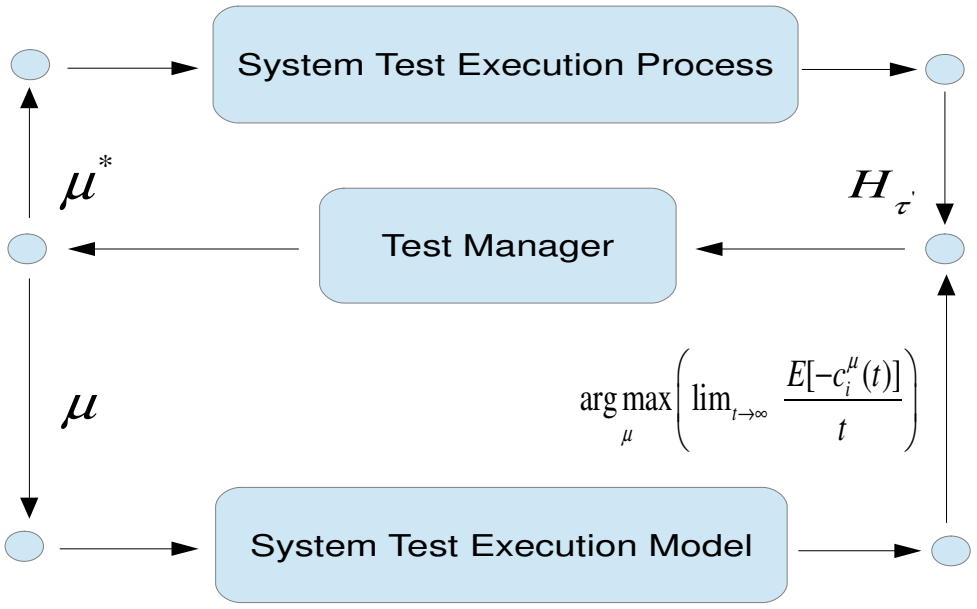
System Test Evaluation and Review Technique

Stochastic Modeling and Optimal Control for System Testing of Software

Author: Aniket Mitra

Supervisor: dr. Sandjai Bhulai



VU University, Amsterdam

Faculty of Sciences

1081 HV Amsterdam

May, 2013

Preface

This work is part of my Master's degree program at VU University, where each student is required to perform a research regarding a specific problem motivated by practice. I would like to extend my gratitude to dr. Sandjai Bhulai whose course in Stochastic Optimization instilled in me the belief that it was mathematically feasible to build a model for System Test Execution and Test Management. His research, ideas, efforts and enthusiasm have helped me enormously in writing this paper. I am grateful to Shunji Osaki and Hisashi Mine for their paper on Semi-Markov Decision Processes which helped me gain deep insight into the subject. Also, I would like to extend my gratitude to Salah E Elmaghraby for his work on GERT and SMP's which was enormously helpful for writing this paper. Finally, to James A Whittaker who pioneered the work on Markov Chain Modeling of Software Testing which in turn became the foundation on which I could perform my research in this area. This paper is my humble effort to extend the research in the field of Modeling of Software Testing and Test Management.

Aniket Mitra
aniketmitra001@gmail.com
May, 2013

Executive Summary

System testing of software is defined as the *"investigation conducted to evaluate whether a complete and integrated software system complies with its specified requirements"*[1]. Thus system testing is a process that requires creation of test cases for every function point of the software and execution of the test cases to validate whether the function point conforms to the specified requirements. In case of failure of a test, a defect is logged that is fixed by the development team and again re-tested. The system testing process can therefore be described by the following steps: (a) Requirements analysis, (b) Test estimation and strategy (c) Test planning (d) Creation of test scripts based on requirements (e) Execution of test scripts on the software product (f) Reporting of defects (g) Retesting of fixed defects and (h) Test Closure

The system testing process described in steps (e)-(g) is a cyclic process and requires allocation of resources (software testers) to complete testing in allocated time. However, as businesses increasingly tend to reduce the time to market of their products and services, coupled with the fact that upstream activities (Requirement Gathering, Infrastructure setup, Development etc) often consume more time that what is allocated, the time allocated to testing is often squeezed. This in turn leads to a tradeoff between resources (costs) and allocated time. This leads to the decision problem that every test manager has to deal with, as to what is the optimal investment in execution of a test case (or a collection of test cases) that in turn ensures that system testing meets the schedule and budget constraints.

In sequential decision making problems, a decision maker or agent chooses consecutive actions according to a system status and her preferences to form a decision policy. The essence of these problems is that the decisions made now can have both immediate and long term effects and the ultimate goal of the agent's actions would be to maximize the expected utility of such decisions. Markov and semi-Markov Decision Processes provide a mathematical representation of these problems. Decisions are made sequentially, obtaining an immediate utility after each decision, which also modifies the environment for future decisions. Semi-Markov Decision Processes generalize the Markov Decision Processes mainly by allowing the actions to be history dependent and by modeling the transition time distribution of each action. Our primary endeavor in this paper is to implement a semi-Markov decision process on the test execution Markov chain and thus build a model that solves the sequential decision problem of software test management

As business managers demand more control over IT expenditures and especially the return on investment in quality assurance activities, the decisions that software test managers make in terms of resources and time (i.e. test management) will need to be much more scientific and not just driven by intuition and experience, though any amount of mathematical modeling will not be a substitute for wisdom, and we certainly believe that the best decisions are made at the confluence of scientific rigor and intuition.

1 IEEE Standard Glossary of Software Engineering Terminology

Table of Contents

Introduction

System testing, as defined by the IEEE standard glossary of software engineering terminology is the *"investigation conducted to evaluate whether a complete and integrated software system complies with its specified requirements"*. *Downs* [6] proposed that test cases should be considered as independent Bernoulli trials and test execution as a sequence of Bernoulli trails on logic paths /function points through the software product. *Downs* [6] derived the failure rate of a software system, the distribution of the number of faults in a path and using these parameters proposed different testing strategies. Finally, the optimum test execution profile of a software product was estimated. However, the interdependence among test cases (and therefore test execution) requires a Markov Chain modeling of system testing as proposed by *Whittaker et al.* [3] wherein the logic paths in a software product can be modeled as a Markov chain and hence the test execution process along these logic paths. A popular model for software project management is PERT. Though easy to compute, PERT suffers from a drawback of not allowing loops between two activities which is very important for testing, as a test might fail multiple times due to improper bug fixing and might have to be executed many times. *Pritsker* [5] proposed a model called GERT (Graphical Evaluation and Review Technique) wherein he showed that a set of interconnected activities that take a random amount of time to complete can be modeled as a semi-Markov process. *Elmaghraby* [1] further showed that GERT can be used as a model for project management of activities which contain loops. *Cangussu et al.*[4], [11] take a system theoretic approach to propose a feedback control model for software test process that takes into account presence of unforeseen perturbations and noise in the data.

The models of system testing referred above do not model test execution as a stochastic process wherein the time required to execute a test case is a random variable defined on a probability space and depends on the next test case to be executed. Therefore the probability of test execution being in a particular state at time T depends on the transition probability of reaching the state and the probability that the state will be reached in time T. Moreover, when test execution on a logic path fails, the software tester may decide to move on to test another logic path with a certain transition probability while the defect in the first logic path is being fixed by the development team. Similarly, execution of a test case might fail and the tester might decide to retest with a certain transition probability depending on the complexity of the test case. Such properties of test execution further necessitate the need to model test execution as a walk (with certain transition probabilities) on a finite graph. Therefore, the stochastic nature of the test execution process requires a stochastic control model that enables a test manager to take optimal decisions regarding cost and time.

The aim of this paper is to model system testing as a semi-Markov process and then extend it to a semi-Markov decision process for the purpose of test management. This model enables us to compute the stationary probability that test execution is in a given state when execution time is a random variable, and the average time to execute a test case. Moreover, the model computes the expected time to complete execution of all test cases in the test suite. The modeling as a decision process allows us to compute the long run average cost of test execution and we use policy iteration to analyze the impact of additional investment on future costs and time. Therefore, based on the history of the system testing process, the test manager, at a decision epoch can analyze how a decision regarding additional investment will impact the budget and schedule in future. This model will enable the test manager to take a decision that minimizes the costs while ensuring that the schedule constraints are met. We also propose a Linear Programming

formulation for computing the optimal additional investment, such that the probability of completing system testing on or before a specified time is maximized.

2. Exponential Distribution and Test Execution Time

Here we consider two important properties of the exponential distribution namely '*a constant rate*' and '*momorylessness*' and try to analyze this in the framework of software testing.

The exponential distribution is characterized by the rate parameter $\lambda \geq 0$ which is the expected number of events that occur per unit of time in a '*continuous time counting process'*. This continuous time counting process $\{N_t, t \geq 0\}$ is also known as a homogeneous Poisson process and satisfies the following properties.

1. The rate parameter λ does not change over time
2. N is a counting process i.e
   - $N_t \in \mathbb{Z}^+ \forall t \geq 0$
   - $\forall t \geq s, N_t \geq N_s$
   - (no two occurrence can occur simultaneously), $\forall t \geq 0, \lim_{s \downarrow t} N_s \leq \lim_{s \uparrow t} N_s + 1$
3. $N_0 = 0$ a.s.
4. (***Independence of increments***) $\sigma(N_t - N_s)$ and $\sigma(N_u, u \leq s)$ are independent i.e the number of occurrences counted in disjoint intervals are independent of each other.
5. (stationarity of increments) $N_t - N_s \overset{\text{d}}{=} N_{t-s} \forall s \leq t$ i.e the probability distribution of the number of occurrences counted in any time interval only depends on the length of the interval

If all the five properties mentioned above are satisfied, then the time between occurrences of two events of a homogeneous Poisson process is exponentially distributed. The independence of increments described in property 5, leads to an important property of the exponential distribution called '*momorylessness*' which can be formally defined as

$$P(T > t + \Delta t \mid T > t) = P(T > \Delta t) \forall t, \Delta t \geq 0 \quad (1)$$

Now let us try to analyze these properties for the software test execution process where N_t denotes the number of test case executed till time '*t'*. Properties (2) and (3) are clearly satisfied and therefore need no further explanation. The rate parameter in testing parlance is the test execution rate which is defined as the average number of test cases executed by a software tester per unit of time. Property (5) is also an acceptable regularity condition for test execution wherein we say that the probability distribution of the number of test cases executed during a time interval only depends on the length of the interval.

Let us now look at property (4) which implies that the number of test cases executed in a particular time interval will be independent of the number of test cases executed in the previous time interval. A critical analysis of the software testing process reveals that this might not necessarily be true as there might occur instances, where the tester speeds or slows down the test execution process based on the number of

test cases executed in the previous disjoint interval. Speeding up the process does not have a major impact because the number of test cases that can be executed is constrained by the skill set of the tester and which does not vary during the duration of test execution. However, slowing down when the tester has executed more than the average number of test cases in the previous interval and decides to ease up, renders the process with memory. But, testing projects, like any other activity, are subject to '*student syndrome*' and such cases are rare. Another scenario that instills memory in to the test execution process is that as test execution progresses, the software tester gains a better understanding of the software under test and thus if she finds similar test cases (to already executed ones), she executes them quicker. This is also true for retesting a failed test case, wherein the time to retest a failed test case is quicker as the tester already is familiar with the test case. We therefore model the test execution process as a semi-Markov Process in this paper and present an algorithm in section 5.3 to simulate semi-Markov process when the test execution time is not exponentially distributed. However, there can be many testing projects where the scenarios like the ones mentioned above that instill memory to the test execution process can be safely ignored and in such cases property (4) is satisfied. Therefore, if we assume that the test execution processes is memoryless, then test execution can be modeled as pure birth-death process where the lifespan of a test case is described as follows. A test case is active till the time

a) It passes or
b) It fails

Now, we turn our attention to property (1).In the case of manual testing, the execution rate is primarily a function of the skills of the software tester and generally does not fluctuate during the course of test execution. However, we can have different execution rates for different types of test cases which can be classified as simple, medium or complex. This can be easily tackled by modeling the execution of each classification of test cases as independent Poisson processes with different rates $(\lambda_S, \lambda_M, \lambda_C)$. We define the set of Poisson processes $N_t^S, N_t^M, N_t^C \forall t \geq 0$ (that satisfy property (2), (3),(4) and (5)) to be independent if for $0 < t_1 < t_2 < ... < t_k, \forall k \in \mathbb{Z}^+$,the random variables $N_{t_1}^S, N_{t_2}^S, ..., N_{t_k}^S; N_{t_1}^M, N_{t_2}^M, ..., N_{t_k}^M$ *and* $N_{t_1}^C, N_{t_2}^C, ..., N_{t_k}^C$ are independent of each other. Therefore, we can combine the three independent Poisson processes to form a Poisson process N_t with rate $\lambda = \lambda_S + \lambda_M + \lambda_C$ Therefore the test execution process satisfies property (1) as well. Thus, with certain assumptions regarding the independence of the number of test cases executed in disjoint time intervals, we can conclude that the time to execute a test case is exponentially distributed.

3. SMC as a model for System Testing

The SMC model is the de facto model used by most test managers to estimate the time required to complete test execution of *'N'* test cases in the test suite by segregating the test cases into three complexity classes (Simple, Medium and Complex), each type having its own execution rate and then taking a weighted average. Let us consider a basic test suite of 16 test cases with 8 test cases of complexity as simple, 4 test cases with complexity as medium and 4 test cases with complexity as complex. The average execution rate per hour λ of each type of test cases is given in the Table 1. As per the SMC model the expected execution time of the 16 test cases would be 260 minutes (4 hours 20 minutes)

Now let us model the Test Execution Process as a homogeneous Poisson process where the test execution time of each test case is exponentially distributed. The first point to note is that the SMC model is just a linear combination of the average rate of execution and does not take into account that the test execution time is a random variable and therefore there is a certain probability associated with the event that execution of a test case will complete within the expected time and also a certain probability associated with the event that test execution will not complete in expected time.

So when expected execution time per test case is $\frac{1}{\lambda}$(because the execution time per test case is exponentially distributed) , the probability of completing execution of a test case by the expected time $\frac{1}{\lambda}$ is given by $P(T \le \frac{1}{\lambda}) = (1 - e^{-\lambda\frac{1}{\lambda}}) = (1 - \frac{1}{e}) = 63.21\%$, while the probability of not completing the execution of a test case by the expected time is $P(T > \frac{1}{\lambda}) = 1 - P(T \le \frac{1}{\lambda}) = \frac{1}{e} = 36.79\%$ and what is important is these probabilities are independent of the complexity of the test cases as well as the execution rate of the test cases belonging to each complexity class. What makes the situation even worse is that the probability that all 16 test cases in the test suite will complete in expected time is $(63.21\%)^{16}$ =0.065% which is an alarmingly low number even for a very small test suite , and will tend to 0 as 'N' gets large.

Table 1: Average test execution time and probability of completing test execution (not) on schedule for test cases belonging to different complexity classes

Complexity	Average Execution Rate per hour(λ)	Average Execution Time per test case, $t = \frac{1}{\lambda}$	Probability of completing execution in time $P(T \le \frac{1}{\lambda})$	Probability of not completing execution in time $P(T > \frac{1}{\lambda})$
S	6	10 min	63.21%	36.79%
M	4	15 min	63.21%	36.79%
C	2	30 min	63.21%	36.79%

Next, we sampled the test execution time from an exponential distribution with parameter $\lambda = 6, 4, 2$ and 12 respectively by using the inverse transform sampling method and plot the figure below. As can be seen from Figure 1, the exponential distribution has fatter tails as λ decreases, which means that the there is a large probability of not being able to complete test execution in the expected time when the average execution rate is low. In other words, this reinforces the intuition in practice that execution of complex test cases has a higher chance of not being completed in expected time. Another interpretation can be that when the testing skills of the software tester is low (and in turn λ is low), the chances of not being able to complete testing in the expected time is quite high which is in line with the intuition in practice.

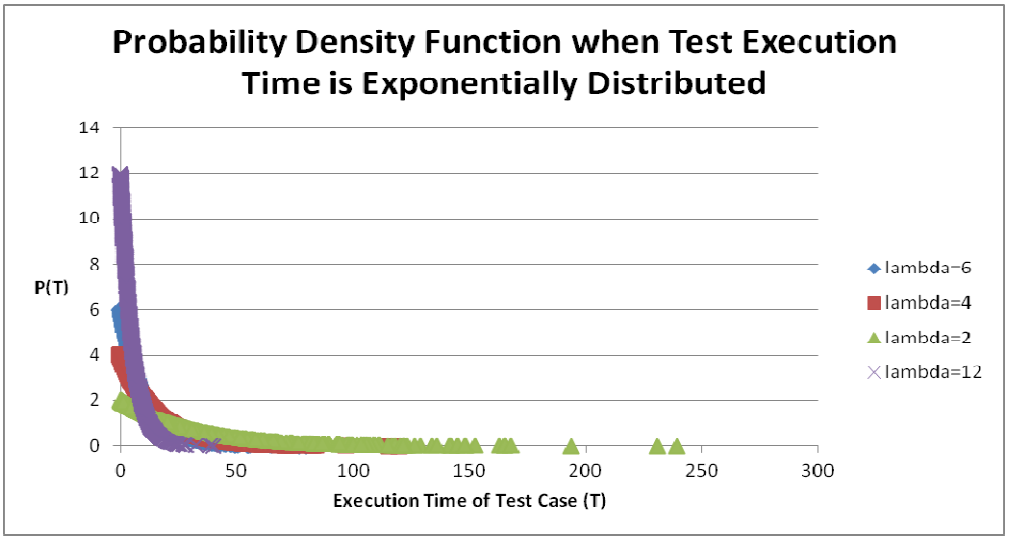


Figure 1: Probability that execution of test cases will complete on or before time, T for different execution rates, λ

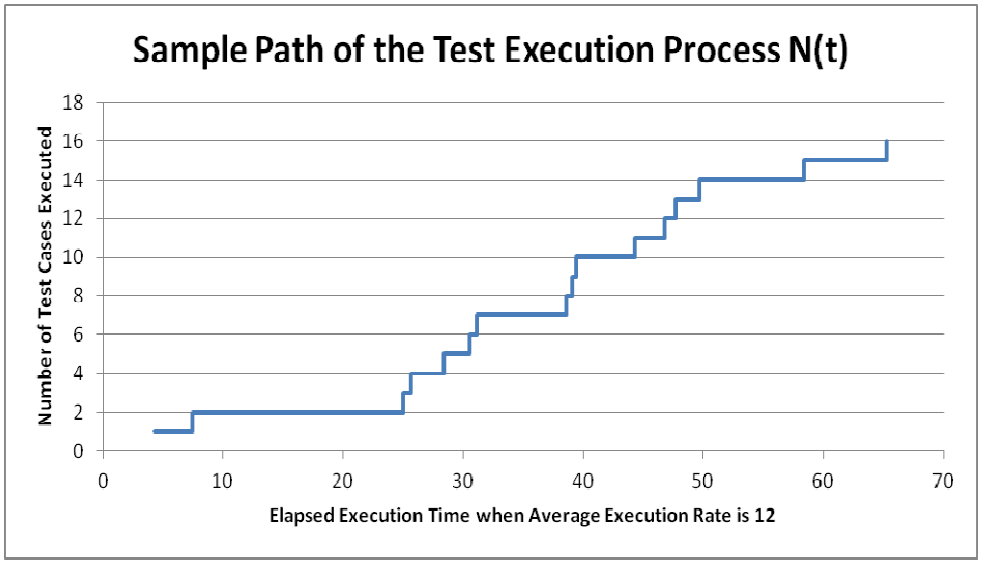


Figure 2: Sample Path of Number of Test Cases Executed v/s Elapsed Test Execution Time

Combining the independent Poisson processes into a Poisson process with rate $\lambda = 12$, ensures that the probability of not completing test execution within the expected time is much lesser than the probability of not completing test execution for each of the individual Poisson processes with rate $\lambda = 6, 4, 2$ respectively. This conclusion is again in line with the belief in practice that *diversification* or *economies of scope* reduce the risk of schedule overrun. Thus, we can conclude that though the modeling of test execution as a homogenous Poisson process requires us to make a simplifying assumption that the process

is memoryless, the model is a very good approximation of test execution in practice and is more accurate than the SMC model currently used. This insight opens up the path for modeling of software test execution as a Markov chain which we treat in the next sections.

4. Markov Chain Modeling of System Testing

We use the TDLR (Top Down Left Right) naming convention for numbering test cases as introduced by Lavenberg and Shedler [8]. Let us denote the number of test cases in the test suite by $N \in \mathbb{N}$ Therefore the state space $\aleph = \{0,1,...N\} < \infty$ .Let ***i, j***=1,..,N denote the current state/test case that is being executed and the state/test cases immediately connected to the current test case . In case of a failure of a test the system stays in the current state i.e *i=j* or moves to the test case in the next branch. The transition probability $p_{ij} \in \mathbb{R}^+; p_{ij}(0) = 0; \sum_{j=1}^{N} p_{ij}(\infty) = 1$, is the probability of transitioning from test case *'i'* to test case *'j'* and P is the $N \times N$ transition probability matrix. Moreover we impose the conditions that $p_{ij}^{(n)} > 0, \forall i, j$ and $p_{ij} < 1, \forall i = j$. Let the stochastic process $\{X_t \in \aleph; t \geq 0\}$ denote test execution status wherein $X_t = i$ denotes that test case *'i'* is being executed at time*'t'*. A stochastic process $\{X_t \in \aleph; t \geq 0\}$ is called a Markov Chain if for all times $t \geq 0$ and all states $i, j \in \aleph$,

$$P(X_{t+1} = j \mid X_t = i, X_{t-1} = i-1, ..., X_0 = 1) = p_{ij} \quad (2)$$

This means that the probability of transition from test case *'i'* to test case *'j'* at any time *'t'* depends only on being in test case *'i'* at that particular time *'t'*, but its independent of the history of arrival at test case *'i'* .Test execution along a given logic path means that the execution of the next test case *'j'* at time *'t+1'* is dependent on the current test case *'i'* and *'i'* has absorbed all the information up to time *'t'*. So a walk along the test chain when test cases keep passing satisfies the Markov property. It is however important to consider the case when a test case fails. Once a defect is fixed (after the test case failed), the test case that had lead to the identification of the defect is re-executed to validate the fix. Now we can assume that this test case behaves like a new test case (though this is not true in practice as the time required to perform a test is a decreasing function of the number of retests and therefore has memory, but this is not the case for automated testing where test execution again satisfies the memoryless property). Thus we assume that the test execution satisfies the Markov property though we relax this assumption in section [5.3], wherein we present an algorithm for simulating a semi-Markov process where holding times do not have an exponential distribution.

The transition probability p_{ij} also depends on the complexity of the test case i.e complex test case will have the lowest transition probability to the next test case in the logic path and the highest transition probability to itself. On the other hand, a simple test case will have the lowest transition probability to itself and the highest transition probability to the next test case in the logic path. This is explained by the fact the complex test cases have the highest probability of failure while simple test cases have the lowest probability of failure.

A Markov chain model for test execution of sixteen test cases is shown in Figure 3.

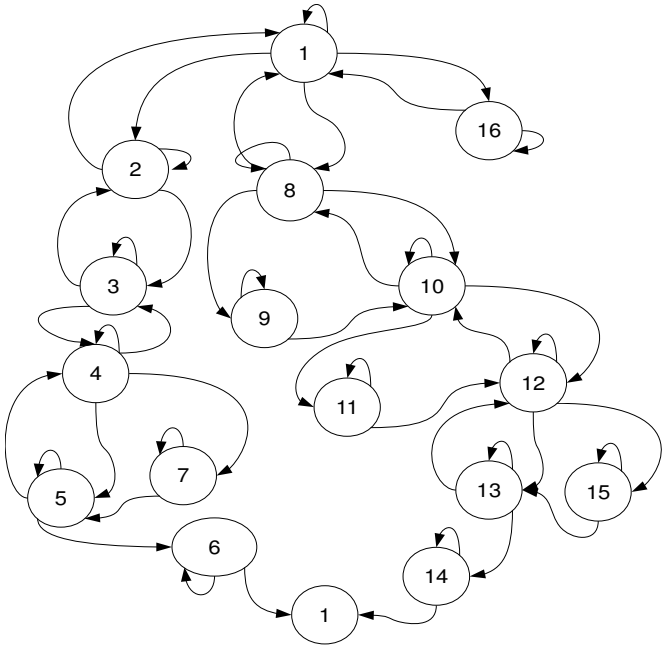


Figure 3: Markov Chain Model of System Test Execution

In the Markov chain model of system testing as shown above, the terminal test cases are linked to the initial state/test case (for example the path from test case 6 to test case 1). This is explained by the fact that , once test execution reaches the terminal test case, it is evident that all test cases in that logic path have already passed and the next option would be to move to the next branch. Another important point to note is that if there is more than one branch for a test case, and if one of the branches has a terminal test case while the other branch has further sub branches, then once the terminal test case passes, test execution jumps to the corresponding test case that exists at the same level (for example the path from test case 9 to test case 10). The explanation for this as the same as given above. The system test chain also has test cases for which there exists a path to the previous test case (for example the path from test case 2 to test case 1). This models the phenomenon, that once a test case fails execution, the tester might decide to jump to the next logic path instead of waiting at the same test case. Self loops model re-testing of a failed test case. As evident above, the test execution process is a random walk on a finite graph and as discussed by *Sarkar* [9], a stationary distribution ($\prod_*$) can exist iff the chain is

a) **Aperiodic** i.e $P(X_t = i \mid X_0 = i) > 0$ which can be obtained by ensuring that the GCD of all cycle lengths in the Markov chain is 1.

b) **Irreducible** i.e $P(X_t = j \mid X_0 = i) = p_{ij}^{(n)} > 0$ which can be obtained by ensuring that there exists a path from every state to every other state.

c) **Recurrent** i.e $\tau_i = \inf\{t \geq 1; X_t = i \mid X_0 = i\}; P(\tau_i < \infty) = 1$ where τ_i is the hitting time

Therefore it is imperative to ensure that the Markov chain model of the test suite satisfies (a), (b) and (c) in order to attain a stationary distribution.

4.1 Stationary Distribution of System Testing

As shown by *Bhulai and Koole* [11], a Markov chain has an invariant measure (stationary distribution), $\lim_{t\to\infty}\Pi_t = \Pi_*$ for some arbitrary distribution $\Pi_0; \sum_{j=1}^{N}\Pi_j = 1; \Pi_j \geq 0$ iff the Markov chain is ergodic (aperiodic, recurrent and irreducible). Then the distribution Π_* is the unique solution of

$$\Pi_* = \Pi_* P \tag{3}$$

Which is obtained by iterating through $\prod_{t+1} = \prod_t P$ until convergence is attained.

4.2 Expected Time required to complete System Testing

We use the algorithms given by *Sigman*[12] for simulating Markov Chains and modify it to run until all states of the Markov Chain have been walked at least once.

4.2.1 Algorithm for Simulating System Testing in Discrete Time

1. Set $i = 1,\ \tau = 0$, *wlkVec=[i]*

2. If $\exists i \in wlkVec \forall i \in \aleph$ *then stop; otherwise goto step 3*

3. Generate $u \sim U[0,1]$, *set* $i = j$ *where* $u \leq \sum_{k=1}^{j} p_{i,k}$ *; set* $\tau = \tau + 1$ *;add i to wlkVec*

4. Go to step 2

The Weak Law of Large Numbers implies

$$\lim_{n\to\infty} \mathrm{P}\left(\left|\overline{\tau}_n - E[\mathrm{T}]\right| > \varepsilon\right) = 0 \forall \varepsilon > 0 \tag{4}$$

Therefore, if we run the simulation for a sufficiently long time such that $\left|\overline{\tau}_n - E[\mathrm{T}]\right| \leq \varepsilon, \forall \varepsilon > 0$, then the average of the time ($\overline{\tau}$) obtained from each run of the simulator will converge in probability to the expected time (T) required to walk all the states. This is an important insight for the test manager because it helps her see the relationship between test design and expected completion time of test execution.

5. Modeling System Testing as a Semi-Markov Process

Elmaghraby [1] defines a semi-Markov Process as a stochastic process that makes its transition from state *'i'* to any other state *'j'* (including *i=j*), according to the transitional probability matrix of a Markov Process but whose time between transitions is a random variable that may depend on both *'i'* and *'j'*. Software Test Execution has a structural similarity to the above because once a tester reaches test case *'i'*, she chooses the next test, *'j'* to be executed. This decision is made according to the probabilities p_{ij} of the transition probability matrix. Now, test case *'i'* is executed for a random duration $t \in T_{ij}; T_{ij} \in [0,\infty)$ and this duration depends on both *'i'* and *'j'* as the decision to move on to test case *'j'* influences the inputs to test case *'i'* and thus the duration of time spent on test case *'i'*. Moreover, test case *'i'* can be an absorbing state/trapping state in case the test case fails but the recurrence property of the imbedded Markov chain defined above and $p_{ij} < 1, \forall i = j$ we can safely conclude that the process will jump out of the trapping state in finite time.

The duration of executing a test case is denoted by the random variable, $T_{ij} \in [0,\infty)$ where $P(T_{ij} \in [0,\infty)) = F_{T_{ij}}(t) = \int_0^t f_{T_{ij}}(\tau)d\tau = 1$ and $0 \le P(T_{ij} = t) = f_{T_{ij}}(t) \le 1, \forall t \in [0,\infty)$. The expected duration of executing a test case is denoted by $E[T_{ij}] = \int_0^\infty t f_{T_{ij}}(t)\,dt, \forall\, t \in [0,\infty)$. Since $P(T_{ij} < \infty) = 1$ and assuming that *absolute integrability* (Lebesgue Integrability) is satisfied i.e $\int_0^\infty |t|\, f_{T_{ij}}(t)dt < \infty$, implies $E[T_{ij}] < \infty$. The Laplace-Stieltjes transform $\tilde{T}_{ij}(s)$ for the non negative random variable T_{ij} is defined as

$$\tilde{T}_{ij}(s) = E(e^{-sT_{ij}}) = \int_{t=0}^{\infty} e^{-st} dF_{T_{ij}}(t) = \int_{t=0}^{\infty} e^{-st} f_{T_{ij}}(t)dt \;,\; s \ge 0; \left|\tilde{T}_{ij}(s)\right| \le 1, \forall s \ge 0; \tilde{T}_{ij}(0) = 1,\; \tilde{T}_{ij}'(0) = -\mathrm{E}(T_{ij}),\; \tilde{T}_{ij}^{(r)}(0) = (-1)^r \mathrm{E}(T_{ij}^{\,r})$$

We define $p_{ij}F_{T_{ij}}(t)$ as the transition distribution from state *'i'* to state *'j'*.

A test manager, watching the system in state *'i'* can only infer the unconditional probability density function of the waiting time at state *'i'* (as it's the tester who is executing the test case and has made the decision of moving to test case *'j'* from *'i'*).Therefore, let the unconditional waiting time in state *'i'* be denoted by the random variable $W_i(t) \in [0,\infty)\ \forall\ t \in [0,\infty)$ where the density function of W_i is given by $f_{W_i}(t) = \sum_{j=1}^{|N|} p_{ij} f_{T_{ij}}(t)$ and $P(W_i \in [0,\infty)) = F_{W_i}(t) = \int_0^t f_{W_i}(\tau)d\tau = 1; 0 \le P(W_i = t) = f_{W_i}(t) \le 1, \forall t \in [0,\infty)$. The expected unconditional waiting time in test case *'i'* is given by:

$$E[W_i] = \int_0^\infty t f_{W_i}(t)dt = \int_0^\infty t\Big(\sum_{j=1}^{N} p_{ij} f_{T_{ij}}(t)\Big)dt = \sum_{j=1}^{N} p_{ij} \int_0^\infty t f_{T_{ij}}(t)dt = \sum_{j=1}^{N} p_{ij} E[T_{ij}] \qquad (5)$$

Similar to the expected value of the conditional waiting/transition time we assume that the expected value of the unconditional waiting/transition time satisfies absolute integrability and therefore exists and is finite. The interchange of sum and integral in (5) is due to *Tonelli's theorem* since $\sum_{j=1}^{N} p_{ij} f_{T_{ij}}(t)dt > 0$.

The Laplace-Stieltjes transform of the unconditional waiting time is given by:

$$\tilde{W}_i(s) = E(e^{-sW_i}) = \int_{t=0}^{\infty} e^{-st} dF_{W_i}(t) = \int_{t=0}^{\infty} e^{-st} f_{W_i}(t)dt \;,\; s \geq 0; \left|\tilde{W}_i(s)\right| \leq 1, \forall s \geq 0; \tilde{W}_i(0) = 1,\; \tilde{W}_i'(0) = -\mathrm{E}(W_i),\; \tilde{W}_i^{(r)}(0) = (-1)^r \mathrm{E}(W_i^r)$$

5.1 Interval Transition Probability of System Testing

Let the interval transition probability be denoted by $\phi_{ij}(t) = \mathrm{P}(\mathrm{X}_t = j \mid X_0 = i)$ which is the (conditional) probability that test case *'j'* is being executed at time *'t'*, given that that tester was at test case *'i'* at *t=0*.Therefore $\phi_{ij}(t)$ is the (conditional) probability of being in test case *'j'* at time *'t'*, through multiple transitions, starting from test case *'i'*. Whereas, p_{ij} is the (conditional) probability of a single transition from test case *'i'* to *'j'*, *'j'* being the immediate successor of *'i'* in this case .ϕ is often called the interval transition probability.

Let use introduce the Kronecker delta, $\delta_{ij} = \begin{cases} 1 \text{ if i=j} \\ 0 \quad \text{o.w.} \end{cases}$ .Where $\delta_{ij} = 1$ denotes that the test case has failed and the test execution is stuck at this state. The probability of the unconditional waiting time when a test case fails is given by $\overline{f}_{W_i}(t) = 1 - \sum_{j=1}^{N} p_{ij} f_{T_{ij}}(t) = p_{ii} \overline{f}_{T_{ij}}(\mathrm{t}) = 1 - f_{W_i}(t)$ where $p_{ii} = 1 - \sum_{j=1}^{N} p_{ij}, \forall i \neq j$ is the probability that the test case fails after being executed for a duration *'t'*. Therefore, the distribution of the (unconditional) waiting time in the event of failure of a test case can be represented as $\mathrm{F}_{W_i}(\mathrm{t}) = 1 - F_{W_i}(t)$. Intuitively, if $F_{W_i}(t)$ is the probability of completing execution of test case 'i' in time 't' and then moving on to a successive test case, then $\overline{\mathrm{F}}_{W_i}(\mathrm{t}) = 1 - F_{W_i}(t)$ is the probability of remaining in state *'i'* even after time *'t'*, which can only occur in case of a failure.

Therefore the probability that the test execution started at test case *'i'* and has remained there all the time is given by $\delta_{ij} \overline{F}_{W_i}(t)$ .Let *'k'* be an intermediate test case between *'i'* and *'j'* and $0 \leq \tau < t$ be the time required to execute test case *'i'*. Thus, if *'i'* and *'j'* are connected by test case *'k'* and if it takes a duration *'t'* to reach test case *'j'*, then τ is the amount of time required to reach test case *'k'* and the remaining

$t-\tau$ is the amount of time required to transition from test case *'k'* to test case *'j'*. The following figure shows the execution time for different transitions of the testing Markov chain.

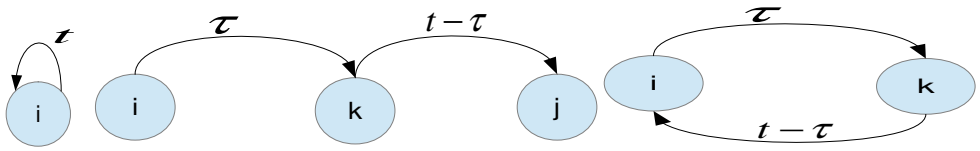


Figure 4: Transition Time of the System Test Markov Chain

As shown by *Osaki and Mine* [2] and also by *Elmaghraby* [1], $\phi_{ij}(t)$ can be formally defined with the following recursive equation

$$\phi_{ij}(t) = \delta_{ij}\overline{F}_{W_i}(t) + \sum_{k=1}^{N} p_{ik}\int_0^t f_{T_{ik}}(\tau)\phi_{kj}(t-\tau)d\tau \tag{6}$$

This implies that when *i=j*, there are two events, one is that the system has stayed in the same state till time *'t'* and another is that the system visits state *'k'* at time τ and returns to state *'i'* at time *'t'*. *These two events are mutually exclusive* .When $i \neq j$, the system visits state *'k'* in time τ and transitions from state *'k'* to state *'j'* in time $t-\tau$.

The convolution in (1) can be written as a Laplace transform as follows,

$$\int_0^t f_{T_{ik}}(\tau)\phi_{kj}(t-\tau)d\tau = \tilde{T}_{ik}(s)\tilde{\phi}_{kj}(s)\ ;\tilde{\phi}_{kj}(s) = E(e^{-s\phi_{kj}}) = \int_{t=0}^{\infty} e^{-st}dF_{\phi_{kj}}(t) = \int_{t=0}^{\infty} e^{-st}f_{\phi_{kj}}(t)dt\ ,\ s\geq 0$$

$$\left|\tilde{\phi}_{kj}(s)\right| \leq 1, \forall s \geq 0\ ;\tilde{\phi}_{kj}(0) = 1,\ \tilde{\phi}_{kj}^{'}(0) = -\mathrm{E}(\phi_{kj}),\ \tilde{\phi}_{kj}^{(r)}(0) = (-1)^r \mathrm{E}(\phi_{kj}^{\ r})\ .\text{Also, } L(1) = \int_0^{\infty} e^{-st}dt = \frac{1}{s}$$

Therefore transforming (6), we obtain

$$\tilde{\phi}_{ij}(\mathrm{s}) = \frac{1}{s}\delta_{ij}[1-\tilde{F}_{W_i}(s)] + \sum_{k=1}^{N} p_{ik}\tilde{T}_{ik}(s)\tilde{\phi}_{kj}(s) \tag{7}$$

Let, $\tilde{\phi}(s) = [\tilde{\phi}_{ij}(s)], \tilde{q}(s) = [p_{ij}\tilde{T}_{ij}(s)], \tilde{F}_W(s) = [\delta_{ij}\tilde{F}_{W_i}(s)], \delta_{ij} = I$ be $N \times N$ matrices. Then (7) can be rewritten in matrix form as $\tilde{\phi}(s) = \frac{1}{s}[I - \tilde{F}_W(s)] + \tilde{q}(s)\tilde{\phi}(s) \Rightarrow [I-\tilde{q}(s)]\tilde{\phi}(\mathrm{s}) = \frac{1}{s}[I-\tilde{F}_W(s)]$

$$\Rightarrow\ \tilde{\phi}(s) = [I - \tilde{q}(s)]^{-1}\frac{1}{s}[I - \tilde{F}_W(s)] \tag{8}$$

We assume here that the transition distribution $p_{ij}F_{T_{ij}}(t)$ is invertible which is a necessary condition for the existence of $[I-\tilde{q}(s)]^{-1}$.

5.2 Stationary Distribution (invariant measure) of System Testing

Osaki and Mine [2] define a semi-Markov process to be ergodic if the imbedded Markov chain is ergodic. We now discuss the limiting behavior of an ergodic semi-Markov process. To derive the stationary performance, the behavior of the system as $t\to\infty$, we use the final value theorem $\lim_{t\to\infty} f(t)=\lim_{s\to 0} s\tilde{F}(s)$ of Laplace-Stieltjes transform to obtain $\lim_{t\to\infty}\phi(t)=\lim_{s\to 0} s\tilde{\phi}(s)$.

Thus (8) can be written in the form $\lim_{s\to 0} s\tilde{\phi}(s)=\lim_{s\to 0} s[I-\tilde{q}(s)]^{-1}\frac{1}{s}[I-\tilde{F}_W(s)]$. Using L'Hopital's rule we get $\lim_{s\to 0}\frac{1}{s}[I-\tilde{F}_W(s)]=\left.\frac{\frac{d}{ds}[I-\tilde{F}_W(s)]}{\frac{d}{ds}[s]}\right|_{s=0}=-\tilde{F}_W^{'}(0)=[\delta_{ij}E(W_i)]$. *Howard* (1964) proved that

$$lim_{s\to 0}s[I-\tilde{q}(s)]^{-1}=\frac{1}{\sum_{j=1}^{N}\Pi_j E(W_j)}\begin{bmatrix}\Pi_1 & \dots & \Pi_N\\ : & \dots & :\\ \Pi_1 & \dots & \Pi_N\end{bmatrix}$$

where the vector $\Pi=(\Pi_1,\Pi_2,\dots,\Pi_N)$ is the vector of the steady state probabilities of the 'imbedded Markov Chain'. Therefore, the limiting transition probability, that after a long period of time, test execution will be at test case *'j'* is given by

$$\phi_j=\frac{\Pi_j E[W_j]}{\sum_{j=1}^{N}\Pi_j E[W_j]};\ j=1,2,\dots,N \qquad (9)$$

5.3 Simulating the Semi-Markov Process of System Testing

If the execution time of a test case T_{ij} does not have an exponential distribution, then the resulting semi-Markov process does not possess the Markov property. This means that the execution time T_{ij} of test case *'i'* has a general distribution $F_{T_{ij}}$. We assume that $F_{T_{ij}}$ can be computed analytically and has a tractable expression for the probabilities and is invertible. When $F_{T_{ij}}$ cannot be computed analytically and has no tractable expression, we need to resort to Monte Carlo Markov Chain methods (MCMC) to simulate the Semi-Markov Process and is out of the scope of this paper. We use the algorithm given by *Sigman*[12] for simulating semi-Markov Chains when T_{ij} follows a general analytically tractable distribution $F_{T_{ij}}$ and modify it to run until all states of the semi-Markov Chain have been walked at least once.

5.3.1 Algorithm for Simulating System Testing in Continuous Time

1. Set $i = 1$, *Generate* $T_{ij} \sim F_{T_{ij}}$,*set* $\tau = T_{ij}$, *wlkVec=[i]*

2. If $\exists i \in wlkVec \forall i \in \aleph$ *then stop; otherwise goto step 3*

3. Generate $u \sim U[0,1]$, *set j where* $u \leq \sum_{k=1}^{j} p_{i,k}$

4. if j>=i and $k \notin wlkVec \forall k : j \leq k \leq N; p_{jk} > 0$ *goto step 5 else goto step 6*

5. Generate $T_{ij} \sim F_{T_{ij}}$ *and set* $\tau = \tau + T_{ij}$

6. set $i = j$ *; add i to wlkVec*

7. Go to step 2

We generate $T_{ij} \sim Exp(\lambda_i)$ *when the execution time is exponentially distributed and as a result the Markov Process is a Continuous Time Markov Chain.*

Again, by the Weak Law of Large Numbers given in equation(4) we conclude that if we run the simulation for a sufficiently long time such that $\left|\overline{\tau}_n - E[\mathrm{T}]\right| \leq \varepsilon, \forall \varepsilon > 0$, then the average of the time ($\overline{\tau}$) obtained from each run of the simulator will converge in probability to the expected time (T) required to walk all the states.

6. Optimal Control of System Testing-The Test Management Model

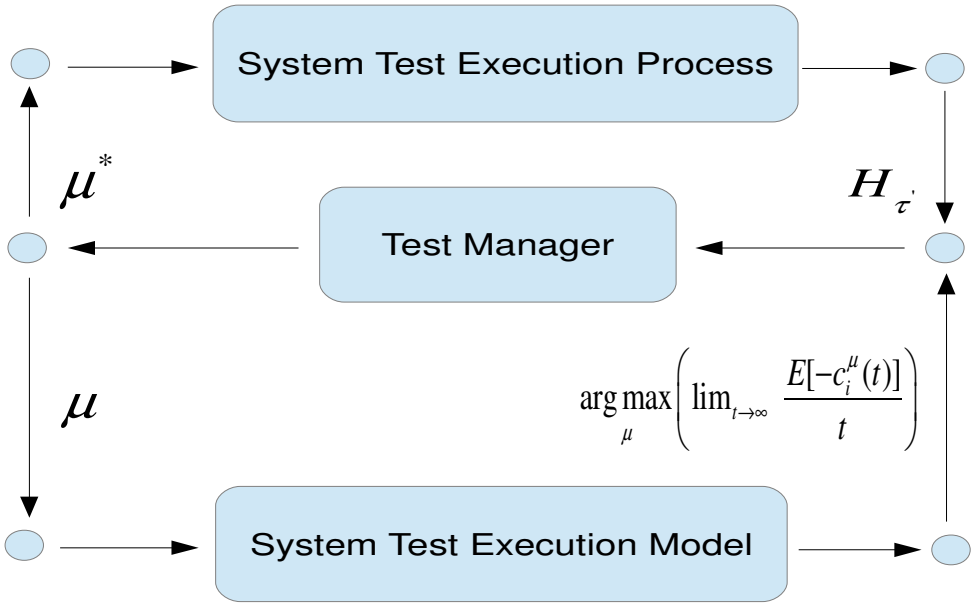


Figure 5: Optimal Control Model for System Test Execution Process

Gimbert[13] defines a controllable Markov Chain $(\aleph, U, \mu, p)$ to satisfy the following properties

- The state space $|\aleph| < \infty$ and the control space $|U| < \infty$
- For each state $X_t \in \aleph$ a control $U_t \in U$ can be selected according to some policy $\mu : \aleph \to U$
- The transition probability, $\mathrm{P} : \aleph \times U \to \aleph$ where the triplet (i, u, j) represents a transition with transition probability $\mathrm{P}(j \mid i, u) > 0, i, j \in \aleph$ and $u \in U$

The cost space C would depend on the action space, $C(t) : \aleph \times U \to \mathbb{R}^+, \forall C(t) \in C$ .A policy μ is a decision making function of control strategy of the agent, representing a mapping from the state space to the control space.

Let us first define and elaborate on the decision epoch. A test manager reviews the test execution progress (based on the *history* of the process) at scheduled checkpoints, during the testing cycle and it is at these discrete points in time that the manager takes a decision thus applying a feedback control in order to the stabilize (maximize the expected utility) the test execution process.

A *history* of the process until decision epoch $\tau' = 1, 2, \ldots$ denoted by a random vector $H_{\tau'}$ is a sequence of transitions such that $H_{\tau'} = \begin{cases} (X_{\tau'} = i, t, X_{\tau'+1} = j) \mid \forall i \; \exists j \mid p_{ij} \neq 0, t \sim f_{T_{ij}}(t) \text{ if } \tau' = 1 \\ (X_{\tau'} = i, U_{\tau'} = u, t, X_{\tau'+1} = j) \forall i \; \exists j \mid p_{ij} \neq 0, t \sim f_{T_{ij}}(t), \mathrm{u} \in U \text{ if } \tau' > 1 \end{cases}$

The semi-Markov decision process therefore should satisfy the following Markov Property

$$P(T_{ij}=t, X_{\tau'+1}=j \mid H_{\tau'}, U_{\tau'}=u) = P(T_{ij}=t, X_{\tau'+1}=j \mid X_{\tau'}=i, U_{\tau'}=u) \quad (10)$$

The test management problem is a finite horizon stochastic optimal control problem and can be solved by Stochastic Dynamic Programming. Since we have considered a simplified model of software test execution in this paper with only one tester, the state space is not multi dimensional and therefore does not suffer from the *curse of dimensionality*.

We denote $E[c_i^{\mu}(t)]$ as the expected cost up to time *'t'* beginning from test case *'i'* as the initial state and applying policy $\mu : \aleph \to U$ .Thus our optimal control problem is to find a policy μ^* that minimizes the average expected long run cost when starting in test case *'i'* over execution of N test cases which can be formulated as

$$\arg\max_{\mu}\left(\lim_{t\to\infty}\frac{E[-c_i^{\mu}(t)]}{t}\right) \quad (11)$$

6.1 The Cost Function of System Testing

The costing of most testing projects is either 'Time and Material' or 'Fixed Price'. In this paper we will only consider 'Time and Material' costing wherein cost of test execution is calculated as follows:

$$\text{Cost of Executing a Test Case=Duration of Execution} \times \text{Billing Rate} \quad (12)$$

The billing rate is a contractual agreement and does not fluctuate during the course of test execution (and is therefore constant).Thus, we can safely assume that the cost of test execution is a linear function of time and the total cost of test execution is a linear combination of the cost of executing each test case in the test suite.

Let $c_i \in \mathbb{R}^+$ be the cost of executing test case *'i'* $(i = 1,..,\mathrm{N})$ .Therefore, we define cost as a monotone non decreasing function $c_i : \mathbb{R}^+ \to \mathbb{R}^+$ of duration (T_{ij}) wherein $F_{c_i(t)}(y) = P(c_i(F_{T_{ij}}(\mathrm{t})) \le y)$,the function of a random variable is also a random variable. We define the random execution time of test case *'i'*, T_{ij} on the probability space (Ω, F, P) .

Then we can write $(c_i, t) \mapsto f_{c_i}(t) : ([0,\infty)\times\Omega, B([0,\infty)) \otimes F) \to (\mathbb{R}^d, B(\mathbb{R}^d))$ to define the dependence of cost c_i on $f_{T_{ij}}(t)$ .It is important to note that we denote $c_i(1)$ as c_i which is a constant. Moreover, since the test execution Markov chain is recurrent, the duration of time spent in execution of test case *'i'* is finite. This implies by definition that the cost $c_i(t)$ of executing test case *'i'* is finite as well and $c_i(t) \uparrow c_i$ as $t \to \infty$. Thus, by the *Monotone Convergence Theorem*, we can conclude that the $\lim_{t\to\infty} E[c_i(t)] \uparrow E[\lim_{t\to\infty} c_i(t)]$

Further, assuming absolute integrability $\int_0^{\infty} | c_i(t) | dF_{c_i(t)} < \infty \Rightarrow E[c_i(t)] < \infty$

6.2 Semi-Markov Process Model of System Testing with Costs

Now we shall consider the total expected cost up to time't', $E[c_i(t)] < \infty$ when test execution starts from test case *'i'* at time zero. $E[c_i(t)]$ satisfies the following equation of renewal type, $\forall t \geq 0$

$$E[c_i(t)] = [1 - F_{W_i}(t)]c_i t + \sum_{j=1}^{N} p_{ij} \int_0^{\infty} \{c_i \tau + E[c_j(t-\tau)]\} f_{T_{ij}}(\tau) d\tau \quad (13)$$

The tail distribution of the unconditional waiting time $1 - F_{W_i}(t)$ approaches zero as *'t'* gets large. This is explained by the fact that the probability that a test case will remain in failed state will tend to zero as the number of retests and hence the duration of testing *'t'* gets large. Moreover, $E[W_i] < \infty \Rightarrow [1 - F_{W_i}(t)] \to 0$. Therefore, for large *'t'*, (4) can be re-written as

$$E[c_i(t)] = \sum_{j=1}^{N} p_{ij} \int_0^{\infty} c_i \tau f_{T_{ij}}(\tau) d\tau + \sum_{j=1}^{N} p_{ij} \int_0^{\infty} E[c_j(t-\tau)] f_{T_{ij}}(\tau) d\tau \Rightarrow c_i E[W_i] + \sum_{j=1}^{N} p_{ij} \int_0^{\infty} E[c_j(t-\tau)] f_{T_{ij}}(\tau) d\tau \quad (14)$$

Let us introduce, $E[c(t)]_{N\times 1} = \begin{bmatrix} E[c_1(t)] & E[c_2(t)] & \dots & E[c_N(t)] \end{bmatrix}; c_{N\times 1} = \begin{bmatrix} c_1 & c_2 & \dots & c_N \end{bmatrix}$

$E[W]_{N\times N} = [\delta_{ij} E(W_i)]; q(t)_{N\times N} = [p_{ij} f_{T_{ij}}(t)]$. We can then re-write (5) as

$$E[c(t)] = E[W]c + q(t) * E[c(t)] \quad (15)$$

Where '*' denotes convolution. Taking the Laplace-Stieltjes transform, we get

$$E[\tilde{c}(s)] = \frac{1}{s} E[W]c + \tilde{q}(s) E[\tilde{c}(s)] \Rightarrow E[\tilde{c}(s)] = \frac{1}{s} [I - q(s)]^{-1} E[W]c \quad (16)$$

We are interested in the limiting behavior of (6) i.e $\lim_{t\to\infty} E[c(t)] = \lim_{s\to 0} sE[\tilde{c}(t)]$

6.3 Expected Long-Run Stationary Cost of System Testing

As shown by *Osaki and Mine* [2], the expected long-run stationary cost which is the average cost of executing a test case ($\forall i \in \aleph$), when the duration of test execution $t \to \infty$ can be written as

$$\lim_{t\to\infty} \frac{1}{t} E[\int_0^t c_i(\tau) d\tau] = g = \frac{\sum_{j=1}^{N} \Pi_j E[W_j] c_j}{\sum_{j=1}^{N} \Pi_j E[W_j]} \quad (17)$$

6.4 The Control Space and Control Policy of System Test Management

We model the control space U as $U = \{0,1\}$ where $u = 0$ would mean no additional investment is required while $u = 1$ implies that additional investment is required. Thus we define the control policy as follows: At execution epoch of every test case, $i \in \aleph$, the test manager can decide to make an additional

investment (and thus reduce the time to complete test execution, thereby increasing cost) or choose not to make any additional investment if she finds that there is no risk of schedule overrun. Formally,

$$\mu_i = \begin{cases} 1 \; if \;\; additional \;\; investment \;\; is \;\; made \\ 0 \;\; o.w. \end{cases} \tag{18}$$

When the test manager chooses a control policy $\mu_i = u \forall i \in \aleph, u \in U$, then the system obeys the probability law $p_{ij}^u F_{T_{ij}}^u(t)$ where p_{ij}^u is the transition probability from test case *'i'* to *'j'* when control *'u'* is applied and $F_{T_{ij}}^u$ is the probability distribution of waiting in state *'i'* when the next state *'j'* is chosen and control *'u'* is applied. It is important to note that $p_{ij}^0 \le p_{ij}^1, \forall i < j \text{ or } j = 1; p_{ij}^1 \le p_{ij}^0 \forall i \ge j \;\&\; j \ne 1$ and $F_{T_{ij}}^0(t) = P^0(T_{ij} \le t) < P^1(T_{ij} \le t) = F_{T_{ij}}^1(t) \forall i, j \in \aleph$ This further implies that $E[T_{ij}^1] < E[T_{ij}^0] \forall i, j \in \aleph$ .An intuitive explanation for this is that if the test manager wants to make additional investment in execution of test case *'i'*, then the probability that the test execution will jump out of state *'i'* is equal to or higher than that with no investment. Similarly, an additional investment in execution of test case *'i'*, should reduce the probability of waiting in state *'i'* upto time *'t'* and hence the expected waiting time as well.

Moreover, c_i^u will denote the cost of executing test case *'i'* per unit time under the influence of control *'u'* and $E[W_i^u] = \sum_{j=1}^{N} p_{ij}^u E[T_{ij}^u]$ is the expected unconditional waiting time in test case *'i'* when control *'u'* is applied. As a consequence, $c_i^0 < c_i^1 \forall i \in \aleph$ and $E[W_i^0] > E[W_i^1] \forall i \in \aleph$ .Therefore we obtain the following expression for the expected long-run stationary cost under control policy μ

$$\lim_{t \to \infty} \frac{1}{t} E[\int_0^t c_i^\mu(\tau) d\tau] = g^\mu = \frac{\sum_{j=1}^{N} \prod_j^u c_j^u E[W_j^u]}{\sum_{j=1}^{N} \prod_j^u E[W_j^u]} \tag{19}$$

6.5 Semi-Markov Decision Process Model for System Test Management

We use the same arguments as given by *Bhulai and Koole* [11] to derive the average expected long-run costs $E[c_i^\mu]$ when starting in test case *'i'*. As defined earlier, $E[c_i^\mu(t)]$ denotes the total expected cost till time *'t'* when beginning from test case *'i'* at time 0, and control $u \in U$ is applied. From (5), we have $E[c_i^\mu(t)] = c_i^\mu E[W_i^\mu] + \sum_{j=1}^{N} p_{ij}^\mu \int_0^\infty E[c_j^\mu(t-\tau)] f_{T_{ij}}^\mu(\tau) d\tau$ and $E[c_i^\mu] = \lim_{t \to \infty} \{E[c_i^\mu(t)] - g^\mu t\}$ is the total expected difference in costs between starting in test case *'i'* and starting in stationarity. Therefore we can write, $E[c_i^\mu(t)] + g^\mu E[W_i^\mu] + o(1) = c_i^\mu E[W_i^\mu] + \sum_{j=1}^{N} p_{ij}^\mu E[c_j^\mu(t)]$

Subtracting $g^\mu t$ from both sides and taking the limit $T \to \infty$ leads to the Poisson equation

$$E[c_i^{\mu}] + g^{\mu} E[W_i^{\mu}] = c_i^{\mu} E[W_i^{\mu}] + \sum_{j=1}^{N} p_{ij}^{\mu} E[c_j^{\mu}] \tag{20}$$

An important point to note here is $E[c_1^{\mu}] = \lim_{t\to\infty}\{E[c_1^{\mu}(t)] - g^{\mu}t\} = 0$ which is implies from (19) because $\lim_{t\to\infty} E[\int_0^t c_i^{\mu}(\tau)d\tau] = g^{\mu}t$ which is the expected of cost executing a test case over all test cases in the system and with time $t \to \infty$. It is also important to note the condition. Another condition that is necessary to impose on the Poisson equation in (20) is that $E[c_j^{\mu}] = 0 \forall j \le i$. This follows from the argument given for the formulation equation (14) that as $t \to \infty$, the probability that the test case will keep failing tends to 0 and therefore the probability of waiting in the same test case, $1 - F_{W_i}(t)$ converges to 0 faster than $c_i(t)$ diverges.

6.5.1 Policy Iteration to find the Optimal Policy

Next we present the algorithm given be *Bhulai and Koole* [11] to find the optimal policy using *Bellman Equation* in (21)

1. Choose $\mu_i = 0$ *or* 1

2. Compute g^{μ} *and* $E[-c_i^{\mu}]$ $\forall i \in \aleph$.

3. Find the optimal policy μ_i^* *.If* $\mu_i = \mu_i^*$ *, then stop else goto step 4*

$$\mu_i^* = \underset{u\in U}{\operatorname{argmax}}\{(-c_i^u + g^{\mu})E[W_i^u] + \sum_{j=1}^{N} p_{ij}^u E[-c_j^{\mu}]\} \tag{21}$$

4. Set $\mu_i = \mu_i^*$ *and goto step 2.*

7. The Optimal Additional Investment in System Testing

In this section we consider the solution proposed by *Elmaghraby* [1] for computing the optimum additional investment and present it in the framework of system testing. Let us assume that an additional amount $r_{ij} \in [0,\infty)$ is invested in order to reduce the expected time taken to execute test case '*i*' (and thus realize test path '*ij*'). We have already defined the random execution time of test case '*i*', T_{ij} on the probability space (Ω, F, P).Then we can write $(r,t) \mapsto f_{r_{ij}}(t) : ([0,\infty)\times\Omega, B([0,\infty)) \otimes F) \to (\mathbb{R}^d, B(\mathbb{R}^d))$ to define the dependence of $f_{T_{ij}}(t)$ on additional investment r_{ij}, wherein $f^0_{T_{ij}}(t)$ is the probability of completing execution of test path '*ij*' at time t, without any additional investment. This would mean that the expected value of the execution time i.e. $E[T_{ij}]$ reduces by a certain amount whenever and additional investment is made. Let the new expected value of the execution time be denoted as $\hat{E}[T_{ij}] < E[T_{ij}]$. Please note that if the above inequality does not hold true for any $r_{ij} > 0$, it would be futile to invest any additional amount in the first place. Let us denote the reduction in expected execution time as a function of additional investment, $\zeta(r_{ij})$.Therefore, $\hat{E}[T_{ij}] = E[T_{ij}] - \zeta(r_{ij})$ and $\hat{E}[c_{ij}(t)] = E[c_{ij}(t)] + r_{ij}$. Moreover, we make a simplifying assumption that the decrease in average duration is a linear function of investment. Thus, we can rewrite the average duration and cost introduced earlier as follows

$$\hat{E}[T_{ij}] = E[T_{ij}] + qr \ , -\frac{E[T_{ij}]}{r} < q < 0; \hat{E}[c_{ij}(t)] = E[m\hat{T}_{ij}] + r = mE[\hat{T}_{ij}] + r, m \geq 0 \qquad (22)$$

Therefore, we can also write $\hat{E}[c_{ij}(t)] = mE[T_{ij}] + (mq+1)r, m \geq 0 \Rightarrow E[c_{ij}(t)] + (mq+1)r, m \geq 0$

where q is the marginal decrease in the test execution time per unit increase in investment and '*m*' is the linear slope of the cost function. It's important to note that the amount of capital r available to a test manager is finite and therefore q is bounded. Also it's important to note that the probability of realizing a test case depends on the path that is chosen (there can exist multiple logic paths through a test case).Therefore, the expected cost (without any additional investment) given by the '*law of unconscious statistician*' is $E[c_{ij}(t)] = \int_a^b c_{ij}(t) dF_{T_{ij}}{}^0(t)$ where '*a*' and '*b*' are the limits on the duration of execution of test path '*ij*'. Further, if an investment r_{ij} is made in test path '*ij*', the new expected cost will be given by $\hat{E}[c_{ij}(t)] = r_{ij} + \int_{\hat{a}}^{\hat{b}} c_{ij}(\hat{t}) dF_{T_{ij}}{}^{r_{ij}}(\hat{t})$ where $\hat{a}$ and $\hat{b}$ are the new limits on the duration when investment r_{ij} has been made in test path '*ij*'. Now, either of the following can be true $\forall r_{ij} > 0$, $E[\hat{c}_{ij}(t)] \leq E[c_{ij}(t)]$ or $E[\hat{c}_{ij}(t)] \geq E[c_{ij}(t)]$. We will only consider $E[\hat{c}_{ij}(t)] \leq E[c_{ij}(t)]$ as that is the objective of a test manager which means the additional investment r_{ij} is adequately compensated by the decrease in duration of test

case *'i'* and eventually reduces the cost. Therefore there exists a minimum additional investment for test path *'ij'* in order to minimize the expected cost. Suppose, that the status of testing (as revealed in the test execution report) at some time t>0 after its initiation, reveals that the execution of one or more test cases has taken so long to complete that the probability of completing one or more test/logic paths is dangerously low. Then the optimization problem translates to

*'what is the optimal allocation of a fixed amount of capital **K** among the remaining test cases such that the probability of realizing the terminal test cases on or before a specified time is maximized'*

Let us denote the set of terminal test cases as $L=\{l\in N \mid p_{ll}=1\}$ .Therefore, $T_l \forall l\in L$; $C_l \forall l\in L$ and $P_l \forall l\in L$ is the execution time from the first test case to the terminal test case, the cost of reaching the terminal test case starting from the first test case and the probability of reaching the terminal test case respectively. Let $\pi_l^{\,x}$ denote the path *'x'* from the starting test case to the terminal test case *'l'* and $P(\pi_l^{\,x})$ or $P_l^x, l\in L$ denote the probability of realizing test case $l\in L$ along path *'x'*. Similarly, let T^x be the duration of realizing path *'x'*, therefore $T_l^x=\sum_{(ij)\in\pi^x} T_{ij}$ is the sum of independent random variables T_{ij} .Hence, $P_l T_l=\sum_k P_l^k T_l^k$ implies $E[T_l]=E\left[\frac{\sum_k P_l^k T^k}{P_l}\right]$ .Since, $\sum_k P_l^k T^k$ where $T^k=\sum_{ij\in\pi^k} T_{ij}$, using *Linearity Property* , we can write the following $E[T_l]=\frac{\sum_k P_l^k E[T^k]}{P_l}=\frac{\sum_k P_l^k E[\sum_{ij\in\pi^k} T_{ij}]}{P_l}=\frac{\sum_k P_l^k \sum_{ij\in\pi^k} E[T_{ij}]}{P_l}$ .Replacing, $E[T_{ij}]=E[\hat{T}_{ij}]-q_{ij}r_{ij}$, we get $E[T_l]=\frac{\sum_k P_l^k \sum_{ij\in\pi^k}(E[\hat{T}_{ij}]-q_{ij}r_{ij})}{P_l}=E[\hat{T}_l]-\frac{\sum_k P_l^k \sum_{ij\in\pi^k} q_{ij}r_{ij}}{P_l}$. Therefore,

$$E[\hat{T}_l]=E[T_l]+\frac{\sum_k P_l^k \sum_{ij\in\pi^k} q_{ij}r_{ij}}{P_l} \tag{23}$$

$$E[\hat{C}_l]=E[C_l]+\frac{\sum_k P_l^k \sum_{ij\in\pi^k} (mq_{ij}+1)r_{ij}}{P_l} \tag{24}$$

Let us denote the estimated completion time for terminal test case l as τ_l .Then the optimization problem is

$$\begin{aligned} &\text{maximize } \mathrm{P}[\hat{\mathrm{T}}_l \le \tau_l] \\ &s.t. \sum_k \sum_{ij\in\pi^k} r_{ij}^k \le K \\ &0 \le r_{ij} \le \overline{r}_{ij}, \forall ij \in \pi^k \end{aligned} \tag{25}$$

Where $\overline{r}_{ij}$ is the upper limit on the amount to be invested in test path *'ij'* such that $\overline{r}_{ij} \le K$ and $\sum \overline{r}_{ij} > K$ We assumed earlier that any additional investment i.e $\forall r_{ij} > 0$, reduces the expected execution time test path *'ij'* by a certain amount such that $\hat{E}[T_{ij}] < E[T_{ij}]$.

Consequently, $\mathrm{P}[\hat{\mathrm{T}}_l \le \tau_l] = F^{\sum_k \sum_{ij\in\pi^k} r_{ij}^k}_{\sum_k \pi_l^k}(\tau_l) > \mathrm{P}[\mathrm{T}_l \le \tau_l] = F^0_{\sum_k \pi_l^k}(\tau_l)$ .Furthermore, by *continuity and linearity* of $E[\hat{T}_{ij}] < E[T_{ij}]$ in r_{ij} we deduce that $\mathrm{P}[\hat{\mathrm{T}}_l \le \tau_l]$ is a *continuous and monotone non decreasing function* of $\sum_k \sum_{ij\in\pi^k} r_{ij}^k$ for any τ_l .Therefore maximizing $\mathrm{P}[\hat{\mathrm{T}}_l \le \tau_l]$ is equivalent to minimizing $E[\hat{T}_l]$ .Hence, we can rewrite the objective function of optimization problem defined above as $\text{maximize } E[T_l] - E[\hat{T}_l]$. An important point to note is that there might be investments which lead to the same reduction in execution time of a test path. Which investment to choose in such a case? To correct this problem, we modify the objective function further to $\underset{\mathrm{r}}{\text{maximize}} \ \dfrac{E[T_l] - E[\hat{T}_l]}{E[\hat{C}_l] - E[C_l]}$. Therefore the optimization problem in (25) can be re-written as

$$\begin{aligned} &\text{maximize} \\ &\qquad \frac{-\sum_k P_l^k \sum_{ij\in\pi^k} q_{ij} r_{ij}}{\sum_k P_l^k \sum_{ij\in\pi^k} (mq_{ij}+1) r_{ij}} \\ &\text{subject to} \\ &\qquad \sum_k \sum_{ij\in\pi^k} r_{ij}^k \le K \\ &\qquad 0 \le r_{ij} \le \overline{r}_{ij}, \forall ij \in \pi^k \end{aligned} \tag{26}$$

Since the objective function in (26) is non-linear we use a *Charnes-Cooper transformation* as shown by *Borza et all.* [14], to translate the linear-fractional program to an LP problem. We substitute

$$y = \frac{1}{\sum_{k} P_l^k \sum_{ij \in \pi^k} (mq_{ij} + 1) r_{ij}} \tag{27}$$

Therefore the optimization problem in (26) translates to the following Linear Programming Problem

$$\begin{aligned} &\text{maximize} \\ &\qquad -\sum_{k} P_l^k \sum_{ij \in \pi^k} q_{ij} r_{ij} y \\ &\text{subject to} \\ &\qquad \sum_{k} \sum_{ij \in \pi^k} r_{ij} y - Ky \le 0 \\ &\qquad y \ge 1 \\ &\qquad 0 \le r_{ij} y \le \overline{r}_{ij} y, \forall ij \in \pi^k \end{aligned} \tag{28}$$

8. Results & Discussion

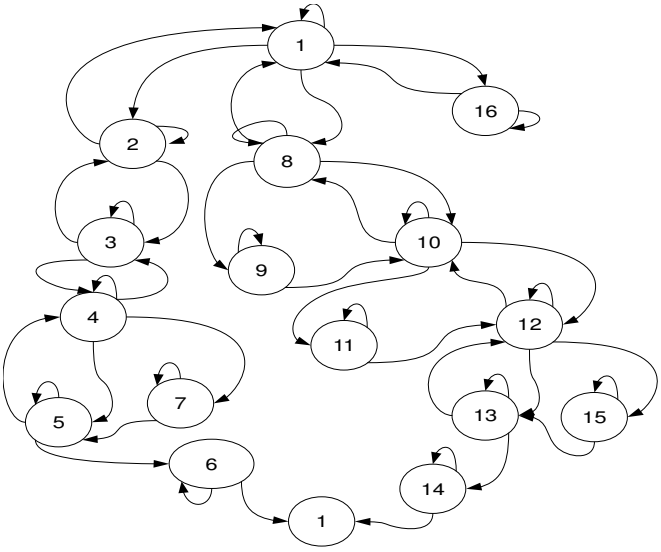


Figure 6: Test Execution Markov Chain

Table 2: Average test execution time (in minutes) for test case belonging to each complexity class (Simple, Medium and Complex)

Test Case Id	Complexity	Average Execution Time (minutes)
1	S	10
2	M	15
3	M	15
4	C	30
5	M	15
6	S	10
7	S	10
8	C	30
9	S	10
10	C	30
11	S	10
12	C	30
13	M	15
14	S	10
15	S	10
16	S	10

Table 3: The Transition Probability Matrix, P

p_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	5.00%	35.00%	0	0	0	0	0	50.00%	0	0	0	0	0	0	0	10.00%
2	5.00%	10.00%	85.00%	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	5.00%	10.00%	85.00%	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	10.00%	30.00%	45.00%	0	15.00%	0	0	0	0	0	0	0	0	0
5	0	0	0	5.00%	10.00%	85.00%	0	0	0	0	0	0	0	0	0	0
6	95.00%	0	0	0	0	5.00%	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	95.00%	0	5.00%	0	0	0	0	0	0	0	0	0
8	10.00%	0	0	0	0	0	0	30.00%	15.00%	45.00%	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	5.00%	95.00%	0	0	0	0	0	0
10	0	0	0	0	0	0	0	10.00%	0	30.00%	15.00%	45.00%	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	5.00%	95.00%	0	0	0	0
12	0	0	0	0	0	0	0	0	0	10.00%	0	30.00%	45.00%	0	15.00%	0
13	0	0	0	0	0	0	0	0	0	0	0	5.00%	10.00%	85.00%	0	0
14	95.00%	0	0	0	0	0	0	0	0	0	0	0	0	5.00%	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	95.00%	0	5.00%	0
16	95.00%	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.00%

Table 4: Stationary Distribution Π_*

	Test Case ID															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	S	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
Π_*	14.14%	5.86%	6.45%	8.22%	5.48%	4.90%	1.30%	11.75%	1.86%	11.56%	1.83%	10.40%	6.93%	6.20%	1.64%	1.49%

We consider a test suite of 16 test cases (Figure 6) and the average test execution rate as shown in Table 2. As described in section [4], we obtained the transition probability matrix P as shown in Table 3.Using the forward recurrence formula given in (3), we obtained the stationary distribution $\prod_*$ for each of the 16 test cases as show in the Table 4. Convergence was attained in 40 steps. It's important to note that the stationary distribution obtained for each test case corresponds to its complexity class. For instance, the system testing process has the highest probability (14.13%) of being in test case 1 (which is evident as it's the starting test case) followed by test case 8 (11.75%), which is a complex test case and has the highest number of sub branches in the test suite. Therefore the logic path of test case 8 has the highest probability of failure and the probability that the system testing process will be in test case 8 is the highest. Similarly, the model assigns a lower probability to test cases of medium complexity followed by simple test cases.

Next, we simulated a walk on the System Testing Markov Chain in discrete time using the algorithm presented in section (4.2.1) and plot a sample path. As we can see in Figure 7, the test execution process, visits test case 14 quite a few times. This is explained by the fact that the logic path to which test case 14 belongs to have the maximum number of test cases that belong to the complexity class 'C' (namely 8, 10 and 12) and therefore the probability that one or more test cases in this branch fail is the highest. This in turn results in test case 14 being visited the most number of times along with test case 1(which is evident as it acts a connector between the start and end of testing along a logic path).We also observe that system testing process oscillates a few times between test case 8 and test case 10 which is expected because test case 8 and test case 10 both belong to the complexity class 'C' and therefore have a high probability of failure. A close look at the plot also reveals some flat paths (at test case 12), which explains the phenomenon of retesting of a test case, while the jump to the previous test case (from test case 2 to test case 1) is explained by the fact that the test case under execution failed and the tester decided to test the test cases in the next logic path while the defect identified by test case 2 is being fixed.

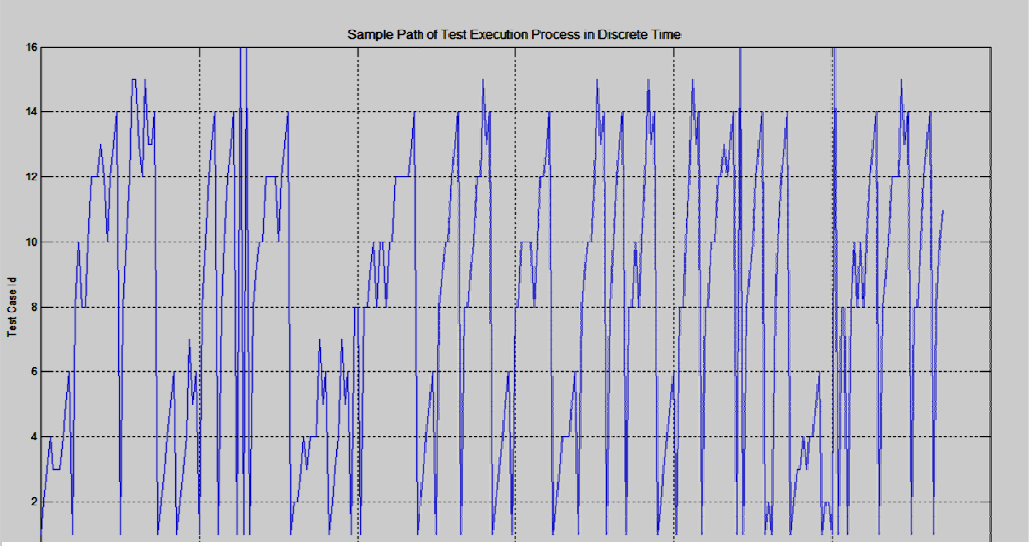


Figure 7: Sample Path of Test Execution Process in Discrete Time. The plot shows the jumps of the test execution process to each test case over time

We applied the Weak Law of Large Numbers given in equation (4) to obtain the expected number of steps to complete test execution. As can be seen in Figure 8, the average of the steps obtained from each run of the simulator converges in probability to the expected number of steps which is approximately 145. It is important to mention that we only consider convergence in probability because we choose $\varepsilon = 0.001$ which in turn allows the simulator to return the results in a limited number of recursions. However, the average of the steps obtained from each run of the simulator converges almost surely to the expected number of steps, if we allow the simulator to run for a considerably long time and with greater number of recursions. Though, it is important to note that as the size of the test suite grows large, the steps to convergence will also get larger. Moreover, test managers are mostly concerned with an approximate estimate of the average completion time; hence a convergence in probability would be sufficient.

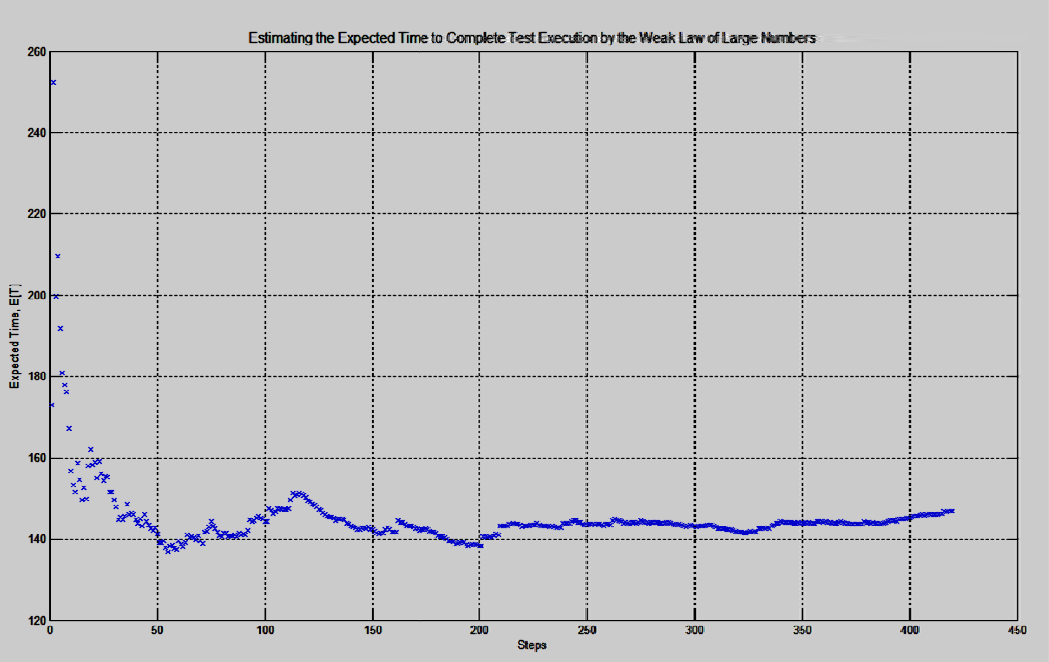


Figure 8: Convergence of the Expected Time, E[T] to Complete Test Execution

Then using the algorithm in section 5.3.1, we simulated a walk on the System Testing Markov Chain with the execution time being exponentially distributed with rate of execution as given in the table above. A plot of the sample path is shown in Figure 9. As can be seen, the test execution process starts at test case 1 and jumps to test case 8.However, test execution of test case 8 failed (it being a complex test case) and the system jumped back to test case 1.By the time, the system jumped back to test case 1, the defect that resulted in failure of test case 8 was fixed. However, since 1 has already been tested, it now just acts as transition case and no longer has an execution time (and thus does not add to the counting process), so the jump from test case 8 to test case 1 is shown by a steep line. During the retest though, test case 8 passes and execution moves on to the subsequent test cases in the logic path until it reaches test case 12, which fails and continues to fail the retests for a considerable amount of time. One important point to note here is that the system does not jump back to test case 10 (and subsequently to test case 11) during this process. This is because test case 12 is a complex test case while test case 11 is a simple test case. Thus the probability that the system would prioritize test case 12 and execution will stay in this test case even after multiple failures is high. Towards the extreme right of the plot, the jumps are steep, again due to the reason, that once a test case/sequence of test cases have passed, they do not have an execution time and merely act as a connector to the test cases that haven't yet passed.

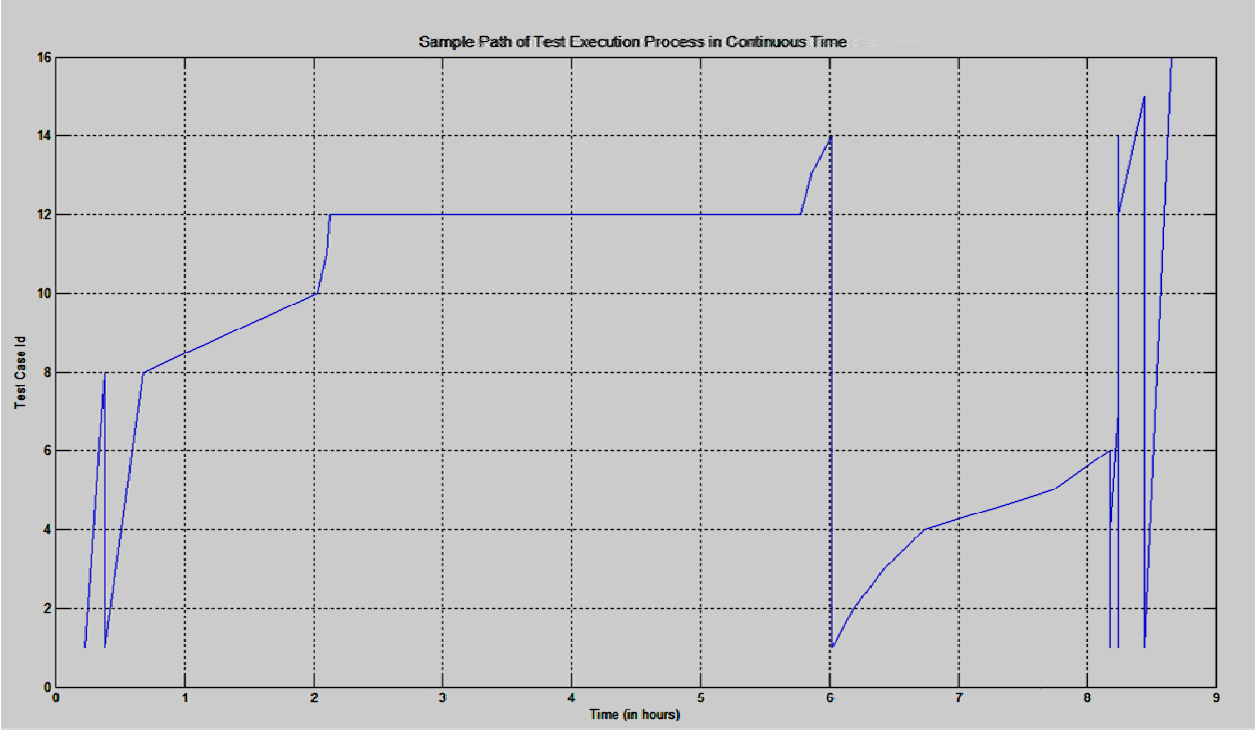


Figure 9: Sample Path of Test Execution Process in Continuous Time. The plot shows the jumps of the test execution process to each test case when the execution time is exponentially distributed

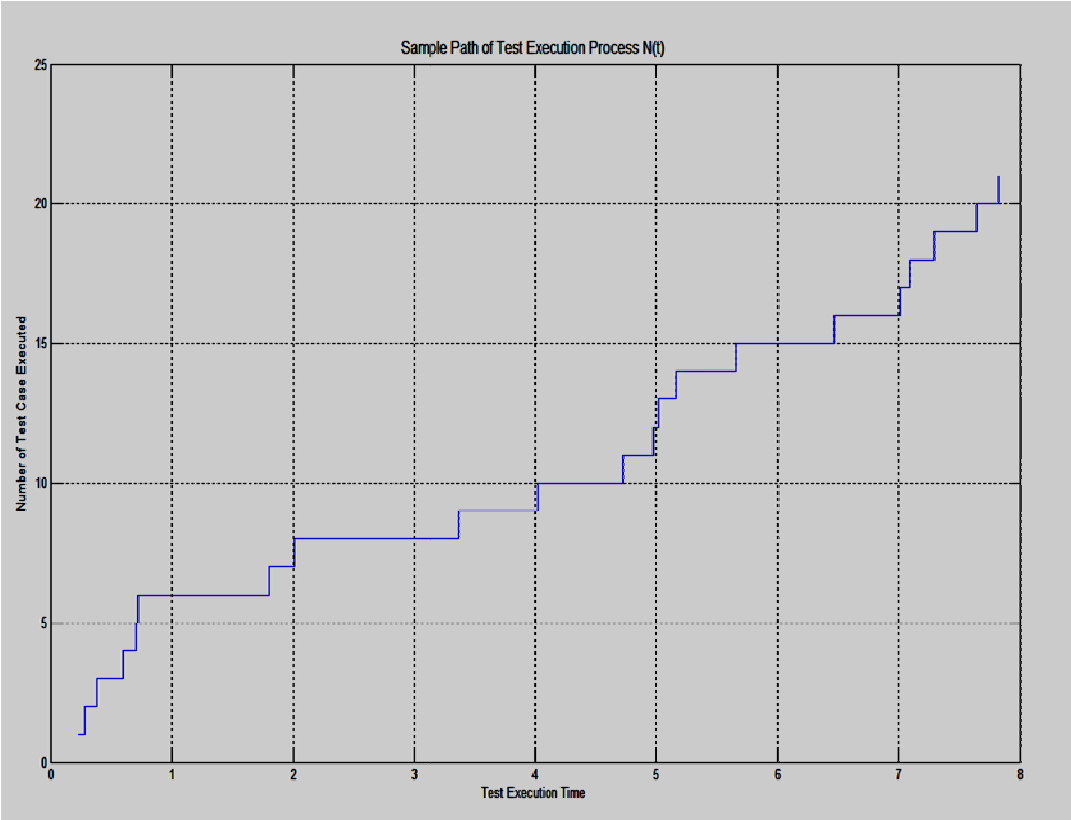


Figure 10: Sample Path of Number of Test Cases Executed v/s Elapsed Test Execution Time

As described in section 3, we also plot the sample path (Figure10) of number of test cases executed N_t and as expected the path is same as that of a Poisson Process. The plot N_t shows that the process attains a value of 20 test cases

while the total number of test cases in the test suite is 16. This is because as per the properties of a homogeneous Poisson Process described in section (2) we count retesting of a test cases as a new event of the counting process. Similar to the discrete case, we applied the Weak Law of Large Number as presented in equation (4) to obtain the expected time to complete test execution. As can be seen in the plot below, the average of the time obtained from each run of the simulator converges in probability to the expected time of approximately 5.6 hours. We chose $\varepsilon = 0.0001$ for the continuous time case. This is higher than the estimate obtained by the SMC method but the difference is not large and the estimate obtained by our model is more reliable because as shown above, it replicates the system testing process quite well by modeling the time to retest due to failure of test cases. Now, if we compute the probability of not completing test execution of all test cases within 5.6 hours even with the slowest rate of $\lambda = 2$, we get $P(T > 5.6) = e^{-5.6\lambda} = 1.37 \times 10^{-5}$ which is a very small probability of schedule overrun.

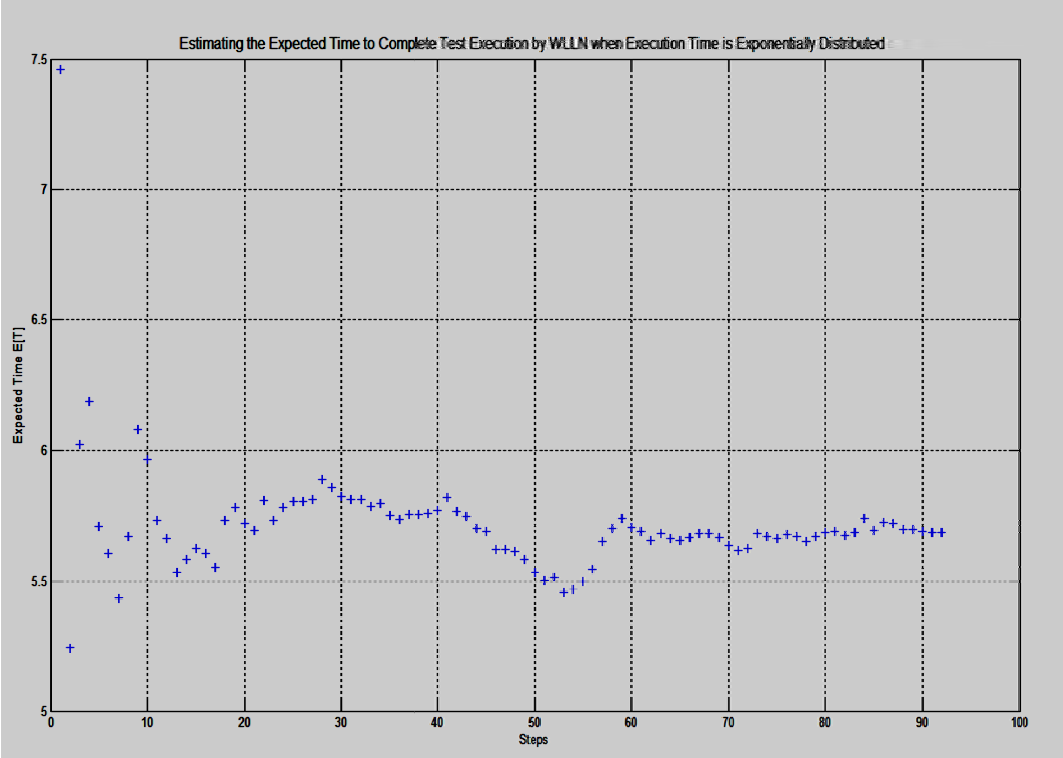


Figure 11: Convergence of Expected Time (in hours), E[T] to Complete Test Execution

The expected execution time for each test case $E[T_{ij}]$ is shown in Table 5. Therefore, using equation (5) we computed $E[W_j]$ for each test case as shown in Table 6. It is important to note that $E[W_j] = E[T_{ij}]$ in this example. This is because we have considered the test execution time to be exponentially distributed and do not differentiate in the rate of execution of test case *'i'* if the next test case in the execution chain is either a simple, medium or complex test case. If, however the expected execution time of test case *'i'*, $E[T_{ij}]$ is made dependent on the next test case *'j'* to be executed then the values in each row of the above matrix would be different and $E[W_j]$ would have a different value from $E[T_{ij}]$.The semi-Markov model of system testing accommodates this scenario as well

Table 5: Expected Test Execution Time (in hours)

$E[T_{ij}]$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1/6	1/6	0	0	0	0	0	1/6	0	0	0	0	0	0	0	1/6
2	1/4	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1/4	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1/2	1/2	1/2	0	1/2	0	0	0	0	0	0	0	0	0
5	0	0	0	1/4	1/4	1/4	0	0	0	0	0	0	0	0	0	0
6	1/6	0	0	0	0	1/6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1/6	0	1/6	0	0	0	0	0	0	0	0	0
8	1/2	0	0	0	0	0	0	1/2	1/2	1/2	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1/6	1/6	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1/2	0	1/2	1/2	1/2	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1/6	1/6	0	0	0	0
12	0	0	0	0	0	0	0	0	0	1/2	0	1/2	1/2	0	1/2	0
13	0	0	0	0	0	0	0	0	0	0	0	1/4	1/4	1/4	0	0
14	1/6	0	0	0	0	0	0	0	0	0	0	0	0	1/6	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	1/6	0	1/6	0
16	1/6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1/6

Table 6: Unconditional Expected Waiting Time at each Test Case (in hours)

	Test Case ID															
	S	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$E[W_j]$	1/6	1/4	1/4	1/2	1/4	1/6	1/6	1/2	1/6	1/2	1/6	1/2	1/4	1/6	1/6	1/6

Next, we computed the limiting transition probabilities ϕ_j by equation (9) and the values are presented in Table 7. An important inference from the result of the limiting transition probability ϕ_j is that it computes the probability of being in test case 1 to be 7.2 % in contrast to the probability of 14.13% obtained from the stationary distribution $\prod_*$.Though by design test case 1 acts as a connecting state for most test cases, its complexity is simple, which means that a test manager observing the system testing process for a long time will see test execution at test case 1 with low probability as compared to complex test cases.

Table 7: Limiting Transition Probability ϕ_j

	Test Case ID															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	S	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
ϕ_j	7.21%	4.48%	4.93%	12.56%	4.19%	2.50%	0.66%	17.97%	0.95%	17.67%	0.93%	15.90%	5.30%	3.16%	0.84%	0.76%

As expected, the test execution process has the highest probability of being in test case 8 (17.96%) as it has the highest complexity (given by the fact that it is the parent of two complex test cases namely test case 10 and test case 12). The other complex test cases, 10, 12 and 4 are assigned probabilities in decreasing order which corresponds to the complexity of test cases linked to each of them. Similarly, test cases of medium complexity have a higher limiting transition probability than simple test cases. If we look at the limiting transition probabilities of test case 2 and test case 3 both of which belong to the complexity class of medium, we see that test case 3 has a higher limiting transition probability(4.9%) than test case 2(4.4%).This is because, test case 4 which is the immediate test case that follows 3 is a complex test case and therefore has a higher probability of failure whereas that test case 2 is linked to test cases 3 and 4, which are both simple test cases. Similarly, if we compare the limiting transition probabilities of two simple test cases, namely test case 6 and test case 9, we observe that test case 6(2.4%) has a higher limiting transition probability than test case 9(0.9%) which is again explained by the fact that test case 9 belongs to a much shorter logic path than test case 6 and the probability of one or more test cases failing

in the logic path of test case 6 is higher than that of test case 9.We obtained the average waiting time at a test case (or the expected time between two jumps of the test execution process) as $\sum_{j=1}^{N}\prod_j E[W_j] = 19.62$ minutes. Therefore the average time to complete test execution would be $N\sum_{j=1}^{N}\prod_j E[W_j] = 5.23$ hours which is approximates to the expected completion time of 5.6 hours obtained from simulation.

Table 8: Cost of test execution per hour

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	S	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
C_j	€ 10.00	€ 30.00	€ 30.00	€ 45.00	€ 30.00	€ 20.00	€ 5.00	€ 60.00	€ 5.00	€ 55.00	€ 5.00	€ 50.00	€ 30.00	€ 10.00	€ 5.00	€ 5.00

Table 8 shows the hourly cost of executing each test case. We obtained the average cost of executing a test case as $\sum_{j=1}^{N}\prod_j E[W_j]c_j = 13.58$ euros. This results in an average cost of executing all 16 test cases as $N\sum_{j=1}^{N}\prod_j E[W_j]c_j = 217.24$ euros. Then using equation (17), we divided the average cost of executing a test case by the average waiting time of 0.327 hours(19.62 minutes) to obtained the long run average cost (stationary cost) of executing a test case as, $g = 41.51$ euros which in turn results in the long run average cost (stationary cost) of executing all 16 test cases as $Ng = 644.28$ euros. We contrast this to the computation where the test manager only takes into account the execution time of each test case and multiplies it with the cost per hour ($\sum_{j=1}^{N} E[W_j]c_j$) thereby underestimating the budget to €145.83 , thus failing to take into account the fact that failure of a test case results in retesting and hence adds to the cost. As shown in section 6.4, this further necessitates the need to have a control model for system testing in order to ensure that system testing is on track.

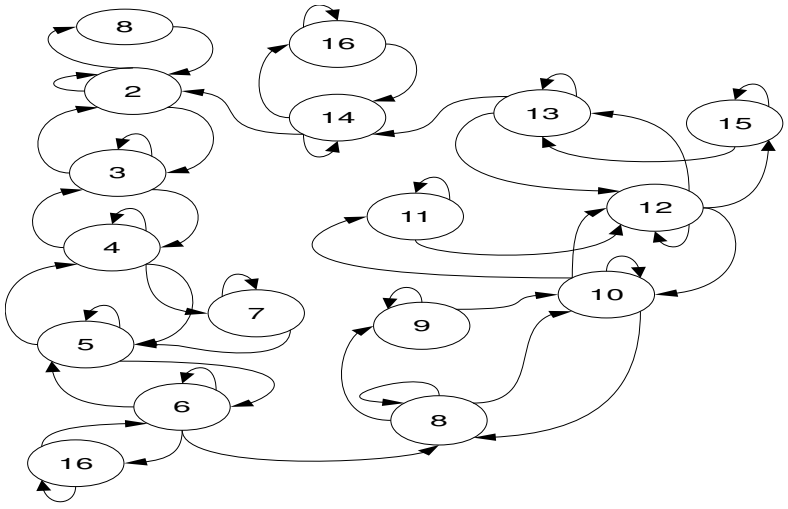


Figure 12: Test Execution Markov Chain

Test Case 1 is a simple test case (and in most cases the test initiation state) and after it passes, it acts just as a transition test cases without adding to the test execution time. Therefore, we remodel the test execution chain as shown in Figure 12, wherein we remove test case 1 and connect test case 6 to test case 8 and test case 16. Therefore, once a terminal test case on an execution branch is reached, execution can jump to the next branch without having to pass through a transition case. Similarly, once test execution reaches test case 14, it jumps to the first test case of the next branch i.e test case 2 or test case 16.

The transition probability matrix P , the expected execution time of each test case $E[T_{ij}]$ and the expected time to complete test execution E[T] for the remodeled test execution chain of Figure 12 is given in Table 9 , Table 10 and Figure 13 respectively. We obtained the expected time to complete test execution E[T]=5.7 hours.

Table 9: Transition Probability Matrix, P

p_{ij}	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	10.00%	85.00%	0	0	0	0	5.00%	0	0	0	0	0	0	0	0
3	5.00%	10.00%	85.00%	0	0	0	0	0	0	0	0	0	0	0	0
4	0	10.00%	30.00%	45.00%	0	15.00%	0	0	0	0	0	0	0	0	0
5	0	0	5.00%	10.00%	85.00%	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	5.00%	0	85.00%	0	0	0	0	0	0	0	10.00%
7	0	0	0	95.00%	0	5.00%	0	0	0	0	0	0	0	0	0
8	10.00%	0	0	0	0	0	30.00%	15.00%	45.00%	0	0	0	0	0	0
9	0	0	0	0	0	0	0	5.00%	95.00%	0	0	0	0	0	0
10	0	0	0	0	0	0	10.00%	0	30.00%	15.00%	45.00%	0	0	0	0
11	0	0	0	0	0	0	0	0	0	5.00%	95.00%	0	0	0	0
12	0	0	0	0	0	0	0	0	10.00%	0	30.00%	45.00%	0	15.00%	0
13	0	0	0	0	0	0	0	0	0	0	5.00%	10.00%	85.00%	0	0
14	85.00%	0	0	0	0	0	0	0	0	0	0	0	5.00%	0	10.00%
15	0	0	0	0	0	0	0	0	0	0	0	95.00%	0	5.00%	0
16	0	0	0	0	47.50%	0	0	0	0	0	0	0	47.50%	0	5.00%

Table 10: Expected Test Execution Time (in hours)

$E[T_{ij}]$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1/4	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1/2	0	0	1/2	0	0	0	0	0	0	0	0	0
5	0	0	0	1/4	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1/6	0	0	0	0	0	0	0	1/6
7	0	0	0	1/6	0	1/6	0	0	0	0	0	0	0	0	0
8	1/2	0	0	0	0	0	1/2	1/2	1/2	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1/6	1/6	0	0	0	0	0	0
10	0	0	0	0	0	0	1/2	0	1/2	1/2	1/2	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1/6	1/6	0	0	0	0
12	0	0	0	0	0	0	0	0	1/2	0	1/2	1/2	0	1/2	0
13	0	0	0	0	0	0	0	0	0	0	1/4	1/4	1/4	0	0
14	1/6	0	0	0	0	0	0	0	0	0	0	0	1/6	0	1/6
15	0	0	0	0	0	0	0	0	0	0	0	1/6	0	1/6	0
16	0	0	0	0	1/6	0	0	0	0	0	0	0	1/6	0	1/6

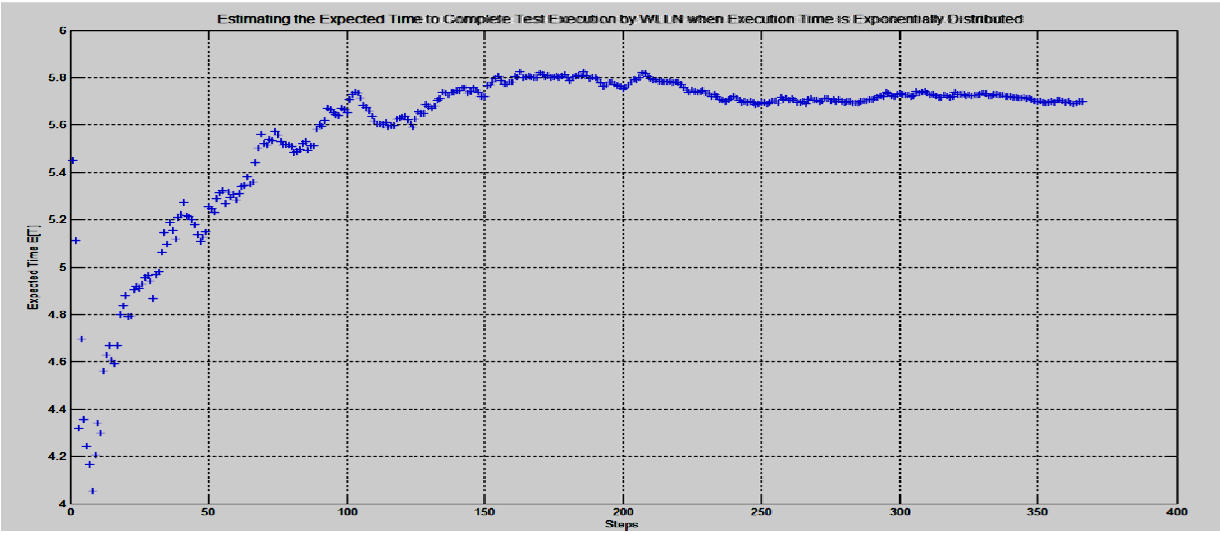


Figure 13: Convergence of Expected Waiting Time (in hours), E[T] to Complete Test Execution

Moreover, we obtained the stationary distribution, the unconditional expected execution time and the limiting transition probability for the remodeled Markov chain as shown in Table 11. In contrast to the limiting probabilities (Table 7) of the first model, the probabilities are evenly distributed with test cases of medium complexity having a higher limiting probability. We obtained the average waiting time at a test case $\sum_{j=1}^{N} \prod_j E[W_j] = 20.56$ minutes. Using the same cost matrix as shown in Table 8, we obtained g=42.65 euros.

Table 11: Stationary Distribution Π*, Unconditional Expected Test Execution Time, E[Wj] and Limiting Transition Probability ϕj

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
Π*	8.29%	9.12%	11.63%	7.75%	7.71%	1.84%	11.57%	1.83%	11.38%	1.80%	10.24%	6.83%	6.87%	1.62%	1.53%
E[Wj]	1/4	1/4	1/2	1/4	1/6	1/6	1/2	1/6	1/2	1/6	1/2	1/4	1/6	1/6	1/6
ϕj	6.04%	6.65%	16.97%	5.66%	3.75%	0.89%	16.88%	0.89%	16.60%	0.87%	14.94%	4.98%	3.34%	0.79%	0.75%

Next, we defined the control policy as shown in equation (18). Therefore, for $\mu_i = 1 \forall i \in \aleph$, the transition probability p_{ij}^1, the expected execution time of each test cases $E[T_{ij}^1]$ and the expected time to complete test execution $E[T^1]$ is Table 12, Table 13 and Figure 14 respectively. As can be seen in Figure 14, the expected time to complete test execution under additional investment reduces to 3.7 hours. Similarly, we obtained the stationary distribution Π_*^1, the unconditional expected execution time $E[W_j^1]$ and the limiting transition probability ϕ_j^1 as shown in Table 14.The average waiting time at a test case under control policy $\mu_i = 1$ is $\sum_{j=1}^{N} \prod_j^1 E[W_j^1] = 14.10$ minutes and using the cost matrix defined in Table 15 we obtained the stationary cost of executing a test case under policy $\mu_i = 1$ as g^1=78.19 euros.

Table 12: Transition Probability Matrix, P under control policy, μi=1

μ=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	5.00%	94.00%	0	0	0	0	1.00%	0	0	0	0	0	0	0	0
3	1.00%	5.00%	94.00%	0	0	0	0	0	0	0	0	0	0	0	0
4	0	5.00%	10.00%	65.00%	0	20.00%	0	0	0	0	0	0	0	0	0
5	0	0	1.00%	5.00%	94.00%	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1.00%	0	89.00%	0	0	0	0	0	0	0	10.00%
7	0	0	0	99.00%	0	1.00%	0	0	0	0	0	0	0	0	0
8	5.00%	0	0	0	0	0	10.00%	20.00%	65.00%	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1.00%	99.00%	0	0	0	0	0	0
10	0	0	0	0	0	0	5.00%	0	10.00%	20.00%	65.00%	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1.00%	99.00%	0	0	0	0
12	0	0	0	0	0	0	0	0	5.00%	0	10.00%	65.00%	0	20.00%	0
13	0	0	0	0	0	0	0	0	0	0	1.00%	5.00%	94.00%	0	0
14	89.00%	0	0	0	0	0	0	0	0	0	0	0	1.00%	0	10.00%
15	0	0	0	0	0	0	0	0	0	0	0	99.00%	0	1.00%	0
16	0	0	0	0	49.50%	0	0	0	0	0	0	0	49.50%	0	1.00%

Table 13: Expected Test Execution Time (in hours) under control policy, μ_i=1

E[T_{ij}]	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	1/5	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1/5	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1/3	0	0	1/3	0	0	0	0	0	0	0	0	0
5	0	0	0	1/5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1/7	0	0	0	0	0	0	0	1/7
7	0	0	0	1/7	0	1/7	0	0	0	0	0	0	0	0	0
8	1/3	0	0	0	0	0	1/3	1/3	1/3	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1/7	1/7	0	0	0	0	0	0
10	0	0	0	0	0	0	1/3	0	1/3	1/3	1/3	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1/7	1/7	0	0	0	0
12	0	0	0	0	0	0	0	0	1/3	0	1/3	1/3	0	1/3	0
13	0	0	0	0	0	0	0	0	0	0	1/5	1/5	1/5	0	0
14	1/7	0	0	0	0	0	0	0	0	0	0	0	1/7	0	1/7
15	0	0	0	0	0	0	0	0	0	0	0	1/7	0	1/7	0
16	0	0	0	0	1/7	0	0	0	0	0	0	0	1/7	0	1/7

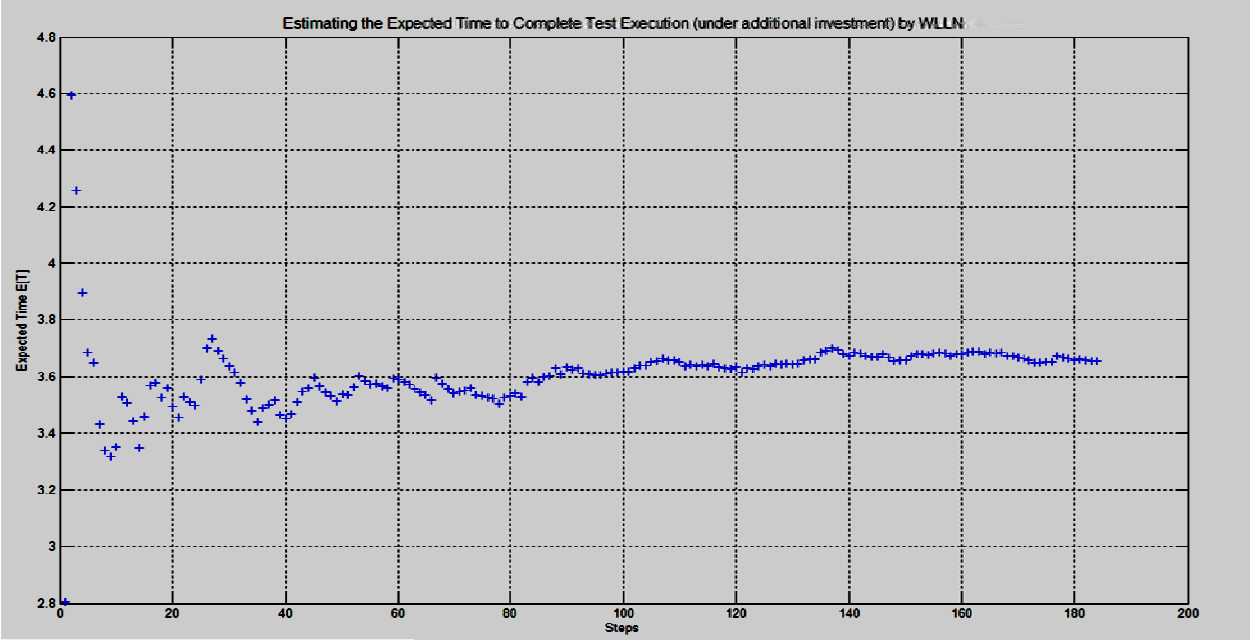


Figure 14: Convergence of Expected Time to Complete Test Execution under control policy, μ_i=1

Table 14: Stationary Distribution Π_*, Unconditional Expected Test Execution Time, E[W_j] and Limiting Transition Probability ϕ_j under control policy, μ_i=1

μ=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
Π^*	8.73%	9.15%	9.65%	8.64%	9.10%	1.95%	9.63%	1.95%	9.60%	1.94%	9.16%	8.19%	8.68%	1.85%	1.80%
E[W_j]	1/5	1/5	1/3	1/5	1/7	1/7	1/3	1/7	1/3	1/7	1/3	1/5	1/7	1/7	1/7
ϕ_j	7.43%	7.78%	13.68%	7.34%	5.53%	1.18%	13.65%	1.18%	13.61%	1.18%	12.98%	6.97%	5.27%	1.12%	1.09%

Table 15: Cost of Test Execution (per hour) under control policy, μ_i=1

μ=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
c_j	€ 60.00	€ 60.00	€ 90.00	€ 60.00	€ 40.00	€ 10.00	€ 120.00	€ 10.00	€ 110.00	€ 10.00	€ 100.00	€ 60.00	€ 20.00	€ 10.00	€ 10.00

As described in section 6.5, as $t \to \infty$, the probability that test cases will keep failing would tend to 0.Therefore we obtain the asymptotic transition probability matrix as shown in Table 16 and Table 17 for control policy $\mu_i = 0$ and $\mu_i = 1$ respectively.

Table 16: Transition Probability Matrix when t→∞ and control μ_i=0

μ=0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	85.00%	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	85.00%	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	45.00%	0	15.00%	0	0	0	0	0	0	0	0	0
5	0	0	0	0	85.00%	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	85.00%	0	0	0	0	0	0	0	10.00%
7	0	0	0	95.00%	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	15.00%	45.00%	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	95.00%	0	0	0	0	0	0
10	0	0	0	0	0	0		0	0	15.00%	45.00%	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	95.00%	0	0	0	0
12	0	0	0	0	0	0	0	0		0	0	45.00%	0	15.00%	0
13	0	0	0	0	0	0	0	0	0	0	0	0	85.00%	0	0
14	85.00%	0	0	0	0	0	0	0	0	0	0	0	0	0	10.00%
15	0	0	0	0	0	0	0	0	0	0	0	95.00%	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17: Transition Probability Matrix when t→∞ and control μ_i=1

μ=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	94.00%	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	94.00%	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	65.00%	0	20.00%	0	0	0	0	0	0	0	0	0
5	0	0	0	0	94.00%	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	89.00%	0	0	0	0	0	0	0	10.00%
7	0	0	0	99.00%	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	20.00%	65.00%	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	99.00%	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	20.00%	65.00%	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	99.00%	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	65.00%	0	20.00%	0
13	0	0	0	0	0	0	0	0	0	0	0	0	94.00%	0	0
14	89.00%	0	0	0	0	0	0	0	0	0	0	0	0	0	10.00%
15	0	0	0	0	0	0	0	0	0	0	0	99.00%	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Moreover, by equation (21), we computed the expected long run costs $E[-c_i^{\mu}], \forall i, \mu$ and the results are presented in Table 18.The expected long run cost for each test case is presented along the diagonal elements of the matrix in Table 18. The values in the columns other than the diagonal entries are the values obtained from the Bellman equation of (21) for $j = 1 \ldots N$. The cells highlighted in green are the maximum value of $E[-c_i^{\mu}]$ across the two policies (without and with investment). These results in the optimal policy vector shown in Table 19.Therefore we obtained an optimal policy wherein additional should be made in the terminal test cases 6 and 14.Thus, when additional investment doubles the cost function as shown in Table 15, additional investment in test cases that connect to the next execution branch minimizes the average expected long run costs.

Table 18: Average expected long run cost when starting at test case 'i' and under control policy μ. The table shows the long run expected difference in costs when starting at test case 'i' and starting in stationarity

μ=0	E[-c_i]														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	**€ 5.68**	€ 2.96	(€ 0.24)	(€ 0.01)	(€ 3.74)	€ 6.26	(€ 9.58)	€ 3.31	(€ 3.12)	€ 9.66	€ 3.56	€ 10.61	€ 8.76	€ 16.36	€ 6.28
3	€ 5.85	**€ 3.11**	(€ 0.06)	€ 0.28	(€ 3.39)	€ 6.54	(€ 9.17)	€ 3.96	(€ 2.44)	€ 10.75	€ 4.71	€ 12.55	€ 11.05	€ 18.20	€ 6.28
4	€ 5.01	€ 2.17	**(€ 0.12)**	€ 0.19	(€ 3.50)	€ 6.45	(€ 9.30)	€ 3.76	(€ 2.65)	€ 10.41	€ 4.35	€ 11.94	€ 10.33	€ 17.62	€ 6.28
5	€ 7.04	€ 4.56	€ 1.65	**€ 0.41**	(€ 3.24)	€ 9.28	(€ 9.00)	€ 4.25	(€ 2.14)	€ 11.23	€ 5.22	€ 13.41	€ 12.06	€ 19.02	€ 6.28
6	€ 8.42	€ 6.18	€ 3.55	€ 6.37	**(€ 3.07)**	€ 12.33	(€ 8.79)	€ 4.58	(€ 1.79)	€ 11.79	€ 5.81	€ 14.41	€ 13.23	€ 19.96	€ 6.28
7															
8	€ 5.96	€ 3.29	€ 0.15	€ 0.64	(€ 2.97)	€ 6.89	**(€ 9.16)**	€ 3.99	(€ 2.41)	€ 10.79	€ 4.76	€ 12.63	€ 11.14	€ 18.28	€ 6.28
9															
10	€ 5.12	€ 2.31	(€ 1.01)	(€ 1.32)	(€ 5.27)	€ 5.02	(€ 11.39)	€ 0.41	**(€ 2.62)**	€ 10.45	€ 4.40	€ 12.03	€ 10.43	€ 17.70	€ 6.28
11															
12	€ 4.90	€ 2.04	(€ 1.32)	(€ 1.85)	(€ 5.90)	€ 4.52	(€ 12.12)	(€ 0.76)	(€ 7.40)	€ 2.79	**€ 4.30**	€ 11.86	€ 10.23	€ 17.55	€ 6.28
13	€ 5.20	€ 2.40	(€ 0.90)	(€ 1.13)	(€ 5.05)	€ 5.20	(€ 11.13)	€ 0.83	(€ 5.74)	€ 5.46	(€ 0.85)	**€ 12.08**	€ 10.49	€ 9.28	€ 6.28
14	€ 5.50	€ 2.75	(€ 0.49)	(€ 0.44)	(€ 4.24)	€ 5.86	(€ 10.17)	€ 2.37	(€ 4.11)	€ 8.07	€ 1.89	€ 7.79	**€ 10.75**	€ 13.68	€ 6.28
15															
16															

μ=1	E[-c_i]														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	**€ 3.95**	€ 0.33	(€ 3.52)	(€ 1.81)	(€ 5.80)	€ 7.95	(€ 13.75)	€ 7.69	(€ 2.08)	€ 17.42	€ 7.75	€ 15.42	€ 12.53	€ 25.00	€ 9.74
3	€ 7.06	**€ 1.49**	(€ 2.28)	(€ 0.35)	(€ 4.25)	€ 9.39	(€ 12.00)	€ 9.73	(€ 0.02)	€ 19.82	€ 10.18	€ 18.28	€ 15.57	€ 27.84	€ 9.74
4	€ 3.59	(€ 0.06)	**(€ 3.54)**	(€ 1.84)	(€ 5.83)	€ 7.92	(€ 13.77)	€ 7.66	(€ 2.11)	€ 17.38	€ 7.72	€ 15.37	€ 12.48	€ 24.96	€ 9.74
5	€ 8.03	€ 4.68	€ 1.10	**€ 0.06**	(€ 3.81)	€ 13.35	(€ 11.50)	€ 10.31	€ 0.57	€ 20.51	€ 10.87	€ 19.09	€ 16.44	€ 28.64	€ 9.74
6	€ 11.88	€ 8.76	€ 5.45	€ 8.77	**(€ 2.06)**	€ 18.42	(€ 9.54)	€ 12.59	€ 2.88	€ 23.21	€ 13.60	€ 22.31	€ 19.86	€ 31.83	€ 9.74
7															
8	€ 3.83	€ 0.20	(€ 3.66)	(€ 1.97)	(€ 5.97)	€ 7.79	**(€ 13.65)**	€ 7.80	(€ 1.96)	€ 17.55	€ 7.89	€ 15.58	€ 12.70	€ 25.16	€ 9.74
9															
10	(€ 0.58)	(€ 4.49)	(€ 8.65)	(€ 7.86)	(€ 12.24)	€ 1.96	(€ 20.97)	(€ 0.75)	**(€ 4.62)**	€ 14.45	€ 4.76	€ 11.88	€ 8.77	€ 21.51	€ 9.74
11															
12	(€ 2.82)	(€ 6.88)	(€ 11.19)	(€ 10.85)	(€ 15.42)	(€ 1.00)	(€ 24.55)	(€ 4.92)	(€ 14.81)	€ 2.55	**€ 3.17**	€ 10.01	€ 6.78	€ 19.65	€ 9.74
13	(€ 0.55)	(€ 4.46)	(€ 8.62)	(€ 7.82)	(€ 12.19)	€ 2.00	(€ 20.93)	(€ 0.70)	(€ 10.54)	€ 7.53	(€ 2.23)	**€ 11.91**	€ 8.80	€ 13.35	€ 9.74
14	€ 2.43	(€ 1.28)	(€ 5.24)	(€ 3.83)	(€ 7.95)	€ 5.95	(€ 16.16)	€ 4.87	(€ 4.92)	€ 14.09	€ 4.40	€ 11.46	**€ 11.46**	€ 21.08	€ 9.74
15															
16															

Table 19: The Optimal Policy Vector

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
μ	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**

Next, we only changed the cost function wherein additional investment increases the cost function by a factor of 1.5 instead of a factor of 2. The new cost function is shown in Table 20.Under the new cost function we obtained g^1 =46.20 euros. Again, we computed $E[-c_i^{\mu}], \forall i, \mu$ using the Bellman equation in (21) and the results are shown in Table 21. Therefore as shown in Table 22, additional investment in test case 6 and test case 8 minimizes the average long run cost

Table 20: Cost of Test Execution (per hour) under control policy, μ_i=1

μ=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
c_i	€ 36.00	€ 36.00	€ 54.00	€ 36.00	€ 24.00	€ 6.00	€ 72.00	€ 6.00	€ 66.00	€ 6.00	€ 60.00	€ 36.00	€ 12.00	€ 6.00	€ 6.00

Table 21: Average expected long run cost when starting at test case 'i' and under control policy μ. The table shows the long run expected difference in costs when starting at test case 'i' and starting in stationarity

μ=0	E[-cᵢ]														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	**€5.68**	€2.96	(€0.24)	(€0.01)	(€3.74)	€6.26	(€9.58)	€3.31	(€3.12)	€9.66	€3.56	€10.61	€8.76	€16.36	€6.28
3	€5.85	**€3.11**	(€0.06)	€0.28	(€3.39)	€6.54	(€9.17)	€3.96	(€2.44)	€10.75	€4.71	€12.55	€11.05	€18.20	€6.28
4	€5.01	€2.17	**(€0.12)**	€0.19	(€3.50)	€6.45	(€9.30)	€3.76	(€2.65)	€10.41	€4.35	€11.94	€10.33	€17.62	€6.28
5	€7.04	€4.56	€1.65	**€0.41**	(€3.24)	€9.28	(€9.00)	€4.25	(€2.14)	€11.23	€5.22	€13.41	€12.06	€19.02	€6.28
6	€8.42	€6.18	€3.55	€6.37	**(€3.07)**	€12.33	(€8.79)	€4.58	(€1.79)	€11.79	€5.81	€14.41	€13.23	€19.96	€6.28
7															
8	€5.96	€3.29	€0.15	€0.64	(€2.97)	€6.89	**(€9.16)**	€3.99	(€2.41)	€10.79	€4.76	€12.63	€11.14	€18.28	€6.28
9															
10	€5.12	€2.31	(€1.01)	(€1.32)	(€5.27)	€5.02	(€11.39)	€0.41	**(€2.62)**	€10.45	€4.40	€12.03	€10.43	€17.70	€6.28
11															
12	€4.90	€2.04	(€1.32)	(€1.85)	(€5.90)	€4.52	(€12.12)	(€0.76)	(€7.40)	€2.79	**€4.30**	€11.86	€10.23	€17.55	€6.28
13	€5.20	€2.40	(€0.90)	(€1.13)	(€5.05)	€5.20	(€11.13)	€0.83	(€5.74)	€5.46	(€0.85)	**€12.08**	€10.49	€9.28	€6.28
14	€5.50	€2.75	(€0.49)	(€0.44)	(€4.24)	€5.86	(€10.17)	€2.37	(€4.11)	€8.07	€1.89	€7.79	**€10.75**	€13.68	€6.28
15															
16															

μ=1	E[-cᵢ]														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	**€2.37**	€0.20	(€2.11)	(€1.09)	(€3.48)	€4.77	(€8.25)	€4.61	(€1.25)	€10.45	€4.65	€9.25	€7.52	€15.00	€5.85
3	€4.24	**€0.90**	(€1.37)	(€0.21)	(€2.55)	€5.64	(€7.20)	€5.84	(€0.01)	€11.89	€6.11	€10.97	€9.34	€16.70	€5.85
4	€2.15	(€0.03)	**(€2.13)**	(€1.10)	(€3.50)	€4.75	(€8.26)	€4.59	(€1.27)	€10.43	€4.63	€9.22	€7.49	€14.98	€5.85
5	€4.82	€2.81	€0.66	**€0.04**	(€2.28)	€8.01	(€6.90)	€6.18	€0.34	€12.30	€6.52	€11.46	€9.86	€17.19	€5.85
6	€7.13	€5.26	€3.27	€5.26	**(€1.24)**	€11.05	(€5.73)	€7.56	€1.73	€13.92	€8.16	€13.39	€11.92	€19.10	€5.85
7															
8	€2.30	€0.12	(€2.19)	(€1.18)	(€3.58)	€4.67	**(€8.19)**	€4.68	(€1.18)	€10.53	€4.73	€9.35	€7.62	€15.10	€5.85
9															
10	(€0.35)	(€2.70)	(€5.19)	(€4.72)	(€7.34)	€1.18	(€12.58)	(€0.45)	**(€2.77)**	€8.67	€2.86	€7.13	€5.26	€12.90	€5.85
11															
12	(€1.69)	(€4.13)	(€6.71)	(€6.51)	(€9.25)	(€0.60)	(€14.73)	(€2.95)	(€8.89)	€1.53	**€1.90**	€6.01	€4.07	€11.79	€5.85
13	(€0.33)	(€2.68)	(€5.17)	(€4.69)	(€7.32)	€1.20	(€12.56)	(€0.42)	(€6.33)	€4.52	(€1.34)	**€7.14**	€5.28	€8.01	€5.85
14	€1.46	(€0.77)	(€3.14)	(€2.30)	(€4.77)	€3.57	(€9.70)	€2.92	(€2.95)	€8.46	€2.64	€6.87	**€6.87**	€12.65	€5.85
15															
16															

Table 22: Optimal Policy Vector

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
μ	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Additionally, we also considered the case where under policy $\mu_i = 1$, the transition probability matrix is the same as the transition probability matrix under policy $\mu_i = 0$ i.e $p_{ij}^0 = p_{ij}^1$.As a consequence we obtained the stationary distribution, unconditional execution time of a test case and the limiting probability as shown in Table 23.

Table 23: Stationary Distribution Π*, Unconditional Expected Test Execution Time, E[Wⱼ] and Limiting Transition Probability ϕ_j under control policy, μ_i=1

μ=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
∏*	8.29%	9.12%	11.63%	7.75%	7.71%	1.84%	11.57%	1.83%	11.38%	1.80%	10.24%	6.83%	6.87%	1.62%	1.53%
E[Wⱼ]	1/5	1/5	1/3	1/5	1/7	1/7	1/3	1/7	1/3	1/7	1/3	1/5	1/7	1/7	1/7
ϕ_j	6.72%	7.40%	15.73%	6.29%	4.47%	1.06%	15.65%	1.06%	15.39%	1.04%	13.85%	5.54%	3.98%	0.94%	0.89%

Furthermore, we obtained the average waiting time at a test case $\sum_{j=1}^{N} \prod_j^1 E[W_j^1] = 14.79$ minutes and $g^1 = 49.41$ euros (the cost matrix being the same as the one given in Table20).

Again using the Bellman equation in (21) we computed $E[-c_i^{\mu}], \forall i, \mu$ as shown in Table 24. Therefore as shown in Table 25, additional investment in test cases 5, 6, 8 and 10 minimizes the average long run cost

Table 24: Average expected long run cost when starting at test case 'i' and under control policy μ. The table shows the long run expected difference in costs when starting at test case 'i' and starting in stationarity

μ=0	E[-cᵢ]														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	**€5.68**	€2.96	(€0.24)	(€0.01)	(€3.74)	€6.26	(€9.58)	€3.31	(€3.12)	€9.66	€3.56	€10.61	€8.76	€16.36	€6.28
3	€5.85	**€3.11**	(€0.06)	€0.28	(€3.39)	€6.54	(€9.17)	€3.96	(€2.44)	€10.75	€4.71	€12.55	€11.05	€18.20	€6.28
4	€5.01	€2.17	**(€0.12)**	€0.19	(€3.50)	€6.45	(€9.30)	€3.76	(€2.65)	€10.41	€4.35	€11.94	€10.33	€17.62	€6.28
5	€7.04	€4.56	€1.65	**€0.41**	(€3.24)	€9.28	(€9.00)	€4.25	(€2.14)	€11.23	€5.22	€13.41	€12.06	€19.02	€6.28
6	€8.42	€6.18	€3.55	€6.37	**(€3.07)**	€12.33	(€8.79)	€4.58	(€1.79)	€11.79	€5.81	€14.41	€13.23	€19.96	€6.28
7															
8	€5.96	€3.29	€0.15	€0.64	(€2.97)	€6.89	**(€9.16)**	€3.99	(€2.41)	€10.79	€4.76	€12.63	€11.14	€18.28	€6.28
9															
10	€5.12	€2.31	(€1.01)	(€1.32)	(€5.27)	€5.02	(€11.39)	€0.41	**(€2.62)**	€10.45	€4.40	€12.03	€10.43	€17.70	€6.28
11															
12	€4.90	€2.04	(€1.32)	(€1.85)	(€5.90)	€4.52	(€12.12)	(€0.76)	(€7.40)	€2.79	**€4.30**	€11.86	€10.23	€17.55	€6.28
13	€5.20	€2.40	(€0.90)	(€1.13)	(€5.05)	€5.20	(€11.13)	€0.83	(€5.74)	€5.46	(€0.85)	**€12.08**	€10.49	€9.28	€6.28
14	€5.50	€2.75	(€0.49)	(€0.44)	(€4.24)	€5.86	(€10.17)	€2.37	(€4.11)	€8.07	€1.89	€7.79	**€10.75**	€13.68	€6.28
15															
16															

μ=1	E[-cᵢ]														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	**€4.68**	€2.36	(€0.39)	€0.36	(€2.73)	€6.54	(€8.22)	€3.60	(€2.74)	€9.19	€3.14	€9.69	€8.25	€15.41	€6.20
3	€4.96	**€2.48**	(€0.24)	€0.61	(€2.44)	€6.78	(€7.88)	€4.15	(€2.16)	€10.12	€4.12	€11.34	€10.18	€16.97	€6.20
4	€3.86	€1.38	**(€0.31)**	€0.49	(€2.58)	€6.66	(€8.04)	€3.89	(€2.44)	€9.67	€3.65	€10.54	€9.25	€16.22	€6.20
5	€5.68	€3.53	€0.99	**€0.68**	(€2.35)	€8.75	(€7.77)	€4.33	(€1.97)	€10.41	€4.43	€11.86	€10.79	€17.47	€6.20
6	€7.00	€5.08	€2.82	€5.77	**(€2.18)**	€11.68	(€7.57)	€4.64	(€1.64)	€10.95	€4.99	€12.81	€11.92	€18.37	€6.20
7															
8	€4.90	€2.61	(€0.09)	€0.86	(€2.15)	€7.02	**(€7.89)**	€4.14	(€2.17)	€10.09	€4.09	€11.29	€10.13	€16.93	€6.20
9															
10	€4.17	€1.75	(€1.09)	(€0.84)	(€4.14)	€5.41	(€9.87)	€0.95	**(€2.36)**	€9.80	€3.78	€10.77	€9.51	€16.43	€6.20
11															
12	€3.96	€1.50	(€1.39)	(€1.33)	(€4.73)	€4.93	(€10.56)	(€0.15)	(€6.69)	€2.85	**€3.69**	€10.61	€9.33	€16.29	€6.20
13	€4.23	€1.82	(€1.01)	(€0.70)	(€3.97)	€5.54	(€9.68)	€1.27	(€5.20)	€5.24	(€1.01)	**€10.81**	€9.56	€8.75	€6.20
14	€4.53	€2.17	(€0.61)	(€0.01)	(€3.17)	€6.19	(€8.73)	€2.78	(€3.60)	€7.80	€1.68	€7.23	**€9.81**	€13.07	€6.20
15															
16															

Table 25: Optimal Policy Vector

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
μ	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0

Next, we computed the optimal allocation of capital ***K***=70 euros among the remaining test case ,starting from test cases 8 such that the probability of realizing test case 14 on or before schedule is maximized. We considered r_{ij} as shown in Table 26.

Table 26: Additional investment to reduce the expected time to complete execution of test case 'i'

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	M	C	M	S	S	C	S	C	S	C	M	S	S	S
rᵢ	€6.00	€6.00	€9.00	€6.00	€4.00	€1.00	€12.00	€1.00	€11.00	€1.00	€10.00	€6.00	€1.00	€1.00	€1.00

Moreover, we defined the marginal decrease in test execution time per unit increase in investment as shown in Table 27.

Table 27: Marginal decrease in test execution time per unit increase in investment

q_{ij}	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	0.015	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0.015	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.01	0	0.01	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0.015	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0.02	0	0	0	0	0	0	0	0.02
7	0	0	0	0.02	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0.01	0.01	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0.02	0	0	0	0	0	0
10	0	0	0	0	0	0		0	0	0.01	0.01	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0.02	0	0	0	0
12	0	0	0	0	0	0	0	0		0	0	0.01	0	0.01	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0	0
14	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0.02
15	0	0	0	0	0	0	0	0	0	0	0	0.02	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

We assumed the billing rate per hour, $m = 40$ euros. The different execution paths from test case 8 to test case 14 are shown in Table 28.We computed the probability of reaching test case 14 along the different execution as shown is Table 29. As expected the probability of execution path π_{14}^{4} is the highest. Finally, we solved the LP problem in (28) obtain the optimal additional investment along the execution paths from test case 8 to test case 14.As can be seen in Table 30, test case 10 requires the highest investment of 10 euros as its is a complex test case with many branches, followed by test case 12 and 13. In contrast to the additional investment assumed in table 26, the optimal solution does not require any additional investment in test cases 8,9,11 and 14.

Table 28: The possible execution paths from test case 8 to test case 14

π_{14}^{1}	8	9	10	11	12	13	14	
π_{14}^{2}	8	10	11	12	13	14		
π_{14}^{3}	8	9	10	12	13	14		
π_{14}^{4}	8	10	12	13	14			
π_{14}^{5}	8	9	10	11	12	15	13	14
π_{14}^{6}	8	10	11	12	15	13	14	
π_{14}^{7}	8	9	10	12	15	13	14	
π_{14}^{8}	8	10	12	15	13	14		

Table 29: Probability of reaching test case 14 along different paths

P_{14}^{1}	P_{14}^{2}	P_{14}^{3}	P_{14}^{4}	P_{14}^{5}	P_{14}^{6}	P_{14}^{7}	P_{14}^{8}
0.77%	2.45%	2.45%	7.35%	0.24%	0.77%	0.77%	2.33%

Table 30: The optimal additional investment

	8	9	10	11	12	13	14
	C	S	C	S	C	M	S
r_i	€ 0.00	€ 0.00	€ 10.00	€ 0.00	€ 3.41	€ 2.00	€ 0.00

9. Conclusion and Future Work

In this research paper we have formulated a stochastic model for system testing and test management which can be used by a test manager to optimize the tradeoff between test execution time and test execution cost. We have shown that test execution can be modeled as a counting process with test execution time being exponentially distributed. Moreover, we have extended the test execution model to a semi-Markov process where test execution time can have a general distribution. Using this model we have computed the expected time required for executing a test case and hence the execution time of all test cases in the test suite. The model also takes into account the complexity of each test case in order to compute the expected execution time. The results obtained provide a good estimation of the expected time to perform system testing and is more accurate than the results obtained from the SMC model.

We have further developed the semi-Markov process model of system testing to incorporate decisions wherein a test manager, based on the history of the system testing process, can decide whether an additional investment should or should not be made in a test case in order to ensure that the schedule constraints are satisfied. The results obtained in this paper contradict the usual assumption that additional investment in defect prevention activities is always optimal. In contrast, out results show that investment in developing the skill sets of the testers, such that their average test execution rate increases can be an optimal decision in some situations. Finally, our model allows the test manager to compute the minimum additional investment that is required in a chain of test cases such that the probability of completing test execution within as specified time is maximized. Our results show that the highest investment should be made in test cases which belong to the complexity class ‘C’ and in turn are parent to other complex test cases.

This paper assumes that test cases are being executed by only one tester. We intend to extend the model to multiple testers and therefore compute the execution time and the minimum cost of test execution. It will also be interesting to investigate into skill based assignment of test cases to testers and analyze the impact on time and costs. The model can also be extended to include a stochastic knapsack which can be used to model a defect tracking system used by testing teams wherein defects are logged as and when they occur and are picked up by developers to be fixed. This will help a test manager estimate the number of developers that should be allocated to fix defects detected by the testing team. Our model of system testing as a random walk on a finite graph also opens up the possibility of research in the area of mixing time, martingale property and optimal stopping of the system testing process.

Appendix

Monotone Convergence Theorem

Let $\{f_n(x)\}$ *be a sequence of functions in* $\sum^+$ *(the class of non negative simple functions) increasing to* $f(x)$ *i.e* $\limsup f_n(x) = f(x), \forall x \in X$ *and* $f_{n+1}(x) \geq f_n(x)$ *a.e.*

Then $f \in \sum^+$ *and* $\mu(f_n(x)) \uparrow \mu(f) \leq \infty$ *. This means* $\lim_{n\to\infty} \int f_n d\mu = \int \lim_{n\to\infty} f_n d\mu = \int f d\mu$

Jensen's Inequality

Let (Ω, F, P) *be a probability space, X an integrable real valued random variable and* φ *a convex function. Then* $\varphi(E[X]) \leq \mathrm{E}(\varphi(\mathrm{X})]$

Linearity Property of Expectation

Let $X_1, X_2 \ldots, X_n$ *be random variables, then* $E\left[\sum X_i\right] = \sum E[X_i]$

Tonelli's Theorem

Let (X, A, μ) *and* $(\mathrm{Y}, \mathrm{B}, \mathrm{v})$ *be* $\sigma-$*finite measure spaces and* $f : X \times Y \to [0, \infty)$. *Then* $\int_{A\times B} | f(x,y) | d(x,y) < \infty$ *and* $\int_A \left(\int_B f(x,y) dy \right) dx = \int_B \left(\int_A f(x,y) dx \right) dy = \int_{A\times B} | f(x,y) | d(x,y)$

Law of the unconscious statistician

Let F *be the cumulative distribution function of* X *, then the expected valued of* $g(X)$ *is given by*

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) dF(x) \ \forall x \in \mathbb{R}$$

References

[1] *Elmaghraby,Salah E. (1977), Activity Networks: Project Planning and Control by Network Models*

[2] *Osaki, Shunji & Mine,Hisashi(1968), Linear Programming Algorithms for Semi-Markovian Decision Processes*

[3] *Whittaker,James A. & Thomason,Michael G.(1994), A Markov Chain Model for Statistical Software Testing*

[4] *Changussu,Joao W.;DeCarlo,Raymond A. & Mathur,Aditya P. (2002), A Formal Model for Software Test Processes*

[5] *Pritsker,A.A.B.(1966),GERT:Graphical Evaluation and Review Technique*

[6] *Downs,Thomas(1985),An Aprroach to the Modeling of Software Testing with Some Applications*

[7] *Koole,G. and Bhulai,S(2011),Stochastic Optimization*

[8] *Lavenberg,S.S and Shedler,G.S.(1976),Stochastic Modeling of Processor Scheduling with Application to Database Management Systems*

[9] *Sarkar,Purnamitra,Random Walks on Graphs: An overview*

[10] *Mitra,A.(2007), A Case Study for Package Testing*

[11] *Cangussu,Joao W.,A Stochastic Control Model of the Software Test Process*

[12] *Sigman,Karl ,Simulating Markov Chains*

[13] *Gimbert,Hugo,Pure stationary Optimal Strategies in Markov decision processes*

[14] *Borza,Mojtaba;Rambely,Azmin Sham & Saraj,Mansour(2012),Solving Linear Fractional Programming Problems with Interval Coefficients in the Objective Function. A New Approach*