

Convergence Assessment of the Calibration Algorithm for the State Variable Model 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

This paper reports an evaluation of the calibration algorithm for the state variable model of the software test process. The data to evaluate the convergence of the estimation of the parameters of the model is generated using simulation. The analysis exposes that even under the assumption of an initial over/under estimation of the parameter, the algorithm presents a fast convergence. The results show that in almost 90% of the cases the algorithm uses data from a short period of time to obtain an accuracy of, at least, 15% in the prediction of the time to reach a desired level in the error reduction. These outcomes show the efficiency of the calibration algorithm and further demonstrate the applicability of the state variable model.

KEY WORDS

Software test process, state variable model, parameter calibration, system identification, feedback control.

1 Introduction

Two major aspects affect the applicability of any model. The first aspect determines how well the model captures the dominant behavior of the process it represents. The second, assesses how accurate is the estimation of the parameters of the model. The state model of the Software Test Process (STP) has shown to properly capture the dominant aspects of the process as observed in the conducted case studies [1] as well as in the results of a sensitivity analysis [2].

A high sensitivity due to {over/under} estimation of the parameters of the model at the beginning of the process was pointed out by the sensitivity analysis [2]. This result indicates the necessity of a calibration algorithm that increases the accuracy of the estimations of the parameters and consequently decreases the effects of an initial {over/under} estimation. The goal of this paper is to show that the calibration algorithm proposed in an early work [1, 3] produces results that fulfill this necessity. The algorithm is described in Section 2.2 and is referred hereafter as \mathcal{CA} .

The assessment of the convergence of the algorithm \mathcal{CA} is based on data generated by simulation runs. Though it is common sense that actual data would be more ap-

propriated, the absence of such data makes simulation a suitable tool to analyze the algorithm. The results of the simulation runs show that algorithm \mathcal{CA} indeed accomplish the requirements to overcome the problem of an initial {over/under} estimation of the parameters.

The remainder of this paper is organized as follow. Section 2 presents a brief description of the state model and the algorithm used to calibrate it. The technique applied to generate data to be used in the convergence assessment of the calibration algorithm is described in Section 3. The sequence of steps used to recalibrate, applying algorithm \mathcal{CA} , the parameters of the state model is presented in Section 4. The results of the simulation runs are presented in Section 5 and the concluding remarks in Section 6.

2 Background

2.1 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 [1]. 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 [4]. A description and justification of this analogy and the choice of a linear model is outside the scope of this paper. [1]

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.

Borrowing from COCOMO [5], we set b to 1.05, 1.12, or 1.20, for *organic*, *semi-detached*, and *embedded* mode projects, respectively.

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 ($\dot{x} = Ax + Bu$) [4, 6] 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.

2.2 Parameter Calibration Algorithm

This section presents algorithm \mathcal{CA} used to calibrate parameters ξ , ζ for the state model of the STP. The algorithm is based on a weighted least square approach using system identification techniques [7]. The parameters are computed from data obtained from the ongoing project. Obviously, this data is not available at the start of the STP. However, a manager may have data from similar past projects that can be used to obtain initial estimates until data from the current project becomes available when the estimates can be improved.

The state model of Eqn. 4 has the general form of

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (5)$$

where $x(t) = [r(t) \dot{r}(t)]^T$, A is the proper 2×2 matrix, B is a 2×1 vector, and $u(t)$ is the input. It is well known that the solution of Eqn. 5 is given by $x(t) = e^{At}x(0)$ for all $t \geq 0$, for a zero input. To compute ξ and ζ we need to compute the eigenvalues of the matrix

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{\zeta w_f}{s_c^{(1+b)}} & -\frac{\xi}{\gamma s_c} \end{bmatrix} \quad (6)$$

from which we can obtain ξ and ζ as all the other parameters are known at this time.

At the starting point ($t = 0$), $r(0)$ is unknown and $\dot{r}(0) = 0$. However, it is assumed here that an initial estimate of $r(0)$ is available using historic data [8, 9]. Further, project data ordinarily consist of the number of errors found and fixed in a time period of length, say T_1 . Although $r(t)$ has the general form of a double exponential solution [1], we use a single exponential approach to obtain a local approximation for $\dot{r}(t)$. Having the initial estimate of $r(0)$, $r(T_1)$ is computed subtracting from $r(0)$ the number of errors found until the moment T_1 when the data is collected. The assumption of an exponential decay leads to $r(T) = r(0)e^{-\lambda \times T} \Rightarrow \lambda = \frac{1}{T}(\ln(r(0)) - \ln(r(T)))$, where λ determines the rate of decay. The approximation for the velocity can now be computed as $\dot{r}(t) = -\lambda r(0)e^{-\lambda \times t}$.

Inherent in the above is a single exponential approximation to obtain a reasonable estimate of the velocity data¹. Now, using data available and the approximated \dot{r} we compute the difference for a specific period of time as $D^i = [r(i) \dot{r}(i)]^T - [r(i-1) \dot{r}(i-1)]^T$. It can be shown that $D^i = M D^{i-1}$, where $M = e^{AT}$, T being the time increment between two consecutive measurements of data, and A the A-matrix of Eqn. 5. Therefore, we can compute M from

$$R_1 = M R_2 \implies M = R_1 R_2^{-R} \quad (7)$$

where $R_1 = [D^n \ D^{n-1} \ D^{n-2} \ \dots \ D^4 \ D^3]$, $R_2 = [D^{n-1} \ D^{n-2} \ D^{n-3} \ \dots \ D^3 \ D^2]$ and $-R$ is the Moore-Penrose pseudo-right inverse [4, 6].

The Spectrum Mapping Theorem [6] shows that the eigenvalues of M , say λ_1^M and λ_2^M , have the following relation with the eigenvalues of matrix A : $\lambda_1^M = e^{\lambda_1 T}$ and $\lambda_2^M = e^{\lambda_2 T}$. Therefore $\lambda_1 = \frac{1}{T} \ln(\lambda_1^M)$ and $\lambda_2 = \frac{1}{T} \ln(\lambda_2^M)$. The eigenvalues are the roots of the characteristic polynomial of the A matrix, as in Eqn. 6, which is

$$\begin{aligned} \Pi_A(\lambda) &= \det[\lambda I - A] \\ &= \lambda^2 + \frac{\xi}{\hat{\gamma} s_c} \lambda + \frac{\zeta \hat{w}_f}{s_c^{(1+b)}} \end{aligned} \quad (8)$$

Since ξ and ζ are the only unknowns at this time, we can compute them by matching the roots of $\Pi_A(\lambda)$ to λ_1 and λ_2 computed above.

An initial estimate of $r(0)$ can also be computed using M obtained from the observed data. Let

$$P = \begin{bmatrix} M^2 - M \\ M^3 - M^2 \\ \vdots \\ M^n - M^{n-1} \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} D^2 \\ D^3 \\ \vdots \\ D^n \end{bmatrix}$$

For $x_0 = [r(0) \dot{r}(0)]^T$, it is known that $Z = P \times x_0$ leading to the computation of $x_0 = P^{-L} \times Z$, where $-L$ is

¹The approximation for the velocity is computed in this paper differently from a previous version of the algorithm [3].

the Moore-Penrose pseudo-left inverse [4, 6]. The analysis of the computation of x_0 is beyond the scope of this paper.

3 Data Generation

As pointed out in Section 1 the availability of data is a major concern for the validation of new techniques such as the state model of the STP. The use of simulation to generate data helps to overcome this issue by exercising the model and, in this case, the calibration algorithm \mathcal{CA} repetitively and under different scenarios. Two scenarios are of interest here. The first generates data presenting an overall slow down in the rate of decay of the number of remaining errors when compared to an initial estimate. The second scenario presents an overall acceleration in the process. A third scenario would result from the combination of the first two, but it leads to data too similar to the initial estimate and therefore of no interest.

The two scenarios cited above generate data based on the same initial estimate. Assume a process \mathcal{P} is about to start and the model is initialized with the following values for the parameters: $s_c = 30$, $w_f = 5$, $\gamma = 0.6$, $b = 1.12$, $\xi = 23$, and $\zeta = 7$. As time advances, data from \mathcal{P} is collected and the parameters of the model are re-calibrated. The collected data, in this case, is obtained by randomly changing the rate of decay of r for the two scenarios described next.

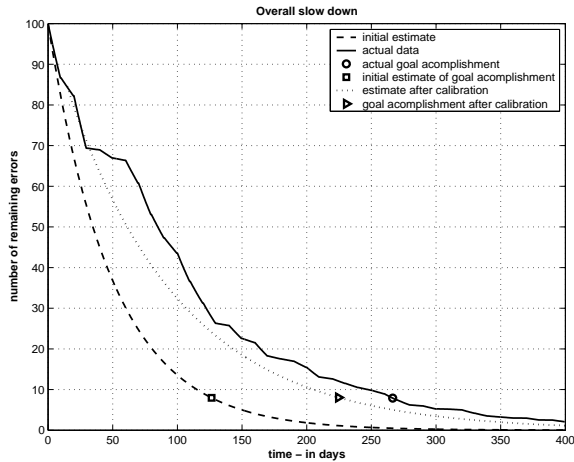


Figure 1. Initial estimate of error reduction (dashed line), the corresponding generated “actual” data for the overall slow down scenario (solid line), and the prediction after calibrating the model using data from the first 10 days of observation (dotted line).

3.1 Scenario I - Overall Slow Down

The first scenario to be analyzed represents an overall slow down for the test process. The decay of the number of errors is expected to have an exponential shape determined

by a double exponential solution [1]. A single exponential approximation $r(t) = r(0)e^{-\lambda t}$ with decay parameter $\lambda = 0.02$ produces almost the same output as when the state model have the parameters set as above. The “actual” data are generated by randomly changing the value of λ to $\lambda \times \alpha$, where $0 < \alpha < 1$. The smaller the decay parameter, the slower the rate of decay and therefore the overall slow down is guaranteed by $\alpha \times \lambda < \lambda$. This change is done for time intervals of 10 days. Figure 1 shows the initial estimate of the decay of r and the generated “actual” data as described above. Two hundred simulation runs were executed to generated the results presented in Section 5.

3.2 Scenario II - Overall Speed Up

The overall speed up of the process is generated similarly to the slow down. The difference is that the rate of decay λ is multiplied by a random number $1 < \beta < 2.5$. An acceleration is ensured since $\lambda < \beta \times \lambda$. The result of one of the 200 simulation runs for the overall acceleration is presented in Figure 2.

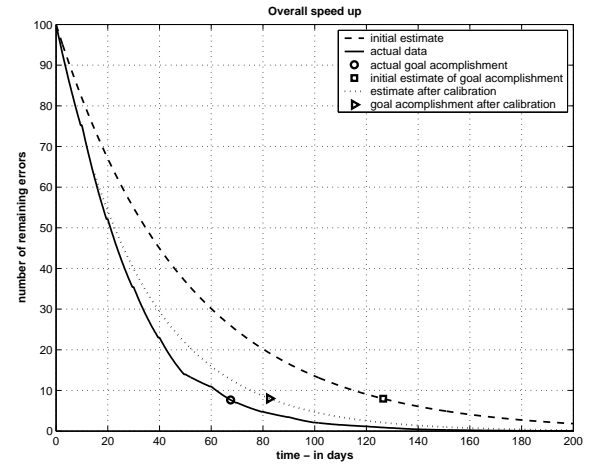


Figure 2. Initial estimate of error reduction (dashed line), the corresponding generated “actual” data for the overall speed up scenario (solid line), and the prediction after calibrating the model using data from the first 5 days of observation (dotted line).

4 Application of Algorithm \mathcal{CA}

Let the goal of the STP be the reduction of the number of errors to 8% of its initial value. The accuracy in predicting the time when the goal will be achieved is used to assess the convergence of algorithm \mathcal{CA} . An example of the actual time to achieve the goal as well as the initial estimate and the prediction after the calibration using data from the first period of time can be observed from Figures 1 and 2. The calibration for the slow down scenario initially uses data collected from the first 10 days with increments of 10

days each time the recalibration needs to be done. The only difference in the acceleration scenario is that the time frame is incremented by 5 instead of 10 after it starts.

Under these conditions, algorithm from Figure 3 is executed 200 times for each scenario. The line “Calibrate model with collect_data” in Figure 3 corresponds to the application of algorithm \mathcal{CA} . The variables “initial_estimated_goal” and “new_estimated_goal” correspond to the time needed to achieve an error reduction of 8% according to, respectively, the initial estimated and the estimation using the “actual” collected data for the time period.

```

generate ``actual`` data for the entire process
actual_goal = time to reduce r to 8%
estimated_goal = initial_estimated_goal
goal_difference = ABS(actual_goal - estimated_goal)
accuracy = (goal_difference*100)/actual_goal
period_size = 10
period = 0
while accuracy > 15
    period = period + 1 ;
    for t=0,..,period_size*period
        collect_data(t)
    Calibrate model with collect_data
    estimated_goal = new_estimated_goal
    goal_difference = ABS(actual_goal - estimated_goal)
    accuracy = (goal_difference*100)/actual_goal
end

```

Figure 3. Application of algorithm \mathcal{CA} to assessment of its convergence

One could always argue that, in reality, the actual time to achieve the goal is unknown. This constitutes the main reason to analyze the convergence of algorithm \mathcal{CA} . As can be observed in Figure 3, the calibration algorithm is applied first using data collected from the first period of time (10 days for the slow down and 5 days for the speed up scenario). In case the accuracy of the prediction to achieve the expected error reduction is not within a 15% range, the algorithm is re-applied using data from the first two periods. The recalibration goes on using more and more data until a 15% accuracy is reached. Therefore, the convergence of algorithm \mathcal{CA} can be measured by keeping track of the number of periods used to achieve the desired accuracy for each of the simulation runs.

5 Results

Before starting analyzing the results of the convergence of algorithm \mathcal{CA} , it is important to evaluate how distant are the initial estimates from the “actual” time to achieve an error reduction of 8%. Figure 4 shows a minimal difference of 57 days and a maximal difference of 251 days with an average and standard deviation of, respectively 138.3 and 29.2 days. These results show an under estimation of the parameters leading to a reasonable inaccuracy in the initial prediction as would be expected in a software process.

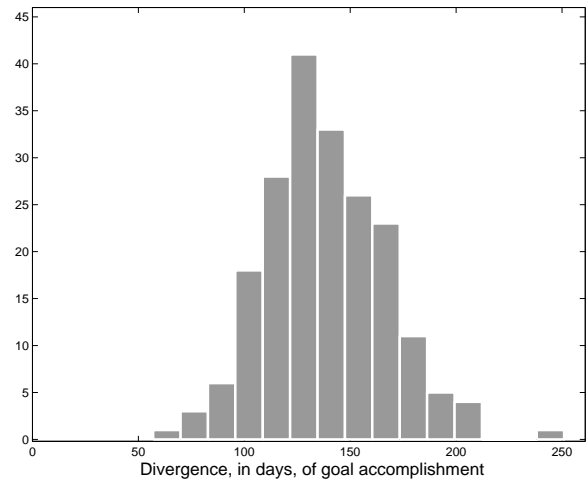


Figure 4. Difference in days between the actual date and the initial estimate to achieve the desired error reduction of the project.

Figure 5 depicts the same results in a percentage manner where a difference ranging from 30.9% to 66.4% with an average of 51.5% and a standard deviation of 5.3%. These values show that not only the absolute but also the relative difference in the initial estimate are significantly inaccurate and properly represent an initial under estimation of the parameters of the model.

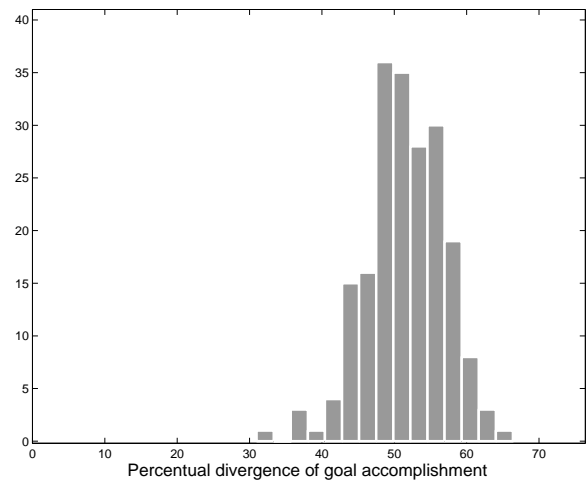


Figure 5. Percentage difference between the actual data and the initial estimate to achieve the desired error reduction for the project.

As stated before, algorithm \mathcal{CA} is used to re-calibrate the model until it reaches a desired accuracy of 15%. Figure 6 shows the new slippage, after the recalibration, to achieve the desired error reduction when compared to the generated “actual” value. As can be observed, the divergence now ranges from zero to a maximum value of 48 days. Comparing this range with the 57,..,251 range of ini-

tial estimate shows a great improvement in predicting the time to accomplish an error reduction of 8%. Furthermore, the average slippage after the recalibration of the model drops to 22.7 days with a standard deviation of 11 days indicating an average improvement of 115 days in the prediction of the time to achieve the specified goal as observed in Figure 7.

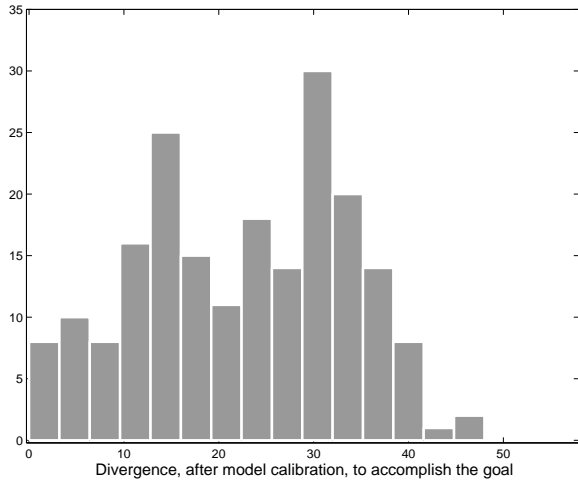


Figure 6. Difference in days between the actual date to achieve the quality goal of the project and the estimated values after the calibration of the parameters of the model.

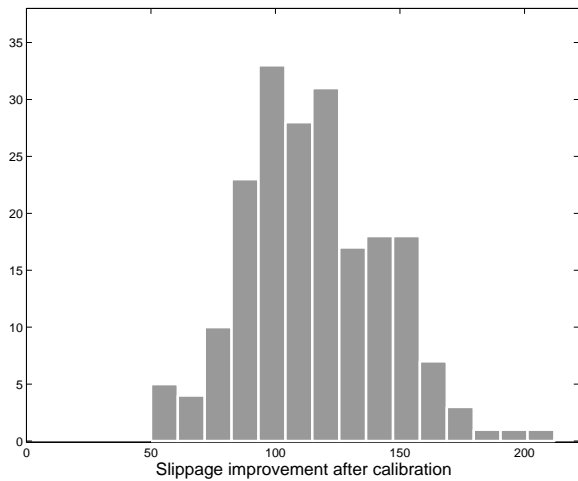


Figure 7. Slippage improvement after the recalibration of the state model using algorithm \mathcal{CA} .

The assumption that algorithm \mathcal{CA} should be executed until an accuracy of at least 15% is achieved is confirmed by the plot in Figure 8. Though the minimum accuracy required is 15%, the average accuracy obtained after the calibration of the model for the 200 simulation runs is 8.5%. The improvement in the slippage is a strong indication of the application of algorithm \mathcal{CA} and consequently of the state variable model for the STP [1, 3].

A very accurate, but somewhat useless, prediction can be achieved when a project is close to its end. The earlier a good prediction is available, the better use a project manager can extract from it. Therefore, one remaining question needs to be answered: How much data is needed to achieve the specified level of accuracy using algorithm \mathcal{CA} ? Figure 9 presents the answer for this question.

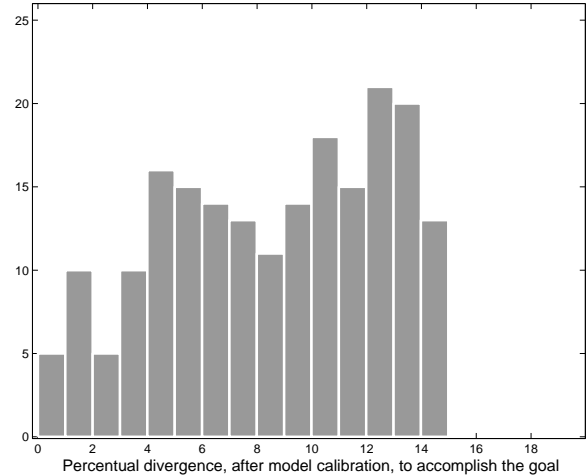


Figure 8. Percentage difference between the actual data to achieve the quality goal of the project and the estimated values after the calibration of the parameters of the model.

It can be observed from Figure 9 that 53 (26.5%) simulation runs needed only the first period of 10 days to achieve a minimum accuracy of 15%. The number of simulation runs needing period of times of 20 and 30 days are respectively, 39 (19.5%) and 45 (22.5%). Adding these three periods together results in 68.5% of all the simulation runs achieving a reasonable level of accuracy using, at most, the first 30 days of observed data. Considering that the average time to reach the desired error reduction for all the simulation runs is 274 days, the algorithm needs around 11% (30 days) of the process life time to produce a good estimate of the time required to achieve the desired results for 68.5% of all the simulated cases. The next three time periods (40, 50, and 60 days) were needed to produce the desired accuracy for 21% of the cases. Adding this to the values of the first three periods results in almost 90% of the cases (simulation runs) needing, at most, 21% of the total time to produce an estimate within 15% accuracy. These results express the applicability and efficiency of algorithm \mathcal{CA} .

From Figure 9 it can also be observed that in a few cases algorithm \mathcal{CA} did not have a very good convergence needing almost 45% of the total time to converge to the desired accuracy. Also, in one case it was necessary to use data from 200 days to achieve this goal. The reason for this is a sequence, not necessarily consecutive, of periods where the generated decay rate ($\lambda \times \alpha$) is close to zero. This creates a delay in the time to achieve the desired error reduction that is approximately the same length of the period

where decay rate is almost null slowing down the convergence of algorithm \mathcal{CA} . When many periods of time have a zero decay rate, the calibration reaches a critical point and algorithm \mathcal{CA} presents a very slow convergence. However, this fact does not compromise the application of the technique presented here for two reasons: i) this scenario is not frequently observed and, at least in the simulation runs, was presented in only one case, and ii) an almost zero decay rate indicates an unforeseen perturbation of the process and the delays associated with them can be predict [10].

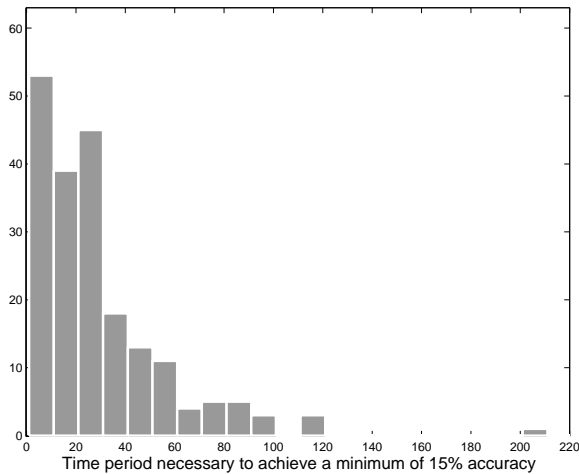


Figure 9. Number of simulation runs (y axis) that need to use data from a period of time of a certain length (x axis) to achieve a minimum of 15% accuracy in the prediction.

The analysis presented so far considered only the overall slow down scenario in Section 3.1. The analysis of the acceleration scenario in Section 3.2 produced even better results than the slow down scenario. The reason for this improvement of performance of the calibration is due to the fact the algorithm absorbs better a fast decay rate than a almost zero (very slow) decay rate. The analysis of the accuracy in the estimation of the initial number of error also produced good results but is beyond the scope of this paper.

6 Concluding Remarks

Any model, no matter how complex it is, is unlikely to perfectly reproduce the actual behavior of a system and a certain degree of approximation is expected. Assume that a model itself is capable of capture the relationship between inputs, outputs, and internal variables in a very precise manner. The outcome of this unlikely perfect model can still be compromised by a “bad” estimation of the parameters, demanding a reasonably accurate calibration procedure.

When “good” historic data is available, it is reasonable to assume that the parameters of a model can be properly calibrated resulting in an accurate prediction prior to

the beginning of the process. However, in many cases, historic data does not provide all the necessary answers to calibrate the model or, simply, it does not exist. In these cases an over/under estimation of the parameters can be expected culminating in an inaccurate initial prediction of the results. This problem can be overcome as data from the current process becomes available and the parameters are re-calibrated. The results presented in this paper show that algorithm \mathcal{CA} used for the calibration of the state variable model of the STP has a fast convergence as indicated in Section 5 decreasing the effects of an initial over/under estimation.

References

- [1] 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.
- [2] 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*. (to appear).
- [3] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur, “A state model for the software test process with automated parameter identification,” in *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference (SMC 2001)*, (Tucson, Arizona), pp. 706–711, October 2001.
- [4] D. G. Luenberger, *Introduction to Dynamic Systems: Theory, models and applications*. John Wiley & Sons, 1979.
- [5] B. W. Boehm and et. al., *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.
- [6] R. A. DeCarlo, *Linear systems : A state variable approach with numerical implementation*. Upper Saddle River, New York: Prentice-Hall, 1989.
- [7] L. Ljung, *System identification: Theory for the user*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.
- [8] S. Biffi, “Using inspection data for defect estimation,” *IEEE Software*, vol. 17, pp. 36–43, November/December 2000.
- [9] C. Wohlin and P. Runeson, “Defect content estimations from review data,” in *20th International Conference on Software Engineering*, (Kyoto, Japan), pp. 400–409, IEEE Computer Society, 1998.
- [10] 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.