From Figure 4(c) it can be observed that the process is stable but Figure 4(d) depicts a not capable process. This results are confirmed by the capability tests at time $t = 80$ where $ST = 4.77$ is not greater than 6 leading to a not capable process. However, as can be noticed from Figure 4(e), the process started to deviate from the expected behavior as early as $t = 40$. At this point, the results

of the capability tests were positive ($ST = 8.672 > 6$ and $DST = 3.388 > 3$) indicating a capable process. As expected, the results indicate that at $t = 40$ the deviation was not large enough to justify changes in the process but has reached reasonable values to justify the changes at $t = 80$.

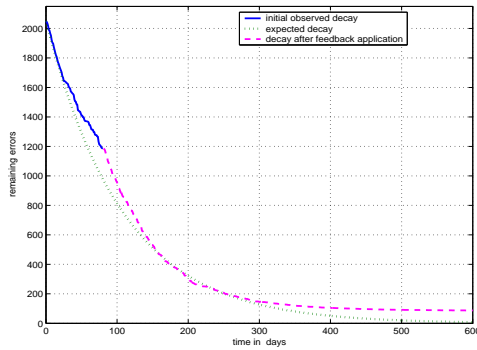


Figure 5. Application of FPC to a process determined to be “not capable” by SPC.

As stated before, SPC does not provide feedback mechanisms to quantify the changes needed to correct the observed deviations. Such mechanism is presented in FPC. Figure 5 shows the expected behavior (dotted line), the observed error reduction (solid line) for the period $t = 0 \dots 80$ and the behavior after the application of feedback (dashed line). As can be noticed, the process converges to the expected behavior after the feedback change. Initially, the largest eigenvalue¹ of the system was $\lambda_{max} = 0.009$. To make the system converge to the expected curve, the new eigenvalue should be $\lambda_{max} = 0.0123$. An increase of the size of the test team by 1 member places λ_{max} at the desired point and achieves the convergence as noticed in Figure 5. The capability tests are applied after the feedback change confirming a capable process ($ST = 8.901 > 6$ and $DST = 3.240 > 3$).

6 Concluding Remarks

The combination of two powerful techniques that have been used to the monitoring and control of software processes was presented in this paper. Each of these techniques has its strength and drawbacks. SPC appear to better determine the time when changes in the process are required but lacks a feedback mechanism to quantify the changes. Such mechanism is available in FPC. Though the prediction ability of FPC allows the identification of deviations from an expected behavior, it does not provide an answer to when is the appropriated time to correct the deviations. It is clear that SPC and FPC complement each other to better assist the manager in the decision making process. Though more experiments and data from actual projects are needed to validate the combined approach, the

¹The largest eigenvalue, denoted here by λ_{max} , dominates the rate of convergence

results from the experiments conducted here provide initial support for the improvement in the overall controllability of the software test process.

References

- [1] M.-H. Chen, M. R. Lyu, and W. E. Wong, “Effect of code coverage on software reliability measurement,” *IEEE Transactions on Reliability*, vol. 50, pp. 165–170, June 2001.
- [2] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur, “A formal model for the software test process,” *IEEE Transaction on Software Engineering*, vol. 28, pp. 782–796, August 2002.
- [3] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur, “Using sensitivity analysis to validate a state variable model of the software test process,” *IEEE Transactions on Software Engineering*, vol. 29, pp. 430–443, May 2003.
- [4] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*. John Wiley & Sons, 1996.
- [5] G. A. Hansen, “Simulating software development processes,” *IEEE Computer*, vol. 29, pp. 73–77, January 1996.
- [6] T. Abdel-Hamid and S. E. Madnick, *Software Project Dynamics: An Integrated Approach*. Prentice Hall Software Series, New Jersey, 1991.
- [7] D. G. Luenberger, *Introduction to Dynamic Systems: Theory, models and applications*. John Wiley & Sons, 1979.
- [8] W. A. Florac and A. D. Carleton, *Measuring the Software Process: Statistical Process Control for Software Process Improvement*. SEI Series in Software Engineering, Addison-Wesley, 1999.
- [9] P. Jalote and A. Saxena, “Optimum control limits for employing statistical process control in software process,” *IEEE Transactions on Software Engineering*, vol. 28, no. 12, pp. 1126–1134, 2002.
- [10] D. J. Wheeler and D. S. Chambers, *Understanding Statistical Process Control*. SPC Press, 1992.
- [11] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur, “Monitoring the software test process using statistical process control: A logarithmic approach,” in *Proceedings of Symposium on the Foundations of Software Engineering*, (Helsinki, Finland), ACM SIGSOFT, September 1-5 2003. (to appear).
- [12] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur, “Effect of disturbances on the convergence of failure intensity,” in *Proceeding of 13th International Symposium on Software Reliability Engineering*, (Annapolis, MD, USA), November 12-15 2002.