Connecting Software Reliability Growth Models to Software Defect Tracking

Maskura Nafreen, Melanie Luperon, and Lance Fiondella

Department of Electrical and Computer Engineering
University of Massachusetts Dartmouth
North Dartmouth, MA 02747, USA
Email: {mnafreen, mluperon, lfiondella}@umassd.edu

Vidhyashree Nagaraju
Tandy School of Computer Science
University of Tulsa
Tulsa, OK 74104, USA
Email: vidhyashree-nagaraju@utulsa.edu

Ying Shi
Goddard Space Flight Center
National Aeronautics and Space Administration
Greenbelt, MD 20771, USA
Email: ying.shi@nasa.gov

Thierry Wandji
Cyber Warfare Detachment
Naval Air Systems Command
Patuxent River, MD, 20670, USA
Email: ketchiozo.wandji@navy.mil

Abstract—Traditional software reliability growth models only consider defect discovery data, yet the practical concern of software engineers is the removal of these defects. Most attempts to model the relationship between defect discovery and resolution have been restricted to differential equation-based models associated with these two activities. However, defect tracking databases offer a practical source of information on the defect lifecycle suitable for more complete reliability and performance models. This paper explicitly connects software reliability growth models to software defect tracking. Data from a NASA project has been employed to develop differential equation-based models of defect discovery and resolution as well as distributional and Markovian models of defect resolution. The states of the Markov model represent thirteen unique stages of the NASA software defect lifecycle. Both state transition probabilities and transition time distributions are computed from the defect database. Illustrations compare the predictive and computational performance of alternative approaches. The results suggest that the simple distributional approach achieves the best tradeoff between these two performance measures, but that enhanced data collection practices could improve the utility of the more advanced approaches and the inferences they enable.

Keywords—Software reliability, software reliability growth model, software defect lifecycle, defect resolution, Markov modeling

I. INTRODUCTION

Software reliability growth models (SRGM) [1] offer to complement the software testing process in order to quantify the decreasing trend in times between the discovery of defects and the corresponding increase in reliability. Myriad papers call out the limitations of these models such as the assumption made by nonhomogeneous Poisson process (NHPP) SRGM that discovered defects are corrected immediately and no additional defects introduced. The many papers offering incremental advancements overshadow more innovative studies that attempt to establish concrete connections to the activities performed as part of software engineering processes. Despite these efforts, theory and practice have diverged, missing opportunities to drive meaningful advances. Marginal modeling efforts that do not explicitly validate underlying theories and poor data collection practices are two primary impediments. Modelers need to characterize the activities performed by processes and practitioners need to implement controls to capture data more consistently and accurately. The ideal outcome is a virtuous cycle in which modelers equip practitioners with practical inferences of high concern to inform process improvement and practitioners enhance data collection to support the livelihood of modelers.

Early studies that have attempted to characterize software defect discovery and resolution include the work of Schneidewind [2] who modeled defect discovery with a discrete exponential mean value function and resolution with a time lag. Xie and Zhao [3] subsequently extended Schneidewind’s model, assuming a defect resolution rate proportional to the number of defects discovered but not yet resolved and also demonstrated the applicability of the Poisson thinning process to model the difference between defects discovered but not yet resolved. Ohba [4] proposed the inflexion S-shaped model to characterize the fact that some defects may need to be discovered and resolved before others can be discovered, while and Yamada and Osaki [5] introduced the delayed S-shaped model, which incorporates a time delay to model this dependence. Similarly, Kapur and Younes [6] modeled leading and dependent defects. Yamada et al. [7] proposed two SRGM with imperfect debugging, where new defects are sometimes introduced when resolving others. Gokhale et al. [8] used a non-homogeneous continuous time Markov chain to model various repair scenarios, analyzing the effect of fault removal policies on the number of defects remaining at the end of the test.

Later studies, include the work of Gokhale and Mullens [9] who developed multi-priority queuing models for the software
defect resolution process, considering the effect of queuing system structures, priority levels, and priority disciplines on the differential mean times to resolution of defects of different severities. Shibata et al. [10] proposed an infinite server queueing model for the defect discovery and resolution process. Huang et al. [11] showed how several existing SRGM can be derived by applying the time-dependent delay function, while Lo and Huang [12] proposed an integrated defect discovery and resolution process modeling framework, in which the defect resolution process is expressed in terms of a time-varying resolution intensity and the difference between the number of defects discovered and resolved. Ullah et al. [13] performed a comparative study of SRGM on discovery and resolution data from dozens of open source software projects. Liu et al. [14] proposed a method to estimate the parameters of a defect removal model from semi-grouped data, which includes approximate times of when defects were discovered and resolved, while Yang et al. [15] model defect detection and correction of multi-release open source software and related optimal release problems. Cinque et al. [16] proposed debugging-workflow-aware SRGM to leverage debugging data managed by companies in bug tracking systems in order to improve accuracy when debugging fails to completely satisfy modelling assumptions. Okamura and Dohi [17] proposed a generalized bivariate fault detection and correction process as well as a model with hyper-Erlang distributions and expectation maximization algorithm to estimate its parameters. Vizarretta et al. [18] found that the infection S-shaped model best characterized the defect resolution of four releases of an open source software defined network controller and subsequently [19] quantified the cumulative distribution function of time to resolution according to defect severity.

This paper develops defect resolution models based on a data set from a NASA project [20]. Specifically, a differential-equation based model from the framework of Lo and Huang [12] as well as distributional and Markovian models of the defect lifecycle, inspired by the future research recommended in Gokhale and Mullen [9]. These models are created from information within the defect tracking database. The distributional model employs censoring to estimate model parameters when some defects have been discovered but not yet resolved, which is necessary to apply the models and make predictions during the testing and defect resolution process. These methods are applied to answer the following two previously unexplored research questions: (i) Does the Markovian model of defect resolution exhibit greater predictive accuracy than the simpler distribution model of defect resolution? and (ii) Is the defect lifecycle modeled by a Markov chain better characterized by a first or second order Markov chain? That is, do transitions in the defect lifecycle depend only on the present state or does information on the present and prior state determine the next state of the defect lifecycle? To answer (i), we apply the distributional and Markovian models to the data periodically throughout the testing process and compare their predictive accuracies, whereas we employ a formal hypothesis test to answer (ii). Our results indicate that, (i) the Markovian model of defect resolution attains only slightly better predictive accuracy and may therefore not be worth the additional computations. However, the Markovian model provides insight into the defect lifecycle and improved data collection practices may increase its attractiveness. The results also indicate that (ii) a one-step transition probability matrix (first order Markov chain) is sufficient to characterize the transitions in the defect lifecycle. These results are specific to the data set analyzed and results may vary. Moreover, we discuss limitations of the data and the steps we took to conduct our analysis in spite of these deficiencies. Regardless of the results obtained on the data set considered, the methods offer greater insight into the software defect discovery and resolution processes. Methods to improve data quality and that could support enhanced modeling to guide software testing are noted throughout.

Our primary contribution includes joint modeling of defect discovery and resolution that

- Does not assume defect resolution times follow a prescribed parametric form with no explanation in terms of the underlying process associated with defect resolution
- Enables an arbitrary defect lifecycle to be characterized by a distribution or semi-Markov process
- Can be applied during the defect detection and correction process to track progress

The remainder of the paper is organized as follows: Section II explains the need to distinguish defect discovery and resolution times, while Section III covers defect discovery and resolution models as well as a hypothesis testing method for the order of a Markov chain. Section IV illustrates the defect resolution models. Section V concludes and suggests future research directions.

II. SOFTWARE DEFECT DISCOVERY AND RESOLUTION: CONCEPTS

For clarity of exposition, Figure 1 shows the less detailed case of defect times data in order to explain many of the core concepts contained in past software defect discovery and resolution models.

![Fig. 1: Software defect discovery and resolution concepts](image)

The upper curve in Figure 1 represents the total defects discovered by the end of the $i^{th}$ interval, which is the cumulative
number of defects ever discovered. This corresponds to the counting process \( N(t) \) to which software reliability growth models are typically applied. However, basing reliability estimates on the non-discovery of additional defects makes strong implicit assumptions about the scope and completeness of testing. Therefore, a more pragmatic approach also employs the data in the bottom left of the figure with white bars, which represents the number of defects discovered but not yet resolved according to their severity, while the curve in the bottom right of the figure with grey bars represents the cumulative number of defect that have been both discovered and resolved \( N_r(t) \). Ideally, the number of unresolved defects goes to zero soon after all defects have been discovered, since the goal of software testing is to both detect and remove defects. Thus, Figure 1 indicates that models that only consider \( N(t) \) do not properly distinguish between detection and resolution and are therefore inadequate to assure software reliability. To support more detailed models of defect discovery and resolution, a natural resource is the defect tracking database, which documents the lifecycle of individual defects, enabling explicit connections between the software process and activities performed. Toward this end, each model developed in the following section explains how data from the defect tracking database can be transformed in support of parameter estimation.

### III. SOFTWARE DEFECT DISCOVERY AND RESOLUTION: MODELS

This section describes alternative methods to model the discovery and resolution of software defects recorded in a defect tracking database, including novel models based on the integrated defect discovery and resolution process modeling framework proposed by Lo and Huang [12] as well as distributional and Markovian resolution models developed here. These defect resolution models enable practical inferences of interest to the software practitioner that can be made throughout the software testing process, including the expected time required to remove (i) all defects discovered up to the present time \( t \) and (ii) all defects discovered up to the present time \( t \) as well as all defects anticipated to be discovered.

#### A. Integrated defect discovery and resolution processes

In the integrated defect discovery and resolution processes modeling framework [12], the rate of defect discovery and resolution are respectively expressed as differential equations possessing forms

\[
\frac{dm(t)}{dt} = \lambda(t)(a - m(t)) \tag{1}
\]

and

\[
\frac{dm_r(t)}{dt} = \lambda_r(t)(m(t) - m_r(t)) \tag{2}
\]

where \( m(t) \) (\( m_r(t) \)) denotes the mean value function of the number of defects discovered (resolved) by time \( t \), \( \lambda(t) \) \( \lambda_r(t) \) the defect discovery (resolution) rate, and \( a > 0 \) the number of defects that would be discovered with infinite testing. Thus, Equation (2) expresses the instantaneous rate of change in defect resolution as the product of the defect resolution intensity multiplied by the defects discovered but not yet resolved by time \( t \).

Lo and Huang [12] derived general forms of the solutions for Equations (1) and (2) with initial conditions \( m(t) = 0 \) and \( m_r(t) = 0 \) to produce expressions for the mean value function of defect discovery and resolution. They subsequently applied these general forms to explicitly show that \( \lambda(t) = \lambda_r(t) = b \) is the special case where the defect discovery and resolution models are characterized by the Goel-Okumoto [21] and Yamada Delayed S-shaped models [5] respectively. They also derived a novel model with unequal discovery and resolution rates \( \lambda(t) = b \) and \( \lambda_r(t) = c \). However, additional discovery and resolution models for the inflection S-shaped [4] as well as exponential and Rayleigh [22] as well as Weibull [23] testing effort functions were only specified implicitly in terms of \( \lambda(t) \) and \( \lambda_r(t) \).

We applied the Goel-Okumoto, Weibull, Yamada Delayed S-shaped, inflection S-shaped, Jelinski-Moranda [24], and Geometric model [25] to a recent NASA data set [20] and found that the inflection S-shaped model best characterized the defect discovery process. Therefore, we derived two explicit forms of \( m_r(t) \) with Equation (2), assuming defect discovery is modeled by the inflection S-shaped model, possessing mean value function

\[
m(t) = \frac{a}{1 + ce^{-bt}} \tag{3}
\]

where, \( b \) is the constant defect discovery rate, \( c \) is the inflection parameter

\[
c = \frac{1 - r}{r}, \quad r \in (0, 1]. \tag{4}
\]

and \( r \) is the inflection rate. As \( r \) approaches 1.0, the inflection S-shaped model reduces to \( m(t) = a(1 - e^{-bt}) \), which is the form of the Goel-Okumoto model.

The first form of the defect resolution model assumes defect resolution intensity \( \lambda_r(t) = b \), is the same as parameter \( b \) present in Equation (3), producing

\[
m_r^b(t) = a \left( 1 - e^{-bt} + (1 + c) \log \left( \frac{1 + c}{c + e^{bt}} \right) e^{-bt} \right) \tag{5}
\]

The second form of the defect resolution model introduces an additional parameter through \( \lambda_r(t) = d \), which implies that the defect resolution intensity is distinct from parameter \( b \) in Equation (3), producing

\[
m_r^d(t) = \frac{a}{c(b + d)} \left( 2F_1 \left( \frac{1,b+d}{b};2 + \frac{d}{b};-\frac{1}{ce^{-bt}} \right) e^{bt} - (b + d) \right)
\]

\[
\frac{2F_1 \left( \frac{1,b+d}{b};2 + \frac{d}{b};-\frac{1}{ce^{-bt}} \right) e^{bt} - (b + d) \right) - d \frac{2F_1 \left( \frac{1,b+d}{b};2 + \frac{d}{b};-\frac{1}{ce^{-bt}} \right) e^{bt} - (b + d) \right)}{c(b + d)} \tag{6}
\]
where \( F_1(a, b, c, z) = \sum_{k=0}^{\infty} \frac{(a)_k(b)_k}{(c+k)_k} \frac{z^k}{k!} \) is the hypergeometric function with \((q)_k = q(q+1) \ldots (q+k-1)\) for \(k > 0\).

Given a defect database possessing discovery and resolution times, \( T = (t_1, t_2, \ldots, t_n) \) and \( T^r = (t'_1, t'_2, \ldots, t'_n) \), the defect detection model in Equation (3) is applied with \( T \), while the defect resolution models in Equations (5) and (6) may be applied directly to \( T^r \).

### B. Distributional Approach

An alternative and potentially simpler method to model the time to correct defects is to compute the times between discovery and resolution and fit a distribution to these statistics. To make resolution time predictions possible throughout the software testing process, it is necessary to rely on the discovery and resolution times available up until time \( t \). However, at any specific time, the number of defects resolved may be strictly less than the number of defects discovered. This discrepancy requires the use of maximum likelihood estimation techniques for censored data [26]. Specifically, the problem of identifying a distribution to best fit the defect resolution times, when only the number of defects discovered and resolved before time \( t \), and \( t \) not yet resolved, \( \theta \) is the vector of parameters of the distribution being fit to the data. Moreover, \( t(i) \) denotes the time between discovery \( t(i) \) and resolution \( t'_i \) of the \( i \)th defect such that \( t(i) = t'_i - t_i \), whereas the term \( 1 - F(T - t_i) \) is the probability that a defect discovered at time \( t_i \) has not been resolved by time \( T \).

In practice, Equation (7) can be maximized with multiple alternative distributions and the one achieving the best goodness of fit employed to make predictions. One may then compute the mean of this distribution of best fit, which can be interpreted as the mean time to resolve defects. In this manner, the MVF of the number of defects resolved by time \( t \) may be expressed as

\[
m_s(t) = m(t - E[T_i])
\]

which translates the mean value function of the defect discovery process to the right by the mean time to resolve defects \( E[T_i] \).

### C. Markovian Approach

A Markov chain [27] is a stochastic process characterized by a sequence of states \( X_i \) which takes values from the finite set of \( m \) possible states. In the context of software defect tracking, the states represent the possible stages of the lifecycle from discovery to resolution. Thus, there is a one-to-one correspondence between the states of the defect lifecycle and the one-step transition probability matrix of the Markov Chain.

The first-order Markov hypothesis states that the present state at time \( t \) is conditionally independent of those up to and including time \( (t - 2) \) and only depends on the state at time \( (t - 1) \) such that

\[
P(X_t = i_0 | X_0 = i_1, \ldots, X_{t-2} = i_1) = P(X_t = i_0 | X_{t-1} = i_1) = q_{i_1,i_0}(t)
\]

where \( i_1, i_0 \in \{1, \ldots, m\} \). Assuming the state transition probability \( q_{i_1,i_0}(t) \) is time-invariant reduces to \( q_{i_1,i_0} \), producing a homogeneous Markov chain.

Considering all combinations of \( i_1 \) and \( i_0 \), we construct the transition matrix

\[
Q = \begin{pmatrix}
1 & \cdots & m \\
q_{1,1} & q_{1,2} & \cdots & q_{1,m} \\
\vdots & \vdots & \ddots & \vdots \\
q_{m,1} & \cdots & q_{m,m}
\end{pmatrix}
\]

where rows denote the previous state \( (X_{t-1}) \), columns the successor state \( (X_t) \), and each row sums to 1.0. In practice, the entries \( q_{i_1,i_0} \) may be determined from the information contained in the defect tracking database by computing the number of times state \( i \) transitioned to state \( j \) and dividing by the total number of transitions out of state \( i \). This is possible because each row of the defect tracking database provides distinguishable data on the lifecycle (sequence of state transitions) within the defect tracking model.

To compute the average time between defect discovery and resolution, it is necessary to compute (i) the average number of visits to each state of the defect lifecycle and (ii) the average time spent in each state \( i \) before transitioning to state \( j \). This first step is accomplished by solving for the fundamental matrix by setting entry \( q_{m,m} = 0 \), and computing

\[
S = \sum_{k=0}^{\infty} Q^k = (I_m - Q)^{-1}
\]

where \( I_m \) is the identity matrix of dimension \( m \times m \). Entry \( s_{i,j} \) is the average number of visits to state \( j \) given that defect tracking began in state \( i \). Thus, \( s_{1,j} \) is the average number of visits to the \( j \)th state of the defect tracking lifecycle. Assuming a unique starting state, this notation can be simplified to \( s_j \) without loss of generality.

The second step determines the distribution of each transition from state \( i \) to state \( j \) \( (T_{i,j}) \) with information contained in the rows of the defect tracking database. Specifically, the time at which each state was entered is recorded as a timestamp. Therefore, it is possible to sort these times to determine the sequence of one-step transitions taken by the defect from discovery to resolution as well as the time between each of these transitions. Computing these transition time statistics for all rows of the database provides multiple observations for each \( (i, j) \) transition. A distribution of best fit is then determined with traditional statistical methods and the expectation of this distribution \( E[T_{i,j}] \) is then computed.
The average time from defect discovery to resolution according to the Markovian approach may be expressed as follows

\[
E[T_t] = \sum_{i=1}^{m} s_i \sum_{j=1}^{m} q_{i,j} E[T_{t,j}]
\]  

(11)

where the inner sum weights the average time to transition from state \( i \) to state \( j \) \((E[T_{t,j}])\) by the probability of taking this transition out of state \( i \) \((q_{i,j})\), while the outer sum weights these transition times out of state \( i \) by the average number of visits to state \( i \) \((s_i)\). This alternative value of \( E[T_t] \) may be substituted into Equation (8) for the mean number of defects resolved by time \( t \).

D. Higher-order Markov Chains

This section describes higher-order Markov chains, maximum likelihood estimation of transition probabilities, and a hypothesis test to identify the order of the Markov chain.

In some circumstances [28], the present state depends on multiple previous states. When the present state depends on the last \( k \) states, a \( k\)th-order Markov chain with transition probabilities of the form

\[
P(X_t = i_0 | X_0 = i_t, ..., X_{t-1} = i_1) = P(X_t = i_0 | X_{t-k} = i_k, ..., X_{t-1} = i_1) = q_{i_k...i_0}(t)
\]

is appropriate.

For example, a second-order (\( k = 2 \)) Markov chain composed of \( m = 3 \) three states is characterized by the transition matrix

\[
Q = \begin{array}{ccc}
X_{t-2} & X_{t-1} & 1 & 2 & 3 \\
1 & 1 & q_{111} & q_{112} & q_{113} \\
2 & 1 & q_{211} & q_{212} & q_{213} \\
3 & 1 & q_{311} & q_{312} & q_{313} \\
1 & 2 & q_{121} & q_{122} & q_{123} \\
2 & 2 & q_{221} & q_{222} & q_{223} \\
3 & 2 & q_{321} & q_{322} & q_{323} \\
1 & 3 & q_{131} & q_{132} & q_{133} \\
2 & 3 & q_{231} & q_{232} & q_{233} \\
3 & 3 & q_{331} & q_{332} & q_{333}
\end{array}
\]

Each possible combination of \( k \) successive observations of the defect tracking process \( X \) is a state of the model such that the number of states is equal to \( m^k \). Moreover, when the order is greater than one, the transition matrix \( Q \) contains several elements corresponding to transitions that cannot occur because the state \( X_{t-1} \) becomes \( X_{t-2} \) at the following time step. Thus, it is impossible to transition from the row where \( X_{t-1} = i \), but \( X_{t-2} \) \( \neq i \) in the subsequent state. Since the probability of such transitions is 0, they are omitted from \( Q \), which is written more compactly above than a square matrix possessing structural zeros.

Regardless of the order \( k \), each row of the matrix \( Q \) possesses \((m - 1)\) independent probabilities, since each row is a probability distribution summing to 1.0 and the \( m^{kth} \) outcome is completely determined by the others. Therefore, the total number of independent parameters to be estimated is \( m^k(m - 1) \). Letting \( n_{i_k...i_0} \) denote the number of times the subsequence of \( k \) states \( X_{t-k} = i_k, ..., X_{t-1} = i_1, X_t = i_0 \) is observed in the data, the maximum likelihood estimate of the corresponding transition probability \( q_{i_k...i_0} \) is

\[
\hat{q}_{i_k...i_0} = \frac{n_{i_k...i_0}}{n_{i_k...i_1}}
\]

(13)

where

\[
n_{i_k...i_0} = \sum_{i_0=1}^{m} n_{i_k...i_0}
\]

is the total number of subsequences that begin with \( n_{i_k...i_0} \) and end with any state \( X_t = i_0 \).

It is possible to compute a \( \chi^2 \) statistic in order to identify the order of a Markov Chain. The \( \chi^2 \) statistic quantifies how expectations compare to observed data under different model assumptions. The most elementary hypothesis test on the order of a Markov chain [29] poses the following null and alternative hypotheses:

- \( H_0 : m = 1 \quad \text{The Markov chain is first order} \)
- \( H_1 : m = 2 \quad \text{The Markov chain is second order} \)

To test the significance of dependence of state \( i_0 \) on \( q_{i_2...i_0} \), the \( \chi^2 \)-test can be used to assess the disagreement between \( q_{i_2...i_0} \) and \( q_{i_1...i_0} \) as

\[
\chi^2 = \sum_{i_2...i_0} n_{i_2...i_0}^*(q_{i_2...i_0} - \hat{q}_{i_1...i_0})^2
\]

(14)

where

\[
n_{i_2...i_0}^* = \sum_{i_0=1}^{m} n_{i_2...i_0}
\]

(15)

and degrees of freedom

\[
df = (r-1)(c-1)
\]

(16)

where the rows \( r \) and columns \( c \) respectively correspond to the unique sequences \( q_{i_2...i_0} \) and \( q_{i_1...i_0} \) taken by defects through the defect lifecycle.

If the null hypothesis is rejected in favor of the alternative hypothesis, then the higher-order transition matrix \( Q \) can be expanded to include structural zeros and Equations (10) and (11) of the Markovian approach applied to estimate the mean time to resolve defects with a higher-order Markov chain.

IV. ILLUSTRATIONS

This section illustrates the approaches to defect resolution modeling developed in Section III in order to assess their relative predictive ability and computational efficiency. Section IV-A describes the NASA defect tracking database, steps taken to clean the data, and resulting the Markov model of the software defect tracking process. Section IV-B tests the order of the Markov model, according to the methods described in Section III-D.
To emphasize the need for software reliability modeling that can be used during the defect and resolution process, two cases are considered: (i) retrospective analysis, characteristic of many historical defect discovery modeling papers, which used all available data and (ii) online analysis, which periodically updates estimates throughout this process to refine estimates and track progress. Section IV-C presents a comparative analysis of the alternative defect resolution models to the complete data and explains the steps taken to apply the Distributional and Markovian approaches. Section IV-D subsequently presents the corresponding analysis of each model’s empirical defect resolution intensity with the complete data. Section IV-E presents online predictive analysis, discussing the tradeoffs between model accuracy and computational efficiency.

A. Data Description, Cleaning, and Modeling

The project database is a rich source of information to model the software defect tracking lifecycle. The database consists of \( n = 1933 \) rows, each of which corresponds to a defect, including the times at which the defect entered a subset of the 13 possible states listed in Table I.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Created</td>
<td>7</td>
<td>In Work</td>
</tr>
<tr>
<td>1</td>
<td>Assigned</td>
<td>8</td>
<td>On Hold</td>
</tr>
<tr>
<td>2</td>
<td>Build Integration</td>
<td>9</td>
<td>Ready for Closure</td>
</tr>
<tr>
<td>3</td>
<td>Canceled</td>
<td>10</td>
<td>Ready for Test</td>
</tr>
<tr>
<td>4</td>
<td>Closed</td>
<td>11</td>
<td>Test Complete</td>
</tr>
<tr>
<td>5</td>
<td>Closed with Defect</td>
<td>12</td>
<td>Work Completed</td>
</tr>
<tr>
<td>6</td>
<td>In Test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For each row in the database, the subset of columns with non-null entries indicated the sequence of states taken by that particular defect. Thus, sorting this subset of states in ascending order and computing the difference between successive times respectively determined the sequence of states and transition times. These state transitions provided the feasible one-step transitions within a Markov model of the software defect tracking lifecycle. However, several rows exhibited one or more of the following problems:

- State sequences violated the logical order in which work would be performed, including very rarely observed sequences as well as degenerate sequences that were incomplete.
- Necessary states were not recorded.
- Multiple successive states possessed transition times equal to zero, suggesting that progression of the defect through the lifecycle was not properly measured. The most natural explanation is that software engineers simply entered the same time into the database because the process required documenting the defect was resolved.

Ultimately, 1478 rows were eliminated for the reasons described above, leaving just 455 or approximately 25% of the data for analysis. While this remaining data is by no means complete and may introduce statistical bias, it is a sufficiently large sample to illustrate some possible inferences that would be possible. In doing so, we seek to justify the institutionalization of better data collection practices. For example, the database also contained information on the software build (version), and defect type (defects vs. change requests). However, the reduction in sample size resulting from inconsistent data collection practices and lack of mechanisms to ensure data quality prevented this level of analysis. Some concrete strategies to improve data quality would be to (i) provide formal training on the defect lifecycle tracking software and (ii) include contractual requirements specifying the minimum precision of information recorded.

Figure 2 shows the state transition model of the software defect tracking process after cleaning the dataset.
Medium severity defects were the most abundant \( (n_1 = 381) \), while fewer low \( (n_0 = 61) \) and high severity \( (n_2 = 13) \) defects were documented. At the peak, around \( t = 1,200 \), over 40 medium severity defects had been discovered but not yet resolved. While the subsequent analysis is not decomposed by severity, Figure 3 provides a window into the complex activities underlying defect tracking and the corresponding opportunities for more detailed modeling.

![Fig. 3: Defect discovery and resolution by severity](image)

**B. Testing the Order of the Markov Chain**

This section applies the hypothesis test described in Section III-D to assess a first or second order the Markov chain would best characterize the transitions observed in the sequences of state transitions contained in the rows of the software defect tracking database used to construct Figure 2. Toward this end, Equation (13) was employed to obtain the maximum likelihood estimates of the second order transition probabilities \( q_{ij(lk)} \) as well as the first order transition probabilities \( q_{ij} \). These estimates were subsequently substituted into Equation (14) to obtain a \( \chi^2 \) value of 317.19, which exceeds the critical value 101.88 at the 0.05 level of significance of the \( \chi^2 \) distribution possessing 80 degrees of freedom, since there were respectively eleven and nine unique second and first order sequences such that \( df = (11 - 1)(9 - 1) \), according to Equation (16). Thus, the null hypothesis that the Markov chain is first order is rejected at the 0.05 level of significance in favor of the alternative hypothesis that the Markov chain is second order.

The second order transition sequences contributing to the rejection of the null hypothesis were further analyzed to determine if this result possessed an intuitive explanation or if further insight could be made regarding the defect tracking state descriptions given in Table I. To perform this analysis, Table II reports second order transition subsequences contributing to the \( \chi^2 \) statistic, the numerical value contributed by that term to Equation (14), and the percentage the term contributed to rejection of the null hypothesis. Table II indicates that the transition sequence 1 → 0 → 7 contributed most heavily to rejection of the null hypothesis. However, upon further examination of the meaning of these states in Table I, it was deemed contrary to the logical flow of the defect tracking process. Specifically, it did not seem appropriate to set a defect to *Assigned* before it was *Created*, raising concerns about the consistency of the development teams understanding and use of the defect tracking database. As a result, the subsequent examples apply the Markovian approach described in Section III-C with a first order Markov chain. The primary reasons for this decision are as follows. Although the order of state transitions found at the beginning of the Markov chain may be slightly out of order, (i) this may not significantly alter the performance analysis of the time from *Start to End* in Figure 2, (ii) the remainder of the state model contains valuable information on the process underlying defect tracking and corresponding resolution times, and (iii) eliminating these slightly out of order sequences would have substantially reduced the sample size, which would have prevented meaningful application of the Markovian approach. Moreover, despite these caveats related to data quality, the Markovian approach to defect resolution modeling may have merit if defect tracking guidelines are followed more consistently.

**TABLE II: Contribution of second order sequences to rejection of the null hypothesis (first order Markov assumption)**

<table>
<thead>
<tr>
<th>Paths</th>
<th>( \chi^2 )</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ( \rightarrow ) 1 ( \rightarrow ) 7</td>
<td>9.36</td>
<td>0.0295</td>
</tr>
<tr>
<td>0 ( \rightarrow ) 1 ( \rightarrow ) 8</td>
<td>0.92</td>
<td>0.0029</td>
</tr>
<tr>
<td>1 ( \rightarrow ) 0 ( \rightarrow ) 3</td>
<td>0.84</td>
<td>0.0026</td>
</tr>
<tr>
<td>1 ( \rightarrow ) 0 ( \rightarrow ) 7</td>
<td>30.286</td>
<td>0.0048</td>
</tr>
<tr>
<td>1 ( \rightarrow ) 0 ( \rightarrow ) 8</td>
<td>2.91</td>
<td>0.0092</td>
</tr>
<tr>
<td>1 ( \rightarrow ) 7 ( \rightarrow ) 3</td>
<td>0.31</td>
<td>0.0010</td>
</tr>
</tbody>
</table>

**C. Defect resolution models**

This example examines the accuracy of the alternative defect resolution modeling approaches.

1) *Retrospective analysis:* Figure 4 shows the defect discovery and resolution counting processes \( \{N(t)\} \) along with the corresponding fitted defect and resolution models as well as distributional and Markovian approaches.

![Fig. 4: Defect discovery and resolution processes with fitted defect models](image)
process (right step function) corresponds to the times at which defects were resolved.

For each of the defect resolution modeling approaches, Table III summarizes the sum of squares error (SSE) and runtime in seconds.

### Table III: Comparative analysis of defect resolution intensity

<table>
<thead>
<tr>
<th>Approach</th>
<th>SSE</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution process (m^R(t))</td>
<td>3.77 x 10^4</td>
<td>175.2</td>
</tr>
<tr>
<td>Resolution process (m^D(t))</td>
<td>2.96 x 10^4</td>
<td>1627.2</td>
</tr>
<tr>
<td>Distributional</td>
<td>3.43 x 10^4</td>
<td>150.1</td>
</tr>
<tr>
<td>Markovian</td>
<td>3.88 x 10^4</td>
<td>208.8</td>
</tr>
</tbody>
</table>

The model with the lowest SSE and runtime, namely the distributional approach, are shown in bold.

The maximum likelihood estimates of \( m_i^b(t) \) were \( \hat{a} = 465.13, \hat{b} = 0.004855 \), and \( \hat{c} = 101.605 \), while the maximum likelihood estimates of \( m_i^d(t) \) were \( \hat{\alpha} = 462.617, \hat{\beta} = 0.004191, \hat{\delta} = 167.91 \), and \( \hat{d} = 0.657733 \). Thus, the unique resolution rate \( d \) in Equation (6) contributes to the poor quality of the model fit because of the higher masking term \( c \), which corresponds to visible underprediction in the number of defects resolved in the interval \( t = (600, 1000) \) of Figure 4.

The mean time from defect discovery to resolution of the distributional approach was \( E[T_i] = 59.74 \), which was determined with the special case of Equation (7), where all data up to the time of the resolution of the \( n^{th} \) defect was available and the corresponding estimate for the Markovian approach from Equation (11) was \( E[T_i] = 81.48 \). Given that the Markovian approach required twice as much time to compute and still exhibited lower accuracy, this level of complexity may not be justified despite the additional process details underlying the model. Moreover, the accuracy and efficiency of the distributional approach may offer a competitive alternative to the resolution process (\( m_i^d(t) \)) of Equation (5).

2) Distributional approach: Seventeen possible distributions were considered, explaining why the calculation in Table III required 150 seconds. These distributions included the Beta, Birnbaum-Saunders, Exponential, Extreme value, Gamma, Generalized extreme value, Generalized Pareto, Inverse Gaussian, Logistic, Log-logistic, Lognormal, Nakagami, Normal, Rayleigh, Rician, \( t \) location-scale, and Weibull distributions. The generalized extreme value (GEV) distribution possessing the following maximum likelihood estimates \( T_{\mu} \sim GEV(\mu = 57.4437, \sigma = 22.6722, \xi = -0.0959) \) attained the best fit to the empirical times between defect discovery and resolution according to AIC [30] and BIC [31] measurements, as shown in Figure 5.

However, this retroactive approach is only useful for model assessment after all of the data has been collected and remains a pervasive problem in the defect discovery modeling literature, namely the over reliance on in sample measures of goodness of fit and the lack of online procedures that can be employed during the defect discovery and resolution process, which are explicitly addressed in Section IV-E.

### 3) Markovian approach: The defect tracking state model shown in Figure 2 contains 26 distinct transitions, each possessing a unique distribution. Since the path each defect takes through the state model can differ, some of these transitions were observed infrequently. In cases where five or fewer transitions were recorded the mean time between transitions was computed according to the arithmetic mean. Table IV reports the 16 transitions between states \( i \) and \( j \) with greater than five observations, the distribution that attained the best fit, sample size (\( n_{i,j} \)), and parameters of the model of best fit.

### Table IV: Empirical state transition distributions

<table>
<thead>
<tr>
<th>Transition</th>
<th>Distribution</th>
<th>( n_{i,j} )</th>
<th>Shape</th>
<th>Scale</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1</td>
<td>Generalized Pareto</td>
<td>62</td>
<td>0.36</td>
<td>7.39</td>
<td>0.08</td>
</tr>
<tr>
<td>0 3</td>
<td>Log-Logistic</td>
<td>8</td>
<td>0.94</td>
<td>2.16</td>
<td></td>
</tr>
<tr>
<td>0 7</td>
<td>Birnbaum-Saunders</td>
<td>12</td>
<td>2.34</td>
<td>7.30</td>
<td></td>
</tr>
<tr>
<td>0 8</td>
<td>Exponential</td>
<td>6</td>
<td>10.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>Rician</td>
<td>13</td>
<td>0.14</td>
<td>0.57</td>
<td></td>
</tr>
<tr>
<td>1 3</td>
<td>Exponential</td>
<td>6</td>
<td>71.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>Birnbaum-Saunders</td>
<td>33</td>
<td>1.91</td>
<td>14.78</td>
<td>11.29</td>
</tr>
<tr>
<td>2 9</td>
<td>Generalized Pareto</td>
<td>33</td>
<td>0.79</td>
<td>10.77</td>
<td>11.29</td>
</tr>
<tr>
<td>2 10</td>
<td>Birnbaum-Saunders</td>
<td>31</td>
<td>1.39</td>
<td>27.98</td>
<td></td>
</tr>
<tr>
<td>6 11</td>
<td>Nakagami</td>
<td>20</td>
<td>63723.31</td>
<td>0.46</td>
<td></td>
</tr>
<tr>
<td>7 12</td>
<td>Generalized Pareto</td>
<td>65</td>
<td>0.89</td>
<td>9.34</td>
<td>0.99</td>
</tr>
<tr>
<td>8 3</td>
<td>Exponential</td>
<td>6</td>
<td>141.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 6</td>
<td>Weibull</td>
<td>23</td>
<td>0.79</td>
<td>61.89</td>
<td></td>
</tr>
<tr>
<td>10 9</td>
<td>Birnbaum-Saunders</td>
<td>9</td>
<td>1.42</td>
<td>83.17</td>
<td></td>
</tr>
<tr>
<td>11 9</td>
<td>Generalized Pareto</td>
<td>40</td>
<td>0.82</td>
<td>49.89</td>
<td>0.99</td>
</tr>
<tr>
<td>12 2</td>
<td>Generalized Pareto</td>
<td>65</td>
<td>0.23</td>
<td>45.44</td>
<td>0.99</td>
</tr>
</tbody>
</table>

The sample sizes reported in Table IV were used to estimate the transition probabilities \( q_{i,j} \). For example, \( \hat{q}_{0,1} = 62 / 7045 = 0.0085 \) is the fraction of transitions from the Created (0) to Assigned (1) state. The expected number of visits to each state was then calculated with Equation (10). The mean time to transition between states \( E[T_{i,j}] \) was then calculated according to the formula for the mean of the distribution of best fit and corresponding numerical parameters or arithmetic mean. Finally, the average time from defect discovery to resolution was computed according to Equation (11).

### D. Defect resolution intensity

This section performs a comparative analysis of the resolution intensity functions of each model. We begin with an explanation of how the empirical intensity was computed.
In some cases, the fidelity of the defect tracking database was limited, containing instances where two or more successive defects were resolved at the exact same time. This led to points at which the empirical defect resolution intensity was infinity, since the reciprocal of the time between resolutions was $\frac{1}{0}$. To enable calculation of the empirical defect resolution intensity despite this limitation, the following more general expression was employed

$$\lambda_r(t_r) = \frac{N_r(t_r) - N_r(t_{r-1})}{\sum_{j \in R} t_{r_j} - t_j}$$ (17)

where $N_r(t_r) - N_r(t_{r-1})$ is the number of defects resolved at the $r^{th}$ unique resolution time $t_r$, $R$ the set of defects resolved (since defects are not necessarily resolved in the order they were discovered), and $t_{r_j} - t_j$ the time between the discovery of defect $j \in R$ which was simultaneously resolved at $t_r$. Thus, the denominator is the total time the defects were worked on prior to their resolution, since defects were not necessarily all discovered at the same time.

Figure 6 shows the step function associated with the empirical resolution intensity computed at each resolution time $t_r$, along with the resolution intensity of each fitted model, computed by differentiating the mean value of function of the defect resolution counting process with respect to $t$ and substituting the MLEs used to plot Figure 4 into these expressions.

![Defect resolution intensity](image1.png)

**Fig. 6: Defect resolution intensity**

With the exception of the model possessing unique resolution rate $d$ of Equation (6), the models track the empirical intensity well.

To compare the models quantitatively, Table V summarizes the sum of squares error between the empirical resolution intensity and fitted curves.

<table>
<thead>
<tr>
<th>Approach</th>
<th>SSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution process $(m_r^0(t))$</td>
<td>$5.627 \times 10^4$</td>
</tr>
<tr>
<td>Resolution process $(m_r^0(t))$</td>
<td>$5.665 \times 10^4$</td>
</tr>
<tr>
<td>Distributional</td>
<td>$5.595 \times 10^4$</td>
</tr>
<tr>
<td>Markovian</td>
<td>$5.486 \times 10^4$</td>
</tr>
</tbody>
</table>

The results indicate that the Markovian approach performed best, followed closely by the distributional approach and simpler resolution process possessing intensity $(\lambda_r(t))$.

E. Comparison of online predictive accuracy and computational efficiency

To further compare the alternative defect resolution models, this section conducts an online assessment of predictive accuracy, which also explicitly considers the computational efficiency of each approach. Starting at time $t = 200$, the defect resolution processes of Equations (5) and (6) were fit directly to the resolution time data extracted from the defect tracking database, while the distributional and Markovian approaches first identified an SRGM that best fit the available defect detection data, and then estimated the mean time to restore defects $(E[T_r])$ with Equations (7) and (11) respectively and substituted this value into Equation (8). In this manner each model, was used to predict the cumulative number of defects resolved 200 time units into the future or $t = 400$ and the percentage error computed between the actual number of defects observed and the predicted values. This process was repeated periodically for each model. However, Table III determined that the model with unique defect resolution rate $(m_r^0(t))$ was over an order of magnitude slower than the distributional approach. Thus, the frequency of predictions made by each model was chosen to be inversely portion to their runtime. This method was implemented to allocate approximately equal computational time to each method.

Figure 7 shows the results of online assessment of predictive accuracy explicitly considering computational efficiency.

![Comparison of error predicted by distributional and Markovian resolution time models](image2.png)

**Fig. 7: Comparison of error predicted by distributional and Markovian resolution time models**

Figure 7 indicates that the distributional approach achieves the lowest error as early as $t = 300$ and remains the most accurate throughout the remainder of the defect testing and resolution process. Moreover, the computational efficiency enables frequent updates, supporting regular online assessment throughout. Unlike the defect resolution processes, which rely solely on the defect resolution count data, the distributional approach explicitly incorporates information on defects discovered and resolved as well as those discovered but not yet resolved, whereas the Markovian approach provides even greater fidelity, including rows associated with defects that
traversed from *Start to End* in Figure 2 as well as defects that have only partially traversed the state model in Figure 2.

V. CONCLUSION

This paper connect software reliability growth models to defect tracking databases. Differential-equation based, distributional, and Markovian approaches were developed. A NASA defect tracking database was cleaned and a state model constructed. A statistical hypothesis test was applied to determine the order if a first or second order Markov chain best characterized the transitions taken during the defect lifecycle. Models were applied to the full data, referred to as retrospective analysis, and the steps taken to apply the distributional and Markovian approaches explained. The goodness of fit of alternative models was also assessed with respect to the defect resolution intensity. Finally, the predictive accuracy and computational efficiency were assessed in an online manner. The results suggested that most of the models fit the defect resolution data well, but that the distributional approach achieved the best fit and also demonstrated the lowest runtime, enabling more frequent updates, which would support online tracking when defect detection and resolution are ongoing. Despite the deficiencies in the data, which posed the most obvious threat to validity, the models demonstrated the potential benefit of renewed efforts to collect high quality data related to the defect tracking lifecycle.

Some useful directions to explore in the future include: (i) application of covariate models to determine the relative effectiveness of alternative activities employed to resolve defects and identify how to allocate resources to these activities efficiently and (ii) identify factors that could be used to modeling the cost to resolve defects to inform resource planning.

ACKNOWLEDGMENT

This work was supported by the National Aeronautics and Space Administration (NASA) under Grant Number (#80NSSC18K0154) and NSF award (#1749635). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES


