Effect of Disturbances on the Convergence of Failure Intensity

João W. Cangussu
Department of Computer Science
University of Texas at Dallas
Richardison-TX 75083-0688, USA
cangussu@utdallas.edu

Aditya P. Mathur*
Department of Computer Sciences
Purdue University
West Lafayette-IN 47907-1398, USA
apm@cs.purdue.edu

Raymond A. DeCarlo
Department of Electrical and
Computer Engineering
Purdue University
West Lafayette-IN 47907-1285, USA
decarlo@ecn.purdue.edu

Abstract

We report a study to determine the impact of four types of disturbances on the failure intensity of a software product undergoing system test. Hardware failures, discovery of a critical fault, attrition in the test team, are examples of disturbances that will likely affect the convergence of the failure intensity to its desired value. Such disturbances are modeled as impulse, pulse, step, and white noise. Our study examined, in quantitative terms, the impact of such disturbances on the convergence behavior of the failure intensity. Results from this study reveal that the behavior of the state model, proposed elsewhere, is consistent with what one might predict. The model is useful in that it provides a quantitative measure of the delay one can expect when a disturbance occurs.

1 Introduction

The system test phase of the software life cycle is subject to various types of disturbances, both unforeseen and known a priori. When a disturbance occurs during the test phase, it may cause a delay in the realization of the objectives. In the study reported here, the failure intensity of the software product under test is of interest to us. Thus we want to understand, and predict, how will the convergence of failure intensity to its desired value be affected due to a disturbance. This study is based entirely on a model of the software test process proposed elsewhere [1].

*Financial support provided in part by SERC and NSF.

1.1 System test, reliability growth, and the problem of interest

The system test phase (STP) in a software life cycle is intended to test a software system for conformance with both functional and non-functional requirements. According to Donnely et al. "System test and field trial activities certify that the software requirements of the product are met and that the product is ready for general use by the customer."[2]. During the system test phase, one may conduct various types of tests such as reliability growth test, regression test, and feature test. We focus on reliability growth test.

Now suppose that a test manager is given a failure intensity objective λ_d that must be met by a deadline t_d. Suppose also that at some time $t < t_d$ the failure intensity of the product under test is $\lambda(t) > \lambda_d$. Under this scenario, the following two questions are of interest to us.

> *Is there a need to alter the parameters of the test process so that the objective is met by t_d ?*
>
> *If the answer to the above question is in the affirmative then what changes, in quantitative terms, should be made to the test process in order for the objective to be met ?*

An answer to the questions above has been proposed using a state variable approach [1]. This approach allowed the development of a first order linear model of the system test phase that is used during the test process to obtain answers

to the two questions above. With respect to this state model, our interest is in obtaining an answer to the following two questions.

> *Disturbance modeling: How does one model quantitatively the disturbances that can affect the convergence of the test process to the objective ?*
>
> *Impact of disturbances: How does a disturbance effect the convergence of the test process towards the failure intensity objective ?*

1.2 Problem context

The use of feedback control during the STP is illustrated in Figure 1. The figure shows four key components that participate in the control process. These are the `Actual STP` which consists of the test engineers, tools, documentation, etc., a `State Model` of the STP which is `Model S`, a `Controller`, and the `test manager`. The management decides how many testers to employ to test the product. This number is the initial value of w_f. The test manager estimates the quality γ of the test process. The complexity of the software under test is computed as a convex combination of several well known complexity metrics such as the number of function points, lines of code, and the number of data flows [3]. An initial estimate of the failure intensity $\lambda(0) \equiv \lambda_0$ is made of the software product ready to enter the test phase. Furthermore, we assume that to plan and monitor the progress of the STP, the test manager divides the entire phase into a sequence of $n > 0$ checkpoints denoted by $\text{cp}_1, \text{cp}_2, \ldots, \text{cp}_n$ with cp_1 being the first checkpoint after testing has begun and cp_n the deadline. The realization of the failure intensity objective is distributed over these checkpoints. Thus, checkpoint cp_i is specified as a pair (t_i, λ_i^d), where t_i is some time prior to the deadline and λ_i^d is the desired failure intensity of the product at checkpoint cp_i.

At checkpoint $\text{cp}_i, i > 0$, λ_i of the product is estimated and compared against the expected λ_i^e computed by the state model. The error signal $\Delta\lambda_i = \lambda_i^e - \lambda_i$ is input to a controller. Using this error signal, s_c, w_f, and γ, the controller computes a set of possible changes $\Delta w_f'$ and $\Delta\gamma'$ that could be made to w_f and γ, respectively, in order for the STP to meet the reliability objective on or before the deadline. The computed changes are made available to the test manager who may or may not choose to ignore them. Though Figure 1 gives the impression that only a single pair ($\Delta w_f'$ and $\Delta\gamma'$) of values is output by the controller, in reality the controller outputs a finite set of such pairs from which the test manager could select. The STP resumes with a workforce of $(w_f + \Delta w_f)$ and a process quality of $(\gamma + \Delta\gamma)$, where

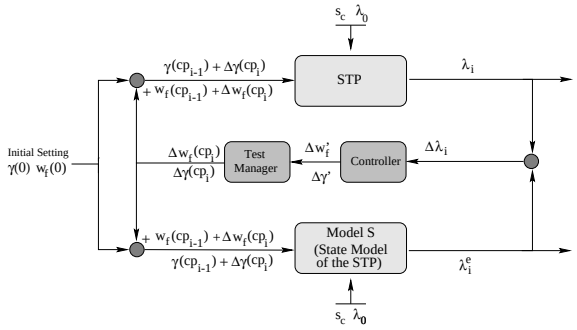

Figure 1. Closed loop control of the software test process using reliability measurements. $\lambda_i(t)$ is the estimate of the failure intensity at checkpoint cp_i. $\lambda_i^e(t)$ is the expected failure intensity of the product computed by `Model S` at checkpoint cp_i.

Δw_f and $\Delta\gamma$ denote the actual changes made by the test manager.

The STP model and the controller cooperate to form a feedback control loop. Unforeseen disturbances in the STP are accounted for by updating estimates of the model parameters. The control loop offers the test manager opportunities to make alterations to the STP at any checkpoint. These alterations could come in many forms such as a change in the number of testers, in the quality of the test process, in the test objectives themselves, or any combination of these. Thus the inherent uncertainty and the variability of the STP is accounted for in the feedback control loop.

The remainder of this paper is organized as follows. Section 2 provides an overview of the state model that serves as the basis for studying the impact of disturbances on the convergence of the failure intensity. The mathematical background required for a complete understanding of the material in this paper is beyond the scope of this paper and is found in standard texts on automatic control [4, 5]. The definitions of different types of input signals, unforeseen perturbations in `Model S`, and their correlation to the STP are provided in Section 3. The results of stimulating `Model S` with impulse, pulse, step, and white noise inputs are presented in Section 4 and analyzed in Section 5. Section 6 summarizes this work.

2 Review of State Model of the STP[1]

A linear model of the STP is based on three assumptions presented below [3]. These assumptions are based on an analogy of the STP with the physical process typified by a

[1] Section mostly extracted from author's previous work [1].

spring-mass-dashpot system and the predator-prey system. A complete description and justification of this analogy and the choice of a linear model is outside the scope of this paper and is found elsewhere. [3]

The widespread use of differential equations to model many different types of systems [5, 6] combined with the fact that most of such models were developed using analogies to physical systems with assumptions similar [7, 8] to ours further justify the choice of a second order linear model.

Assumption 1: The magnitude of the rate of decrease of failure intensity of a software product is proportional to the net applied effort during the test phase and inversely proportional to the complexity of the product.

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

The first assumption also follows from the observation that complex programs require more test effort than relatively simpler programs for approximately the same reduction in their respective failure intensities. While the value of the fault exposure ratio (FER) defined in [9] does play an important role in how the failure intensity changes, we believe that FER itself is a function of the quality of the test process and need not be treated as an independent parameter in the model. For example, a test process that uses advanced methods for test generation, such as those that use coverage measurement tools, is likely to lead to a higher fault exposure ratio, than one that uses only black-box functional testing for the generation of tests. Justification for our belief comes from data presented in earlier work [10].

The net applied effort (e_n) is the balance of all the effort applied during the test phase. This results from the difference of the effective effort applied by the test team minus any "frictional" forces that decrease the applied effort. Since λ represents failure intensity, its first derivative $\dot{\lambda}$ is the failure intensity reduction velocity (v_e). Consequently, $\ddot{\lambda}$, which denotes the rate of change of $\dot{\lambda}$, is an acceleration. Thus, the concepts of velocity and acceleration have counterparts in the test phase.

Assumption 2: The magnitude of the effective test effort is proportional to the product of applied work force and the failure intensity, i.e., for an appropriated ζ.

$$e_f = \zeta(s_c)\ w_f\ \lambda(t) \tag{2}$$

where $\zeta(s_c) = \dfrac{\zeta}{{s_c}^b}$. Parameter b depends on the software project characteristics and will have the values: i-$b = 1.05$ for an *organic mode* project; ii-$b = 1.12$ for an *semi-detached mode* project and; iii-$b = 1.20$ for an *embedded mode* project. This classification and the respective values were defined and empirically validated for the COCOMO model [11].

Assumption 2 can be understood with another analogy. In a spring the restoring force is determined by the spring stiffness and by how much the spring is extended beyond its natural length. Increasing the spring stiffness or the extension increases the restoring force. The effective test effort can be interpreted in an analogous way. The failure intensity is analogous to the spring length. At the beginning of the test phase λ is larger than it is towards the end. Hence, the effective effort decreases as λ decreases. The work force can be related to the spring stiffness. The larger the work force, the greater the restoring force, i.e., the effective effort. Thus spring stiffness is analogous to w_f and spring extension to failure intensity (λ). In Eqn. 2, ζ remains constant over a period and must be calibrated for the project under analysis.

Assumption 3: The resistance to a decrease in the failure intensity opposes, is proportional to the the velocity of failure intensity, and inversely proportional to the overall quality of the test phase, for an appropriate constant ξ.

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

This assumption implies that the faster one tries to reduce the failure intensity the more likely one is to make mistakes leading to possible slowdown in the overall test process. A physical dashpot can be used to explain this behavior. The coefficient of viscosity of the liquid inside the dashpot is $\frac{1}{\gamma}$. Therefore, a small coefficient of viscosity is analogous to a carefully conducted test phase and thus the number of new errors inserted is small. Larger coefficient of viscosity is analogous to the test phase in which more errors are introduced than would be introduced under normal circumstances. The velocity component in the dashpot is analogous to the failure intensity reduction velocity ($\dot{\lambda}$). Thus, the overall quality of the test phase, denoted by (γ), and the rate at which failure intensity is decreasing, determines the failure intensity reduction resistance effort which is analogous to the damping force generated by the dashpot. In Eqn. 3, ξ is merely a constant of proportionality.

Combining Eqs. 1, 2, and 3 in a force balance equation ($-e_f + e_r = e_n$) and organizing it in a state variable format ($\dot{x} = Ax + Bu$) leads to the following system of equations.

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

$$\begin{bmatrix} \lambda(t) \\ \dot{\lambda}(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda(t) \\ \dot{\lambda}(t) \end{bmatrix} \tag{5}$$

`Model S` shown in Figure 1 consists of Eqs. 4 and 5. A solution to these equations, with appropriate estimates of

parameter values, generates $\lambda_i^e(t)$ in Figure 1. Using the relationship in Eq. 6, one computes $R_i^e(t)$.

$$R(\tau) = e^{-\lambda(t)\tau} \tag{6}$$

2.1 Estimation of model parameters

w_f is relatively easy to compute as it is defined to be the number of testers testing the product. The value of w_f must be adjusted for any part-time and temporary personnel. Parameters s_c and γ are computed by applying a convex combination [12] of available metrics. The remaining parameters are estimated through the use of System Identification [13] techniques [14]. An important characteristic of our approach is that estimates of all parameters are updated at each checkpoint thereby improving their accuracy with the passage of time. Changes in the test environment, such as in the workforce and the quality of the STP, are accounted for as and when they occur.[14]

2.2 Computing $\Delta w_f'$ and $\Delta\gamma'$

In a feedback control system, the largest eigenvalue of the system determines the slowest rate of convergence and dominates how fast the output variables converge to their desired values. Therefore, in order to control $\lambda(t)$ so that it reaches its desired value by the deadline t_n, we must adjust the largest eigenvalue appropriately. Given $\lambda(T)$, the failure intensity at time T, and $\lambda(T+\Delta t)$, the desired failure intensity at time Δt later, we use the following equation to determine the amount by which to adjust the largest eigenvalue ς_{max}.[2]

$$\lambda(T+\Delta t) \;=\; \lambda(T)\, e^{-\varsigma_{max}\,\Delta t} \tag{7}$$

We know the values of $\lambda(T)$, $\lambda(T+\Delta t)$ and Δt at checkpoint $\mathrm{cp}_i, i > 0$ and hence we solve Eq. 7 and find the value of ς_{max}. The eigenvalues of a system are defined by the roots of the characteristic polynomial ($\Pi_A(\varsigma) = det[\varsigma I - A]$). Computing the characteristic polynomial of our model leads

$$\begin{aligned} det[\varsigma I - A] &= det\begin{bmatrix} \varsigma & -1 \\ \frac{\zeta\,\hat{w_f}}{s_c^{(1+b)}} & \varsigma + \frac{\xi}{\hat{\gamma}\,s_c} \end{bmatrix} \\ &= \varsigma^2 + \frac{\xi}{\hat{\gamma} s_c}\varsigma + \frac{\zeta \hat{w_f}}{s_c^{(1+b)}} \end{aligned} \tag{8}$$

where $\hat{\gamma} = \gamma + \Delta_\gamma$ and $\hat{w_f} = w_f + \Delta_{w_f}$. Using Eq. 8 we compute the variations in the work force and in the quality of the process necessary to meet the desired quality objective by the deadline.

[2]The symbol λ is used in math literature to represent the eigenvalues of a system. λ is also used in the Software Reliability to represent failure intensity. To avoid confusion and keep uniformity with Software Reliability standards we decided, in this paper, to represent the eigenvalues of system by the symbol ς.

3 Modeling Unforeseen Perturbations

3.1 Disturbances

It is assumed that at the start of the reliability growth testing, a test manager is given a failure intensity objective for the end product and the deadline by which this objective must be met. We use failure intensity as the failure intensity objective; the reliability itself can be computed directly from the failure intensity.[9] A model to help the test manager control the reliability growth testing using feedback has been proposed [1] and is summarized in Eqn. 4 and 5. In this model, the size of the test team (w_f) and the quality of the test process (γ) are the two control variables available for manipulation by the test manager to control the progress of the system test phase towards the desired objective. However, the system test process is subject to various disturbances described in the following section.

3.2 Modeling disturbances

Input signals are, in general, used to drive a system from one state to another. The parameters γ and w_f are the driving forces of the state variable model for the STP. Eqn. 4 indicates that these parameters are not part of the input. The use of γ or w_f as an input would place the system in a nonlinear space making it difficult to apply the techniques from control theory and producing a relatively difficult to understand model. Furthermore, changes in w_f and γ are usually not as frequent as are the unforeseen perturbations modeled by F_d. This observation further justifies the application of parametric control and the use of F_d as the input signal for `Model R`.

Any function can be used to specify the input signal. A square or sinusoidal waive with different frequencies, an exponentially shaped, and a constant value input are a few examples. Though all types of signals are of some concern to the STP, in this work, we focus on four different types of input signals presented in Figure 2. The relationship between these four input signals and actual events in the STP is described next.

3.3 Impulse input

An impulse input is an instantaneous stimulus to a system to drive it from state x_0 to state x_1. This stimulus is represented mathematically by a Dirac delta function that differs from zero for an instant but whose integral over time is unity. Eqn. 9 and Figure 2 are, respectively, a mathematical and a pictorial representations of an impulse.

The question of interest to us is: "What disturbance in an STP can be modeled reasonably by an impulse?" A hardware failure due to power outage prior to the overnight

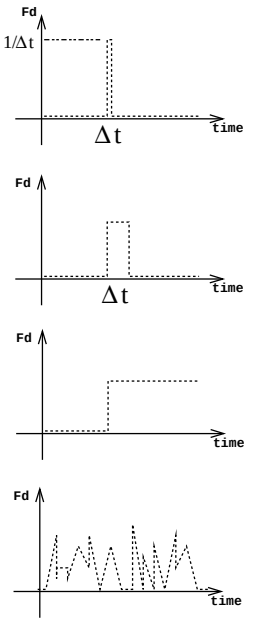

Figure 2. Different types of inputs to represent unforeseen perturbations in a STP.

backup that causes the loss of important files for one day, seems to be a reasonable disturbance to be model by an impulse. Suppose that at the beginning of the day the failure intensity is $\lambda(t_0) = v_0$ and at the end of the day, prior to the power outage, the failure intensity is $\lambda(t_1) = v_1$, for $v_1 < v_0$. An instantaneous stimulus due to the power outage will drive the system from state $\lambda({t_1}^-) = v_1$ to $\lambda({t_1}^+) = v_0$.

Another example of an impulse input for the STP is the replacement of a pre-tested component of the software product by a different, though supposedly functionally equivalent, component. Such replacement may occur due to a need to improve the performance of the application under test. Asuming that most defects in the old component have been removed, this replacement, which serves as an instantaneous stimulus, will likely increase the failure intensity from the current level $\lambda({t_1}^-) = v_0$ to a higher level $\lambda({t_1}^+) = v_0 + \lambda_n c$, where $\lambda_n c$ is an estimate of the failure intensity of the new component. One might argue that the new component may be more reliable than the one it replaced. However, we assume that this is not likely since both components were developed under similar circumstances and the second component had to meet the performance requirements. Therefore, in this case, we need to compute the input F_d in Eqn. 4 that will drive the system from $\lambda({t_1}^-) = v_0$ to $\lambda({t_1}^+) = v_0 + \lambda_n c$. The next subsection addresses this issue. In the remainder of this paper we use the scenario of this second example as the cause of an impulse input signal disturbing the process.

Computing the response to an impulse input

The general format of an impulse input is presented in Eqn. 9. Since the values of the Dirac-delta function are known, we need to compute the vector $[\xi_0 \ \cdots \ \xi_{n-1}]$ of coefficients of $\delta^{(i)}(t)$.

$$u(t) = \sum_{i=0}^{n-1} \xi_i \delta^{(i)}(t) = \xi_0 \delta(t) + \cdots + \xi_{n-1} \delta^{(n-1)}(t) \quad (9)$$

where $\delta(t)$ is the Dirac-delta function and $\delta^{(i)}(t)$ is the i^{th} derivative of $\delta(t)$.

Theorem 1 (Characterization of the Solution)[6]: For the time invariant system dynamics $\dot{x} = Ax + Bu$ and a given $x(0^-)$, suppose that the input assumes the format as described in Eqn. 9. Let $Q = [B \ AB \ \cdots \ A^{n-1}B]$ be the controllability matrix. Then

$$[x(0^+) \ - \ x(0^-)] \quad = \quad Q \times [\xi_0 \ \xi_1 \ \cdots \ \xi_{n-1}]^T \quad (10)$$

Theorem 1 relates $[\xi_0 \ \cdots \ \xi_{n-1}]^T$ to the values of $x(0^+)$ and $x(0^-)$, and the controllability matrix Q, but it does not assert the existence of the vector $[\xi_0 \ \cdots \ \xi_{n-1}]$. The existence of a solution is addressed in the following corollary.

Corollary [6]: Let $Q = [B \ AB \ \cdots \ A^{n-1}B]$ be $n \times n$. For each $x(0^+)$, there exists a unique impulse input, defined in Eqn. 9, which will drive the $x(0^-)$ to $x(0^+)$ if and only if $det(Q) \neq 0$.

The MATLAB function in Figure 3 is used to compute the vector of coefficients $[\xi_0 \ \ \xi_1]^T$ for the impulse input for the system described in Section 2.

```
function ret = Comp_Impulse(sc,zeta,wf, ...
                            b,chi,gamma,x0,x1)
   A(1,1) = 0 ; A(1,2) = 1 ;
   A(2,1) = -(wf*zeta)/(scˆ(1+b)) ;
   A(2,2) = -(chi)/(sc*gamma)  ;
   B = [ 0 1/sc]';
   D = [0];
   C = [ 1 0 ] ;
   sys = ss(A,B,C,D) ;
   Q = ctrb(sys) ;
   X = x1 - x0 ;
   ret = pinv(Q)*X ;
return
```

Figure 3. MATLAB function for the computation of $[\xi_0 \ \ \xi_1]^T$ that composes the impulse input to drive the system from state x_0 to x_1.

3.4 Pulse input

The difference between an impulse and a pulse signal is their respective durations. Whereas Δt is a finite non-zero quantity for a pulse, it tends to 0 for an impulse. The frequency of a pulse might also vary. Though the study of the response of `Model R` to pulses of different frequencies is an important exercise, we focus on a single pulse of fixed length.

A pulse may be used to model an unexpected one-half day training session for the entire or part of the test team. In this case, the disturbance is assumed zero prior to time $t' > 0$, i.e. before the start of the training session. F_d assumes a positive non-zero value at t'. This value of F_d is kept constant over the entire duration of the training after which F_d is reset to zero. The structure of a single pulse input signal is depicted in Figure 2.

Another example of a disturbance that is modeled as a pulse is the migration of the product from the developer's environment to the user's environment. Suppose, for example, that the migration period is one week. The test process does not progress while the system is under migration and a pulse is an appropriate model for this delay. The difference between this example and the training session is in the side effects related to the user's environment. That is, the estimated failure intensity will most likely increase when the product is tested in the user's environment.

3.5 Step input

A step input is a pulse with an infinite width. In our study a step input is modeled by setting F_d to zero prior to some time $t' \geq 0$ and setting it to some constant c soon after t'. A step input is shown in Figure 2.

One disturbance in STP that can be modeled using a step input is the increase in the communication and documentation level due to the replacement of the test manager. Assume that the new test manager requires additional regular meetings and demands more effort in collecting data and documenting the process. Though these changes may increase the overall quality of the process, they may also slow it down. There is a tradeoff in how much can communication and documentation increase without slowing down the process. In this case we assume an over-$\{$communication/documentation$\}$ resulting in deceleration.

3.6 White noise

White noise is a random input signal that never repeats and has a flat frequency spectrum. Usually, there is a large number of sources of disturbances in a STP. A combination of disturbances generated by these sources is modeled as white noise. Sources of disturbances that are assumed to combine into a white noise include personal problems such as illness, software/hardware problems related to the environment in which the test is being conducted, and a fatal failure of a critical hardware component. These problems may or may not occur and there is no easy way to predict the frequency or intensity of any of them or their combination. Therefore, a random signal (white noise) seems to be an appropriate model of such unforeseen perturbations in the STP.

4 Results

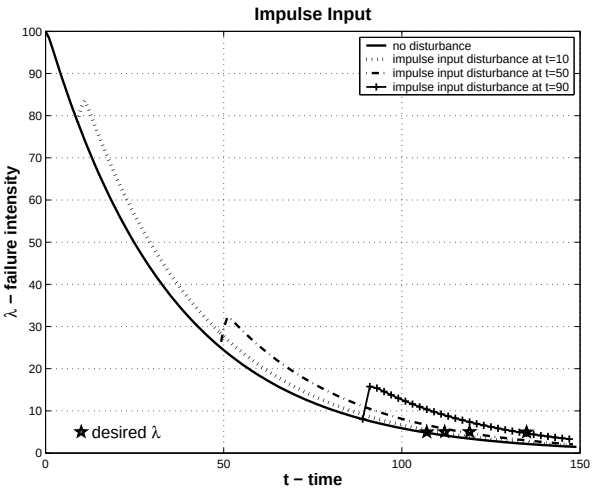

Figure 4. Effect of an impulse input on the failure intensity of a software product. The impulse is modeled so as to cause a 5% increase in $\lambda(t)$ at the time of occurrence. The impulse occurs at the TADs specified earlier.

The impact of various disturbances on the convergence of the failure intensity $\lambda(t)$ was studied by setting F_d in `Model R` to appropriate values and solving the model for $\lambda(t)$. The impact was studied in isolation for each of the four input types, namely impulse, pulse, step, and white noise. Each input was applied (i) early in the test process, (ii) somewhere in the middle of the test process, and (iii) close to the end of the test process. This variation in the time of the application of the input allows us to determine the differences in $\lambda(t)$ due to the different times when the disturbances actually occur during the STP. As the terms "early", "somewhere in the middle", and "close to the end" are fuzzy, we arbitrarily set the times (in days) at which the disturbance is applied to $[10\ 50\ 90]$. We refer to these three times as "time of the application of a disturbance" or, simply, TADs.

Unless stated otherwise, the following parameter values are assumed during the computations: the workforce

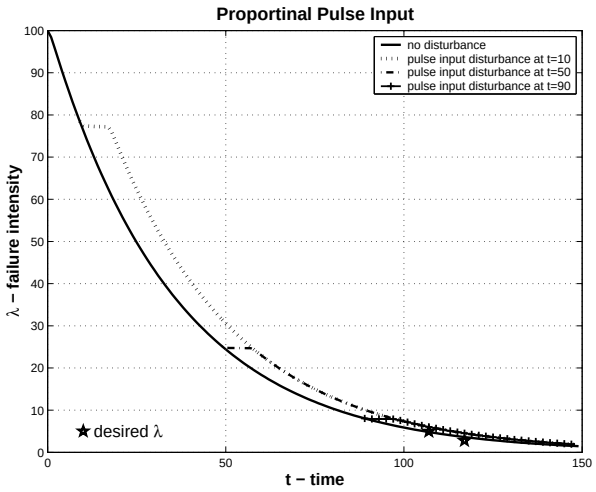

Figure 5. Effect of a pulse input on the failure intensity. The pulse is applied at different TADs. The input at each TAD are equivalent to, respectively, $e_f(10,..,18)$, $e_f(50,..,58)$ and $e_f(90,..,98)$. The duration of each pulse is 8 days.

$w_f = 15$, quality of the test process $\gamma = 0.6$, complexity of the application under test $s_c = 98$, model parameters $\zeta = 45$, $\xi = 117$, and $b = 1.05$. The parameter values have been selected arbitrarily and kept constant in this study. The desired reduction in the failure intensity is measured in terms of percent of the initial value, exact values of the failure intensity are not of concern in this study. It is assumed that the test manager is interested in reducing the failure intensity to 5% of its initial value. Thus, for example, if $\lambda(0) = 20$ failures/day then $\lambda(t_d) = 1$ failure per day, where t_d denotes the deadline by which the system test is to be completed.

4.1 Impulse input

An impulse input models, for example, the replacement of an already tested component by an untested one. We assume that such a replacement causes an increase of 5% in $\lambda(t)$ soon after the disturbance occurs. Eqn. 9 is a precise model of the impulse input. To obtain the value of $\lambda(t)$ in response to the impulse input we need to compute the vector $[\xi_0 \ \xi_1]^T$ to drive the system from its current state $x(t^-) = [r(t) \ v(t)]^T$ to $x(t^+) = [r(t) + 5 \ \ 0]^T$. Computing these values using the script from Figure 3 results in the following impulse inputs for $t = [10\ 50\ 90]$.

$$
\begin{aligned}
u(10) &= 1.1943e^3 \times \delta(10) + 0.49e^3 \times \delta'(10) \\
u(50) &= 1.0451e^3 \times \delta(50) + 0.49e^3 \times \delta'(50) \\
u(90) &= 997.4383 \times \delta(90) + 490 \times \delta'(90)
\end{aligned}
$$

Figure 4 shows the effects of various impulse inputs on $\lambda(t)$. As observed from this figure, the expected time to reduce $\lambda(t)$ to 5% of the value at the start of the test process is 107 days. However, when an impulse occurs at time $t = 10$ the desired reduction takes place with a delay of 5 days. The corresponding delays in the reduction of $\lambda(t)$ to 5% of its initial value are, respectively, 12 and 28 days, for $t = 50$ and $t = 90$.

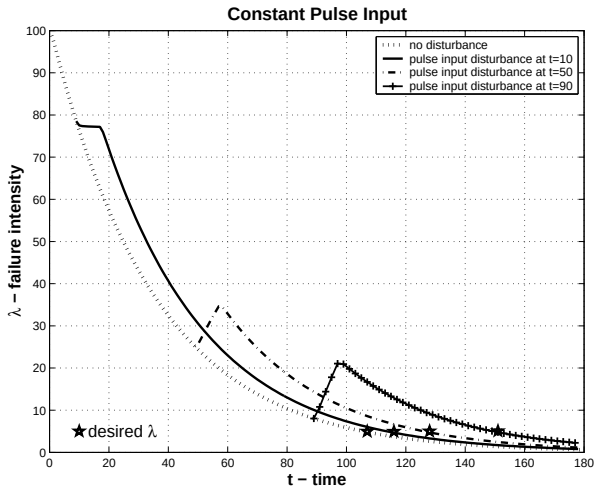

Figure 6. Results of the perturbation of the systems by an pulse input signal at TADs. The input signal generated is equivalent to $e_f(10)$ for all the three time instances and it persists for 8 days.

4.2 Pulse input

Two examples of disturbances in STP that can be modeled by a pulse are presented in Section 3.4. Next we determine the impact of these disturbances on $\lambda(t)$. As shown in Figure 5, a training period of 8 days for the entire test team delays the progress of the test process by the same number of days. The value of F_d is computed by generating an equivalent, though opposite, force to the effective test effort e_f, i.e., $F_d(t) = -e_f(t),\ t = i, \cdots, i+8;\ i = \{10, 50, 90\}$. Notice that the absolute value of F_d is not a constant, but its relative value is proportional to the value of e_f.

Unlike the behavior exhibited in Figure 5, the values of F_d in Figure 6 remain constant over a period of 8 days. For

the three time instances used, F_d is computed as the equivalent of $e_f(10)$, i.e., $F_d(t) = -e_f(10),\ t = i, \cdots, i+8;\ i = \{10, 50, 90\}$. Therefore, the absolute value of F_d is the same for the three periods and the respective delays to achieve the same goal are 9, 21 and 44 days as is observed from Figure 6.

4.3 Step input

An increase in the communication overhead and documentation is modeled as a step input. The step input is applied to the STP at TADs specified earlier. Two different ways are used to to generate the step input. The first way represents an increase in communication equivalent to 60% of the effective test effort, i.e., $F_d(t) = -0.6 \times e_f(t),\ t = i, \cdots, \infty;\ i = \{10, 50, 90\}$ and is named "proportional step." The effects of this step input on $\lambda(t)$ are shown in Figure 7 where delays of 43, 33, and 16 days are observed, respectively, for the three TADs.

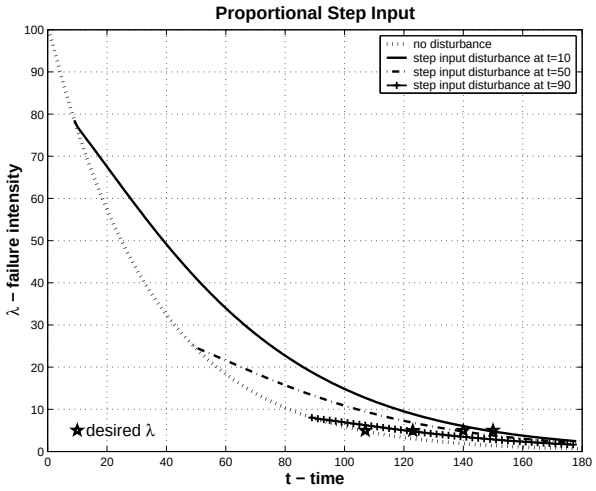

Figure 7. Results of the perturbation of the systems by an step input signal at TADs. The input signals for these time instances are equivalent to $e_f(10, ...)$, $e_f(40, ...)$ and $e_f(60, ...)$, respectively.

The second way to generate the step input is to use an absolute step input. In this case, F_d is computed as a force equivalent to 15% of the effective test effort at time $t = 10$, i.e., $F_d(t) = -0.15 \times e_f(10),\ t = i, \cdots, \infty;\ i = \{10, 50, 90\}$. The effects of step input on $\lambda(t)$ are shown in Figure 8.

4.4 White noise input

We model white noise by setting $F_d(t) = -\alpha(t) \times e_f(t),\ t = i, \cdots, \infty;\ i = \{10, 50, 90\}$. Parameter α is

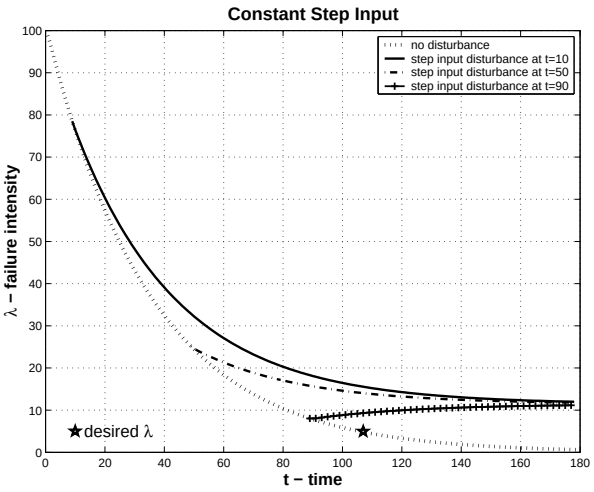

Figure 8. Results of the perturbation of the systems by an step input signal at TADs. The input signal generated is equivalent to $e_f(10)$ for all the three time instances and it persists for the remaining period.

distributed normally over the range $0 \ldots 2$. The mean value of α fluctuates around 1 with a standard deviation of approximately 0.4 for each of the three cases. The effects of applying a white noise at TADs are shown in Figure 9. The delays associated with each of the three cases is 60, 47, and 24, respectively.

5 Analysis

We now analyse the results presented above. Of primary concern is the delay associated with each type of input. Any side effects associated with the input signal are also considered. The delays due to various disturbances are summarized in Table 5.

Impulse input

We observe from Figure 4 that the impact of an impulse input that drives $[\lambda(t^-)\ \dot{\lambda}(t_t)]^T$ at t^- to $[\lambda(t^+) + 5\ 0]^T$ increases with the TAD. To explain why suppose that component C_o is replaced by component C_n. During the early phase of the test cycle it is likely that C_o and its interface with the other components has not been tested. Thus, it is reasonable to assume that the change in failure intensity due to the replacement of C_o by C_n will be only due to the possibly poor quality of C_n leading to a 5% increase in the failure. However, later in the process, additional testing has occurred and the failure intensity of the application is likely to be much less than what it was during the early part of

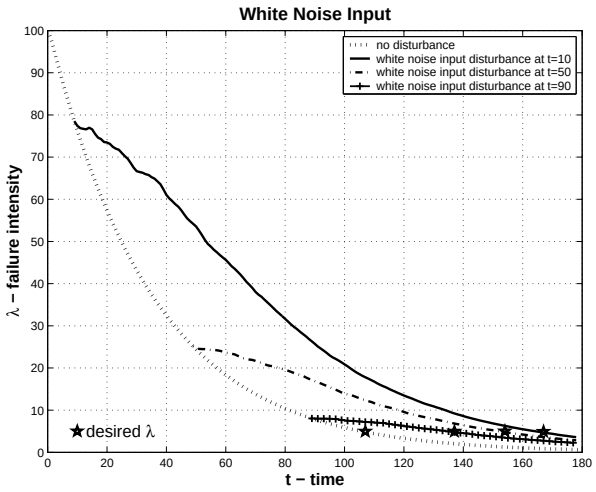

Figure 9. Effects of white noise on the failure intensity. $F_d(t) = -\alpha \times e_f(t)$, where α is distributed normally over $0 \ldots 2$.

Table 1. Delay, measured in "days", in the convergence of the failure intensity to 5% of its initial value. The disturbances occur at times $t = \{10, 50, 90\}$. The expected time for convergence without any disturbance is 107 days.

Type of Disturbance	Delay		
	$t = 10$	$t = 50$	$t = 90$
Impulse	5	12	28
Proportional Pulse	9	9	9
Constant Pulse	9	21	44
Proportional Step	43	33	16
Constant Step	73	73	73
White Noise	60	47	24

the test cycle. Thus, relacement at this time causes a larger increase in the failure intensity and is due to failures associated with C_o and with the interfaces between C_o and the remainder of the system.

Figure 10 shows the delay in the convergence of the failure intensity to its desired value which is 5% of its initial value. The delay increases with the time at which the impulse is applied. Eventually the failure intensity does converge to its desired value as in Figure 10(a). Replacement of a component when $\lambda(t)$ is close to zero leads to an increase in $\lambda(t)$. The time to get $\lambda(t)$ back to its value prevailing just before this increase is nearly constant and explains the eventual convergence in Figure 10(a). The later the replacement of a component the larger the interface failures introduced leading to the overshoot behavior in Figure 10(b). Similar argument justifies the stabilization of the delay.

Pulse input

As pointed out earlier, the pulse is represented in two different ways. The delay due to proportional pulse input does not change with TAD. To explain why, suppose that such disturbance is due to a training period of 8 days for the entire test team. The training will likely delay the process by the same number of days but will not increase or decrease the failure intensity. Therefore, the time when it occurs does not impose any side effect on the test process as can be noticed from Figure 5. Note that training will likely increase the quality of the test process (γ) and thus speed up the convergence of $\lambda(t)$ to its desired value. However, in this study we retain γ to its original value and hence the proportional pulse causes a constant delay regardless of the time of its occurrence in the STP.

The constant pulse input is related to the migration of the system from the test environment to the user's environment. In our study we assumed a total of 8 days for the migration. As seen in Table 5, the delay in convergence is affected by TAD. This is due to the side effects during migration. During migration, at the beginning of the test process, most of the features related to the environment have not been tested and hence the delay in the completion of the test is proportional to the migration time, which is 8 days in this case. If the migration occurs in the middle of the process, i.e. at $t = 50$, some of the environment features have already been tested and an increase in λ results due to the differences between the two environments that exist before and after the migration. This increase in λ results in an increase in the delay as noticed in Figure 6. Any increase in λ is considered an overshoot because we assume that the migration per se does not affect λ.

A behavior similar to the one presented by the impulse input in Figure 10 is observed in Figure 11. The delay increases with TAD until it stabilizes at a certain level as depicted in Figure 11(a). Figure 11(b) is the counterpart of Figure 10(b) for the impulse input. The differences are not in the shape of the curve, but in the values. The delays for the impulse and pulse inputs stabilize at 158 and 187 days, respectively. Also, the overshoot associated with the pulse reaches a maximum value of 15.62, almost five times larger than the overshoot of 3.29 for the impulse input. This behavior is consistent with what one might expect and is due to the longer persistence of the pulse when compared to that of an impulse input.

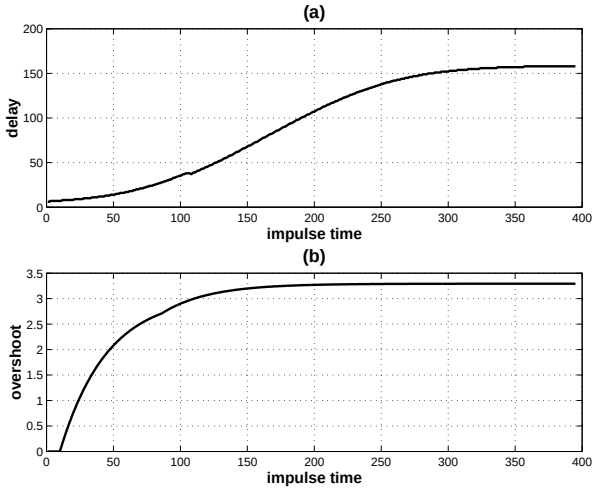

Figure 10. (a)-Delay associated with an impulse input stimulating the system at a certain time t. (b)-Overshoot associated with an impulse input stimulating the system at a certain time t.

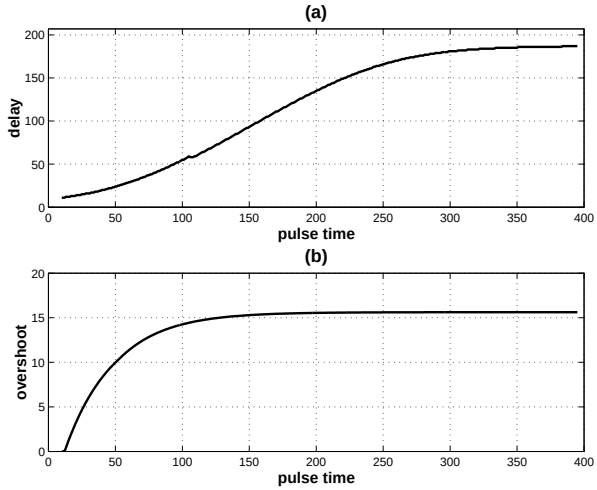

Figure 11. (a)-Delay associated with a pulse input stimulating the system at a certain time t. (b)-Overshoot associated with a pulse input stimulating the system at a certain time t.

Step input

In our study the step input models an increase in the communication overhead. For the proportional step input, $F_d(t)$ assumes a value proportional to $\alpha \times e_f(t)$ and inflicts a large delay when it occurs early in the process because it persists throughout the STP. The delay associated with a step input starts at 43 days even though the input occurs at $t = 10$. The delay decreases to 33 and 16 days, respectively, when the step input occurs at $t = 50$ and $t = 90$. Figure 12 shows the decrease in convergence delay as TAD moves towards the end of the STP. We are aware of the tradeoff in increased communication and here we consider the level of communication has crossed the border of being beneficial. There is no overshoot due to a step input as one would not expect any increase in the failure intensity due to an increase in communication overhead.

A saturation effect can be observed in Figure 8 when the step input assumes a constant value. There is no condition linked to the increase of the communication level that would lead to a saturation in the test process. It is well known that such behavior is related to the criteria used and the quality of the test process. It is inappropriate to model the communication overhead using a constant value step input.

White noise input

Figure 9 shows that the delay associated with TAD of a white noise input ranges from 0 to 2 times the value of the effective test effort. As expected, the earlier the noise starts the longer the delay. The delay is 60 days when the noise signal is applied at $t = 10$ and drops to 47 and 24 days, respectively, for $t = 50$ and $t = 60$. However, we believe that the disturbances that can be modeled as white noise are more prevalent during the early part of the STP than during its later part. Therefore, the frequency of the noise seems to be a more interesting parameter to consider.

Figure 13 shows the delay associated with a white noise input applied starting always at the start of the STP. The x-axis represents how frequently the disturbance modeled by the white noise occurs and ranges from 0 to 1, i.e., from 0% to 100%. As before, the value of the noise is computed as $F_d(t) = -\alpha(t) \times e_f(t)$, for α ranging from 0 to 2. As observed from Figure 13, the delay increases with the frequency of the white noise input. This behavior appears to be consistent with reality.

6 Summary

A study was undertaken to examine the impact of various types of disturbances that might occur during a system test phase on the convergence of the failure intensity of the product under test. Disturbances are modeled as impulse, pulse, step, and white noise inputs to the STP. These inputs are then applied to a state model of the STP, proposed in an earlier work, and the effect observed on the failure intensity. Results confirm the commonly observed phenomenon that the delay in the convergence of the failure intensity to its desired value depends on the type of the input and its time of occurrence. The use of the state model assist in the

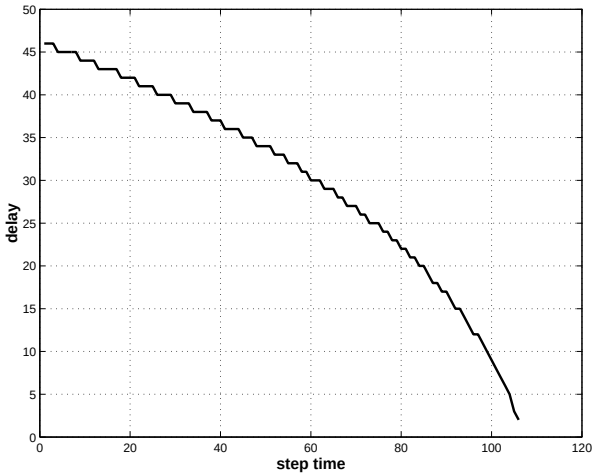

Figure 12. Delay associated with a step input stimulating the system at time t.

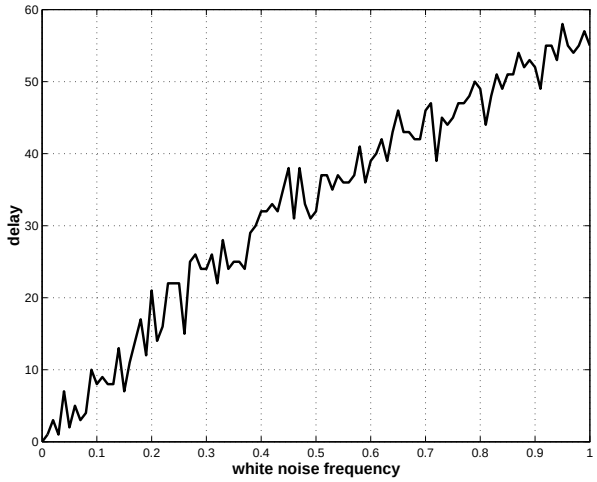

Figure 13. Delay associated with the frequency a white noise occurs during the test process.

quantification of the delays. The study was conducted by isolating various types of disturbances. This allowed us to individually study the impact of each type of input on the convergence of the failure intensity.

References

[1] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur, "Feedback control of the software test process through measurements of software reliability," in *Proceeding of* 12^{th} *International Symposium on Software Reliability Engineering*, (Hong Kong), pp. 232–241, November 2001.

[2] M. Donnely, B. Everrett, J. Musa, and G. Wilson, *Handbook of Software Reliability Engineering, M. R. Lyu, Editor*, ch. "Best Current Practice of SRE", pp. 219–254. New York: McGraw-Hill, 1996.

[3] 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.

[4] G. C. Goodwin, S. F. Graebe, and M. E. Salgado., *Control system design*. Upper Saddle River, New Jersey: Prentice Hall, 2001.

[5] D. G. Luenberger, *Introduction to Dynamic Systems: Theory, models and applications*. John Wiley & Sons, 1979.

[6] R. A. DeCarlo, *Linear systems : A state variable approach with numerical implementation*. Upper Saddle River, New York: Prentice-Hall, 1989.

[7] F. Lanchester, "Aircraft in warfare, the dawn of the fourth arm," Constable, London, 1916.

[8] P. A. Samuelson, "Interactions between the multiplier analysis and the principle of acceleration," *Rev. Economic Statistics*, vol. 21, pp. 75–78, May 1939.

[9] J. Musa, *Software Reliability Engineering*. McGraw-Hill, 1999.

[10] A. P. Mathur, F. D. Frate, P. Garg, and A. Pasquini, "On the correlation between code coverage and software reliability," in *Proceedings of the Sixth International Symposium on Software Reliability Engineering*, (Toulouse, France), pp. 124–132, IEEE Press, October 24-27 1995.

[11] B. W. Boehm and et. al., *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.

[12] S. R. Lay, *Convex Sets and their Applications*. New York: John Wiley & Sons Inc., 1982.

[13] L. Ljung, *System identification: Theory for the user*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[14] 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.